Learning for Feasible and Safe Control with Control Barrier Functions: A Tutorial

Wei Xiao, Calin Belta, and Christos G. Cassandras

***Abstract*—The Control Barrier Function (CBF) method is becoming a popular tool that transforms nonlinear constrained optimal control problems into a sequence of Quadratic Programs (QPs). In this tutorial paper, we show how to employ machine learning techniques to ensure the feasibility of these QPs, which is a challenging problem, especially for high relative degree constraints where High Order CBFs (HOCBFs) are employed. We present two complementary learning approaches: (i) parameter learning for regular unsafe sets; (ii) sampling learning for irregular unsafe sets, where "regularity" of an unsafe set is formally defined in terms of the dependence of QP feasibility on initial system conditions. The first approach compensates for the myopic nature of the QP-based approach by parameterizing the HOCBFs and using machine learning techniques to select parameters that maximize a feasibility robustness metric related to system performance. This feasibility robustness metric measures the extent to which QP feasibility is maintained in the presence of time-varying and unknown unsafe sets. The sampling learning approach addresses "irregular" unsafe sets in which the problem feasibility heavily depends on the initial conditions. This approach learns a new feasibility constraint that guarantees the QP feasibility, and it is then enforced by another HOCBF added to the QPs. The accuracy of the learned feasibility constraint can be recursively improved by the proposed recurrent training algorithm. We demonstrate the advantages of the proposed learning approaches to constrained optimal control problems with specific focus on a robot control problem and on autonomous driving in an unknown environment.**

***Index Terms*—Machine Learning, Lyapunov methods, Safety-Critical Control, Optimal Control.**

I. Introduction

With growing interest in autonomy, optimal control problems become increasingly important but challenging, especially in the presence of both safety constraints and control limitations since they may conflict with each other. It was recently shown that for nonlinear control systems that are affine in controls and cost functions that are quadratic in controls, an optimal control problem with safety constraints can be solved through a sequence of quadratic programs (QPs) that are implemented on-line. Central to this approach is the notion of forward invariance enforced using barrier functions (BF) [1], [2].

W. Xiao is with the Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology, Cambridge, MA, USA weixy@mit.edu

C. Belta is with the Department of Electrical and Computer Engineering and the Department of Computer Science, University of Maryland, College Park, MD, USA cbelta@umd.edu

C. G. Cassandras is with the Division of Systems Engineering and Center for Information and Systems Engineering, Boston University, Brookline, MA, 02446, USA cgc@bu.edu

This work was supported in part by NSF under grants 9500313838, 9500311326, and CNS-2149511 and by ARPAE under grant DE-AR0001282.

BFs are Lyapunov-like functions [3], [4], whose use can be traced back to optimization problems [5]. Control BFs (CBFs) are BFs for control systems that are used to map a constraint defined over system states onto a constraint on the control. Recently, it has been shown that, to stabilize an affine control system while optimizing a quadratic cost and satisfying state and control constraints, CBFs can be combined with Control Lyapunov Functions (CLFs) [6], [7], [8] to form quadratic programs (QPs) [9], [1], [10] that can be efficiently solved in real time. These CBFs work for constraints that have relative degree one. A more general form [11], termed exponential CBF, can work for arbitrarily high relative degree constraints. The high order CBF (HOCBF) proposed in [2] is simpler to use and more general than the exponential CBF. CBFs have been widely used in many robotic systems, such as in autonomous driving [12], [13], [14], soft robots [15], and swarm robots [16].

There are several remaining challenges to be addressed in the CBF-based method, including the determination of (possibly unknown) unsafe sets and (possibly unknown) system dynamics, as well as the feasibility of the associated QPs when both state constraints (enforced by HOCBFs) and control bounds are involved. To determine unsafe sets and system dynamics, machine learning techniques are commonly used. Supervised learning techniques have been proposed to learn safe set definitions from demonstrations [17], and sensor data [18], which are then enforced by CBFs. In [19], data are used to learn system dynamics for CBFs. In a similar setting, the authors of [20] use adaptive control techniques to estimate the unknown system parameters in CBFs. Neural network controllers are often trained using CBFs in the presence of disturbances [21]. However, these works did not consider the *feasibility* of the associated QPs in the presence of strict control bounds, which critically depends on how we define CBFs (or HOCBFs), i.e., the parameters involved in the definition of CBFs. In this paper, we focus on the feasibility of these CBF-based QPs and on the maximization of their future solution space.

Infeasibility can be improved by pre-computing feasible motion spaces [22] [23] or using receding horizon optimization as in Model Predictive Control (MPC) [24]. Optimal control methods can also avoid the infeasibility problem, but their computational complexity becomes prohibitive with non-linear dynamics and constraints. Furthermore, unknown environments make solving such problems even harder. Some approaches to improve feasibility for specific applications have been proposed. For the adaptive cruise control (ACC) problem defined in [1], the infeasibility issue is addressed by considering the minimum braking distance. However, this approach

does not scale well for high-dimensional systems. The penalty method proposed in [2] can improve the recursive feasibility of the QPs and scales well, but it often does not stabilize the system to desired equilibria when "irregular" unsafe sets (defined later) are involved. Feasibility guarantees for CBF-based QPs can also be achieved by finding explicit sufficient conditions [25] expressed themselves as CBFs; however, such conditions are usually hard to find for general constrained control problems.

Machine learning techniques to obtain feasible solutions were proposed for legged robots. Feasibility constraints are learned for probabilistic models in [26] based on simplified models. The learned constraints are complex, and they are simplified by expectation-maximization. Robot footstep limits are modeled as hyperplanes based on success and failure dataset in [27]. Reinforcement learning [28] [29] has the potential to address the infeasibility issue for optimal control problems, but it is difficult to quantify infeasibility as a reward, and the optimized parameters may also go to a local infeasible region where a feasible solution can never be found.

This paper presents a tutorial for *learning-based* CBF methods based on [30] [31]. The main consideration in this paper is to address the CBF-based QP infeasibility problem by avoiding all possible unsafe sets in unknown environments using machine learning techniques. Feasibility pertains to the CBF-associated QPs mentioned earlier, which are used in solving general-purpose optimal control problems. Unsafe sets are assumed to belong to a finite collection of sets whose geometries are known. However, their locations are only detected in real time operation. Our approach is twofold: first, we parameterize the definition of a HOCBF as in [32] and systematically optimize the parameters so as to guarantee recursive feasibility and to maximize the future solution space of the CBF-based QPs; alternatively, we proceed by learning a new feasibility constraint that guarantees feasibility.

The first technique in the parameter-learning approach is the maximization of a *feasibility robustness metric* that we define in terms of maximizing the future action space, i.e., maximally ensuring the feasibility of the CBF-based QPs in the presence of time-varying and unknown unsafe sets. In particular, by measuring robustness as a distance metric to an unsafe set, we aim at making the corresponding HOCBF constraint active *as late as possible* in the QPs. The main benefit of maximizing such a robustness metric is that the QP feasibility can be maintained when the unsafe sets are unknown and their detection is subject to noise. The second technique is to put forward a feasibility-guided method to learn the optimal parameters in a HOCBF corresponding to a specific type of unsafe set such that the robustness metric is maximized. However, this approach does not work well when the unsafe sets are such that the problem feasibility heavily depends on initial conditions (we term such sets "irregular"). In this case, a single set of parameters for a HOCBF may not work well for all possible initial conditions. Determining the optimal parameters that work for all initial conditions is a non-trivial task. This motivates our second approach, which works for irregular unsafe sets.

Building on [30], where the feasibility robustness metric was introduced, the third direction discussed in the paper is the addition of a sampling learning approach [31], in which irregular unsafe sets can also be considered in unknown environments. We learn feasibility constraints for optimal control problems in unknown environments based on the original models (as opposed to simplified models as in [26]). Also, in our approach, we directly implement the feasibility constraints by CBFs, rendering our overall method provably correct. Specifically, for each type of unsafe set, we sample the state space of the system in the proximity of the safety sets, check for feasibility of the QP one step forward for regular unsafe sets and multiple steps forward for irregular unsafe sets, and learn a differentiable classifier (for feasible and infeasible states) that is then added to the set of initial constraints. We show that, if the initial QP is feasible, then any control that satisfies the corresponding CBF constraints renders all the QPs feasible. An additional contribution of the paper is to improve the accuracy of the classifier by introducing a feedback training algorithm that improves the classifier recursively.

Finally, we validate the effectiveness of the proposed learning-based approaches on a robot control problem in an unknown environment that has both regular (e.g., circular) and irregular (e.g., overlapped circular) obstacles, as well as on an application in autonomous driving, in which moving obstacles (such as other vehicles) are usually involved. We also compare the performance of the two proposed learning methods.

The remainder of the paper is organized as follows. In Sec. II, we provide background and preliminary results on HOCBFs and CLFs, and then formulate a constrained optimal control problem in Sec. III. We introduce the parameter learning method in Sec. IV, followed by the sampling learning method in Sec. V. Case studies for a robot control problem and autonomous driving are presented in Sec. VI. We conclude with final remarks and directions for future work in Sec. VII.

II. PRELIMINARIES

We introduce HOCBFs in this section starting with some formal definitions.

Definition 1. *(Class $\mathcal{K}$ function [33]) A continuous function $\alpha : [0, a) \to [0, \infty), a > 0$ is said to belong to class $\mathcal{K}$ if it is strictly increasing and $\alpha(0) = 0$. A continuous function $\beta : \mathbb{R} \to \mathbb{R}$ is said to belong to extended class $\mathcal{K}$ if it is strictly increasing and $\beta(0) = 0$.*

In this paper, we consider an affine control system:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u} \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times q}$ are locally Lipschitz, and $\boldsymbol{u} \in U \subset \mathbb{R}^q$ with the control constraint set U defined as

$$U := \{\boldsymbol{u} \in \mathbb{R}^q : \boldsymbol{u}_{min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{max}\}. \tag{2}$$

with $\boldsymbol{u}_{min}, \boldsymbol{u}_{max} \in \mathbb{R}^q$ (interpreted component-wise).

Definition 2. *(Forward invariance [1]) A set $C \subset \mathbb{R}^n$ is forward invariant for system (1) if its solutions for some $\boldsymbol{u} \in U$ starting at any $\boldsymbol{x}(0) \in C$ satisfy $\boldsymbol{x}(t) \in C$, $\forall t \geq 0$.*

Definition 3. *(Relative degree [33]) The relative degree of a (sufficiently many times) differentiable function $b : \mathbb{R}^n \to \mathbb{R}$ with respect to system (1) is the number of times it needs to be differentiated along its dynamics until any control $\boldsymbol{u}$ explicitly shows in the corresponding derivative.*

In this paper, we assume that if there exists $\boldsymbol{x}$ such that the control shows up in the derivative of b, then it shows up for all $\boldsymbol{x}$. We refer to the relative degree of b as the relative degree of the constraint as it is used to define the constraint. For a constraint $b(\boldsymbol{x}) \geq 0$ with relative degree m, $b : \mathbb{R}^n \to \mathbb{R}$, and $\psi_0(\boldsymbol{x}) := b(\boldsymbol{x})$, we define a sequence of CBFs $\psi_i : \mathbb{R}^n \to \mathbb{R}, i \in \{1, \dots, m\}$:

$$\psi_i(\boldsymbol{x}) := \dot{\psi}_{i-1}(\boldsymbol{x}) + \alpha_i(\psi_{i-1}(\boldsymbol{x})), \quad i \in \{1, \dots, m\}, \quad (3)$$

where $\alpha_i(\cdot), i \in \{1, \dots, m\}$ denotes a $(m-i)^{th}$ order differentiable class $\mathcal{K}$ function.

We further define a sequence of sets $C_i, i \in \{1, \dots, m\}$ associated with (3) in the form:

$$C_i := \{\boldsymbol{x} \in \mathbb{R}^n : \psi_{i-1}(\boldsymbol{x}) \geq 0\}, \quad i \in \{1, \dots, m\}. \quad (4)$$

Definition 4. *(High Order Control Barrier Function (HOCBF) [2]) Let $C_1, \dots, C_m$ be defined by (4) and $\psi_1(\boldsymbol{x}), \dots, \psi_m(\boldsymbol{x})$ be defined by (3). A function $b : \mathbb{R}^n \to \mathbb{R}$ is a High Order Control Barrier Function (HOCBF) of relative degree m for system (1) if there exist $(m-i)^{th}$ order differentiable class $\mathcal{K}$ functions $\alpha_i, i \in \{1, \dots, m-1\}$ and a class $\mathcal{K}$ function α_m such that*

$$\sup_{\boldsymbol{u} \in U} [L_f^m b(\boldsymbol{x}) + [L_g L_f^{m-1} b(\boldsymbol{x})]\boldsymbol{u} + O(b(\boldsymbol{x})) + \alpha_m(\psi_{m-1}(\boldsymbol{x}))] \geq 0, \quad (5)$$

for all $\boldsymbol{x} \in C_1 \cap, \dots, \cap C_m$. In (5), L_f (L_g) denotes Lie derivatives along f (g), and $O(b(\boldsymbol{x})) = \sum_{i=1}^{m-1} L_f^i(\alpha_{m-i} \circ \psi_{m-i-1})(\boldsymbol{x})$. Further, $b(\boldsymbol{x})$ is such that $L_g L_f^{m-1} b(\boldsymbol{x}) \neq 0$ on the boundary of the set $C_1 \cap, \dots, \cap C_m$.

In the above, the existence of class $\mathcal{K}$ functions is crucial for a valid HOCBF. One of the contributions of this work is to determine such class $\mathcal{K}$ functions that maximize the HOCBF feasibility robustness (defined later).

The HOCBF is a general form of the relative degree one CBF [1], [10], [34] (setting $m = 1$ reduces the HOCBF to the common CBF form. We can define $\alpha_i(\cdot), i \in \{1, \dots, m\}$ in Def. 4 to be extended class $\mathcal{K}$ functions to ensure robustness of a HOCBF to perturbations [1]. However, this cannot ensure that a constraint is eventually satisfied if it is initially violated.

Theorem 1. *([2]) Given a HOCBF $b(\boldsymbol{x})$ from Def. 4 with the associated sets $C_i, i \in \{1, \dots, m\}$ defined by (4), if $\boldsymbol{x}(0) \in \cap_{i=1}^m C_i$, then any Lipschitz continuous controller $\boldsymbol{u}(t)$ that satisfies the constraint in (5), $\forall t \geq 0$ renders $\cap_{i=1}^m C_i$ forward invariant for system (1).*

Definition 5. *(Control Lyapunov function (CLF) [8]) A continuously differentiable function $V : \mathbb{R}^n \to \mathbb{R}$ is an exponentially stabilizing control Lyapunov function (CLF) for system (1) if there exist constants $c_1 > 0, c_2 > 0, c_3 > 0$ such that for $\forall \boldsymbol{x} \in \mathbb{R}^n$, $c_1||\boldsymbol{x}||^2 \leq V(\boldsymbol{x}) \leq c_2||\boldsymbol{x}||^2$,*

$$\inf_{u \in U} [L_f V(\boldsymbol{x}) + L_g V(\boldsymbol{x})\boldsymbol{u} + c_3 V(\boldsymbol{x})] \leq 0. \quad (6)$$

Many existing works [1], [11], [35] combine CBFs for systems with relative degree one with quadratic costs to form optimization problems. Time is discretized and an optimization problem with constraints given by the CBFs (inequalities of the form (5)) is solved at each time step. The inter-sampling effect is considered in [35]. If convergence to a state is desired, then a CLF constraint of the form (6) is added, as in [1] [35]. Note that these constraints are linear in control since the state value is fixed at the beginning of the interval, therefore, each optimization problem is a quadratic program (QP) if the cost is quadratic in the control. The optimal control obtained by solving each QP is applied at the current time step and held constant for the whole interval. The state is updated using dynamics (1), and the procedure is repeated. Formally, the CBF-based QP is defined as follows. We partition a time interval $[0, t_f]$ into a set of equal time intervals $\{[0, \Delta t), [\Delta t, 2\Delta t), \dots\}$, where $\Delta t > 0$. In each interval $[\omega \Delta t, (\omega + 1)\Delta t)$ $(\omega = 0, 1, 2, \dots)$, we assume the control is constant (i.e., the overall control will be piece-wise constant). Then at $t = \omega \Delta t$, we solve the QP:

$$\begin{aligned} \min_{\boldsymbol{u}(\omega\Delta t), \delta(\omega\Delta t)} \quad & \boldsymbol{u}^T(\omega\Delta t) H \boldsymbol{u}(\omega\Delta t) + p_0 \delta^2(\omega\Delta t) \\ \text{s.t. } & \boldsymbol{u}_{min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{max} \\ & L_f V(\boldsymbol{x}) + L_g V(\boldsymbol{x})\boldsymbol{u} + \epsilon V(\boldsymbol{x}) \leq \delta, \\ & L_f^m b(\boldsymbol{x}) + [L_g L_f^{m-1} b(\boldsymbol{x})]\boldsymbol{u} + O(b(\boldsymbol{x})) + \alpha_m(\psi_{m-1}(\boldsymbol{x})) \geq 0 \end{aligned} \quad (7)$$

In the above equation, H is positive definite, $\delta(t)$ is a relaxation variable on the CLF constraint used to avoid conflict with the HOCBF constraint, and $p_0 > 0$ is a penalty on the relaxation $\delta(t) \in \mathbb{R}$. This method works conditioned on the fact that the QP at every time step is feasible. However, this is not guaranteed, in particular under tight control bounds (or very limited controls). In this paper, we show how the QP feasibility can be recursively improved by using machine learning techniques.

III. Problem Formulation and Approach

Consider an optimal control problem for system (1) with the cost defined as:

$$\int_0^{t_f} \mathcal{C}(||\boldsymbol{u}(t)||)dt + p_0||\boldsymbol{x}(t_f) - \boldsymbol{K}||^2, \quad (8)$$

where $||\cdot||$ denotes the 2-norm of a vector; t_f denotes the final time; and $\mathcal{C}$ is a strictly increasing function of its argument (usually assumed quadratic). $\boldsymbol{K} \in \mathbb{R}^n$ is an equilibrium for system (1) in the absence of control and $p_0 > 0$.

Constraint 1 (Unsafe state sets): Let S denote an index set for unsafe (state) sets. System (1) avoids each unsafe set $j \in S$ if the state of system (1) satisfies:

$$b_j(\boldsymbol{x}(t)) \geq 0, \forall t \in [0, t_f], \forall j \in S, \quad (9)$$

where $b_j : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function (not a CBF or HOCBF yet).

The geometry/shape of $b_j(\boldsymbol{x}(t)), j \in S$ is assumed to be known (otherwise, it can be identified using one of several known learning techniques [17]), while its location is unknown. We may use on-board sensors to detect the location of

the obstacle, and this location information is then incorporated into the above constraint. In other words, $b_j(\boldsymbol{x}(t))$ is a function of the location of unsafe set j, and it is completely known when both the type and location of unsafe set j are determined (geometries and types will be discussed in more detail later). The geometry/shape of the obstacles/unsafe sets is used to select which learning method to use. Throughout the paper, we refer to the aforementioned setting as an *unknown environment*.

Constraint 2 (State and control limitations): Assume we have a set of constraints on control input of system (1) as in (2) and on the state in the form:

$$\boldsymbol{x}_{min} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{max}, \forall t \in [0, t_f] \tag{10}$$

where $\boldsymbol{x}_{min} \in \mathbb{R}^n$ and $\boldsymbol{x}_{max} \in \mathbb{R}^n$ denote the minimum and maximum state vectors respectively, and the inequalities are interpreted componentwise. Note that system state constraints can usually be relaxed (and the relaxation is minimized, as shown in the CLF in (7)), and therefore are distinguished from the safety constraint (9). The control bounds in (2) usually denote the system control capability, and thus are hard constraints.

A control policy for system (1) is $\boldsymbol{feasible}$ if the hard constraints (9) and (2) are satisfied.

In this paper, we consider the following problem:

Problem 1. *Find a* $\boldsymbol{feasible}$ *control policy for system (1) such that cost (8) is minimized, and state constraints (10) are satisfied. Formally,*

$$\begin{aligned} \min_{\boldsymbol{u}:[0,t_f]\to\mathbb{R}^q} \int_0^{t_f} & \mathcal{C}(||\boldsymbol{u}(t)||)dt + p_0||\boldsymbol{x}(t_f) - \boldsymbol{K}||^2 \\ \text{s.t. } & b_j(\boldsymbol{x}(t)) \geq 0, j \in S, \forall t \in [0, t_f] \\ & \boldsymbol{x}_{min} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{max}, \\ & \boldsymbol{u}_{min} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{max}, \forall t \in [0, t_f] \\ & \dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x})\boldsymbol{u}. \end{aligned}$$

Example 1. *One typical example of Problem 1 is autonomous driving, in which case the ego vehicle is required to minimize the energy consumption, while satisfying all the safety, speed, and control constraints.*

Approach: Problem 1 is a general problem definition and is itself assumed to be feasible. In order to solve it in an online fashion, we replace the safety constraints above by HOCBF constraints. This method is conservative as the satisfaction of the CBF constraint is only a sufficient condition for the satisfaction of the original constraint; hence, the resulting solution is sub-optimal but it provides safety guarantees. This solution can be driven to near-optimality by optimally tracking a desired trajectory through trajectory planning methods. In our approach, we use HOCBFs [2] to enforce the safety constraint (9) and a CLF [8] to enforce the terminal cost in (8). By defining a Lyapunov function $V(\boldsymbol{x}) := (\boldsymbol{x} - \boldsymbol{K})^T P(\boldsymbol{x} - \boldsymbol{K})$, with P positive definite (i.e., c_1, c_2 are determined by P) and $c_3 = \epsilon > 0$ in the definition of a CLF, the control $\boldsymbol{u}(t)$ should satisfy:

$$L_f V(\boldsymbol{x}(t)) + L_g V(\boldsymbol{x}(t))\boldsymbol{u}(t) + \epsilon V(\boldsymbol{x}(t)) \leq \delta(t), \tag{11}$$

where $\delta(t)$ is a time-varying relaxation variable that makes the above CLF constraint compliant with the HOCBF constraint (5). We seek to minimize this relaxation, as shown in (7).

In order to define a HOCBF for a constraint $b(\boldsymbol{x}) \geq 0$ with relative degree m, we start with recursively defining a set of functions (3) and sets (4) such that $\boldsymbol{x}(0)$ belongs to the intersection of these sets. Then, we use the HOCBF constraint (5) to replace the state constraint $b(\boldsymbol{x}) = b_j(\boldsymbol{x}) \geq 0$ for each $j \in S$, i.e., we define a HOCBF for each unsafe set j. If the HOCBF constraint is satisified, then $b(\boldsymbol{x}) \geq 0$ is also satisfied [2]. We make the following definition of the activation of a HOCBF:

Definition 6. *A HOCBF is said to be activated at state* $\boldsymbol{x}$ *if there exists a control* $\boldsymbol{u} \in U$ *such that* $L_g L_f^{m-1} b(\boldsymbol{x})\boldsymbol{u} = -L_f^m b(\boldsymbol{x}) - O(b(\boldsymbol{x})) - \alpha_m(\psi_{m-1}(\boldsymbol{x}))$.

The activation of a HOCBF is equivalent to the activation of the HOCBF constraint. An example for the activation of a HOCBF is shown in Fig. 2. The activation of the HOCBF will change the system trajectory in order to make the system safe.

The approach to Problem 1 proposed in [1] is based on partitioning the time interval $[0, t_f]$ into $[t_k, t_{k+1}), k = \{0, 1, 2, \dots\}, t_0 = 0$, as introduced at the end of Sec. II. Since the state is kept constant at its value at t_k, a HOCBF constraint is linear in control, thus, the optimization problem is a QP if the cost is quadratic in the control at t_k. Such a QP can easily become infeasible since (2) may conflict with the HOCBF constraints corresponding to (9). Depending on how the system initial state may affect the feasibility of the CBF-based QPs, we classify unsafe sets into two classes:

Definition 7. *(Regular and irregular unsafe sets) Assume Problem 1 is feasibile. An unsafe set* $C_u := \{\boldsymbol{x} \in \mathbb{R} : b(\boldsymbol{x}) < 0\}$ *considered in the QPs (7) is defined as regular if the feasibility of all the CBF-based QPs (7) does not depend on the initial state* $\boldsymbol{x}(0)$ *of system (1). Otherwise, we say that the set is irregular.*

The regularities of multiple unsafe sets are checked one by one through the QP (7); these can then be combined into a single QP. An irregular unsafe set generally depends on the dynamics (1) and corresponds to irregular shapes, such as unsafe sets with sharp corners, in which case the system requires (locally) large control input from the CBF-based QP to avoid corners if the dynamics are nonholonomic; however, the CBF-based QPs may still be feasible if the system trajectory never approaches a corner. An example of a regular unsafe set is a circular obstacle, and an example of an irregular unsafe set is a rectangle for a robot with nonholonomic dynamics, as shown in Fig. 1.

Moreover, In order to address the infeasibility problem of the CBF-based QPs in an offline way, we first define unsafe sets as being of the same "type" if they have the same geometry, meaning the conditions for problem feasibility are the same, e.g., circular unsafe sets are the same type if they have the same radius but different locations. Let S_t denote the set indexing all the unsafe set types. In an unknown environment, if unsafe set types are known, then we can

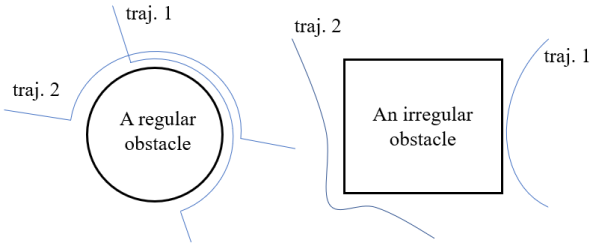


Fig. 1. Regular and irregular obstacle examples for a robot with nonholonomic dynamics. The robot needs the same control input effort (from the CBF-based QPs) in order to avoid the regular circular obstacle, regardless of where the robot is initially located, as shown in example trajectories 1 and 2. However, the robot needs larger control input effort for trajectory 2 than the one for trajectory 1, as there are corners in the irregular rectangular obstacle. Therefore, the feasibility of a CBF-based QP is indeed dependent on the robot initial condition (location).

study the problem feasibility based on these types. Otherwise, we can study the problem feasibility based on some selected set types, and then use these types of unsafe sets to over-approximate other unknown types (a regular circular unsafe set can be used to over-approximiate any shape of unsafe sets). In this paper, we limit ourself to a set of known unsafe set types (a typical application is in autonomous driving where the vehicle types are known).

We revisit two learning-based approaches to ensure the QP feasibility for a certain type of unsafe set: (a) parameter learning for a HOCBF [30]; (b) sampling learning in which HOCBFs are left as originally defined [31]. We begin by informally describing these two approaches, and then we provide a detailed description and analysis for each one in Sections IV and V, respectively.

A. Parameter Learning

In the parameter learning approach [32], we parameterize the definition of a HOCBF (i.e., the class $\mathcal{K}$ functions in Def. 4) for each type of unsafe set, and aim to find the optimal parameters that recursively improve a "feasibility robustness" metric defined as follows.

Feasibility robustness: If the HOCBF constraint corresponding to (9) never becomes active, then safety is guaranteed even without a HOCBF. In this case, the solution space of the QP (7) is clearly at its maximum. Therefore, we only consider the case where the HOCBF constraint will become active. Let $t_j^a \in [0, t_f]$ denote the time instant that the HOCBF constraint (5), corresponding to (9), first becomes active as defined in Def. 6 and (possibly) remains active afterwards if the state of system (1) keeps getting closer to the unsafe set. After the HOCBF constraint in the QP (7) becomes active, the solution space of the QP is limited by the activated constraints, thus, the action space of the system is constrained. As an example, the HOCBF constraint along the red trajectory in Fig. 2 is active earlier than the blue one, therefore, the robot cannot move freely after the CBF constraint becomes active. As the solution approach of the CBF method is point-wise and myopic, we wish to maximize the future action space of the system at each time step. This is equivalent to the feasibility robustness concept we introduce here.

The feasibility robustness of a controller with respect to a constraint (9) can be quantified by the value of $b_j(\boldsymbol{x}(t_j^a))$. The value of $b_j(\boldsymbol{x}(t_j^a))$ may denote a distance metric to a (type of) unsafe set $j \in S_t$ (see the robot control example shown in Fig. 2). In order to maximize the feasibility robustness (equivalently, the future action space, i.e., the future feasible control set a system can select from), we need to minimize the HOCBF value at t_j^a:

$$\min_{t_j^a} b_j(\boldsymbol{x}(t_j^a)), \qquad j \in S_t. \tag{12}$$

The minimization of the above objective is equivalent to minimizing the activation zone of the HOCBF, hence, the feasibility robustness (or future action space) is maximized, as illustrated through the robot control example in Fig. 2.

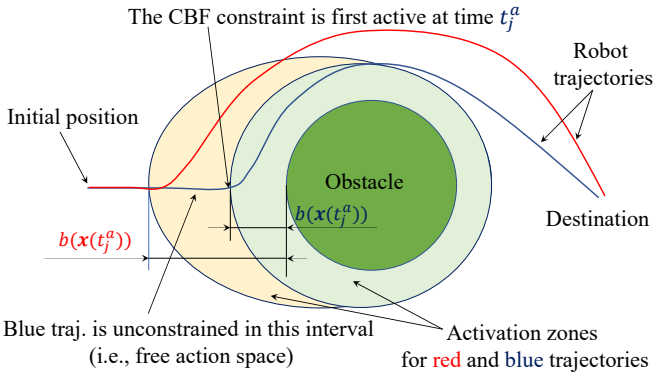


Fig. 2. Illustration of feasibility robustness quantification: CBF activation zones depend on the dimension of the state space as state-feedback is used, but are only visualized in the $x-y$ 2D space. The CBF (HOCBF) constraint (5) first becomes active at time t_j^a when the robot approaches the obstacle. The CBF in the blue trajectory is active later than the one in the red, thus, the robot has more freedom to move before t_j^a and it does not necessarily get close to the obstacle as shown in the figure. When $b_j(\boldsymbol{x}(t_j^a))$ is minimized, the activation zone of the HOCBF is minimized, therefore, the feasibility robustness (future action space) is maximized.

There are three main advantages in maximizing the feasibility robustness of the controller: (i) The QPs are more likely to be feasible since fewer constraints will become active when a system gets close to a number of unsafe sets; (ii) In an *unknown* environment, the controller obtained through the QPs is more robust to changes in the environment and the detection of unknown unsafe sets, since the corresponding HOCBF constraints only become active when a system gets close to these unsafe sets. If the corresponding HOCBF constraints become active before the unsafe sets are detected, the system may fail to avoid these unsafe sets. (iii) There is a higher probability to find a better solution (e.g., energy optimal) if the feasibility robustness is maximized since the QPs are less constrained. Note that proximity to an unsafe set can increase the chance of the state entering it in the presence of disturbances. However, the maximization of feasibility robustness does not mean that the system state has to get close to the unsafe sets, and this performance depends on how we design the CLF to enforce the state convergence in (8).

The robustness objective (12) depends on the time t_j^a, where t_j^a is determined once a HOCBF in the above problem is defined. Therefore, we need to consider objective (12) in the definition of a HOCBF. This approach requires a parameterization of a HOCBF and a learning process for these parameters so as to solve (12).

B. Sampling-based Learning

In the sampling-based learning approach, for each type of unsafe set we sample states in the vicinity of the set. For each of the sampled states, we solve the QP (one or more steps forward, and more steps can help to check the feasibility of a receding horizon) and label the state as +1 if the QP is feasible for all steps, and label it as -1 otherwise. In contrast to parameter learning, sampling-based learning is more myopic as the parameter learning approach considers the horizon from the initial time to the final time. Then, we use a machine learning algorithm to classify all the feasible and infeasible states, and get a feasibility constraint from the classifier hypersurface. This feasibility constraint is then enforced by a HOCBF and added to the QP. This learning approach allows us to deal with irregular unsafe sets in an unknown environment.

Example revisited. Consider the autonomous driving example in Example 1, each type of traffic participants (cars, trucks, pedestrians, bicyclist, etc.) can be treated as a type of unsafe set. We sample the ego vehicle state around each type of unsafe sets, and then learn the corresponding feasibility constraints.

IV. Parameter Learning Approach

The learning objective is to find the optimal class $\mathcal{K}$ function functions of a HOCBF that maximizes the feasibility robustness of the controller with respect to unknown unsafe sets as the QP-based approach for constrained optimal control problems is myopic due to the fact that the QP is solved one single time step forward. The eventual learned parameters can guarantee the feasibility of the CBF-based QP, and improves the system performance, such as optimality. We decompose Problem 1 into two sub-problems: (i) minimize the objective (8) subject to (9), (10), (2) and (1) that is solved with the QP-based method from [1]; (ii) minimize the objective (12) after solving sub-problem (i). We begin with sub-problem (i).

A. Online HOCBF and CLF-based QP (sub-prob. (i))

The approach to sub-problem (i) is based on partitioning the time interval $[0, t_f]$ into a set of equal time intervals $\{[0, \Delta t), [\Delta t, 2\Delta t), \dots\}$, where $\Delta t > 0$. In each interval $[\omega\Delta t, (\omega+1)\Delta t)$ $(\omega = 0, 1, 2, \dots)$, we assume the control is constant (i.e., the overall control will be piece-wise constant). Then, we solve

$$\begin{aligned}\min_{\boldsymbol{u}(\omega\Delta t),\delta(\omega\Delta t)} \; & \mathcal{C}(||\boldsymbol{u}(\omega\Delta t)||) + p_0\delta^2(\omega\Delta t) \\ \text{s.t. } & \boldsymbol{u}_{min} \le \boldsymbol{u} \le \boldsymbol{u}_{max}, \\ & L_fV(\boldsymbol{x}) + L_gV(\boldsymbol{x})\boldsymbol{u} + \epsilon V(\boldsymbol{x}) \le \delta, \\ & L_f^m b_j(\boldsymbol{x}) + [L_gL_f^{m-1}b_j(\boldsymbol{x})]\boldsymbol{u} + O(b_j(\boldsymbol{x})) \\ & \quad + \alpha_m(\psi_{m-1}(\boldsymbol{x})) \ge 0, \forall j \in S.\end{aligned} \tag{13}$$

The above optimization problem can easily become infeasible. In the following sections, we show how we can use machine learning techniques to address this and maximize the feasibility robustness (12).

B. The Parameterization Method

To recursively improve the feasibility of the QP (13), we parameterize the family of class $\mathcal{K}$ functions $\alpha_1(\cdot), \alpha_2(\cdot), \dots, \alpha_m(\cdot)$ as in [32], where m denotes the relative degree of the constraint $b(\boldsymbol{x}) \ge 0$ in the definition of a HOCBF $b(\boldsymbol{x})$. Let $\psi_0(\boldsymbol{x}) := b(\boldsymbol{x})$. Since power functions are the most frequently used class $\mathcal{K}$ functions, we select the $\alpha_i(\cdot)$ functions in (3) as follows:

$$\psi_i(\boldsymbol{x}) := \dot{\psi}_{i-1}(\boldsymbol{x}) + p_i\psi_{i-1}^{q_i}(\boldsymbol{x}), \quad i \in \{1, \dots, m\} \tag{14}$$

where $p_i > 0, i \in \{1, \dots, m\}$ and $q_i \ge 1, i \in \{1, \dots, m\}$. Then, we can obtain the HOCBF constraint (5) when combining with dynamics (1), as shown in Def. 4.

Recall that S_t denotes the set that indexes all the unsafe set types. For each type of unsafe set $j \in S_t$, we consider an arbitrary location for it and get an unsafe set constraint $b_j(\boldsymbol{x}(t)) \ge 0$, similar to (9). Let $\boldsymbol{p} := (p_1, \dots, p_m)$, $\boldsymbol{q} := (q_1, \dots, q_m)$. We know from [32] that the values of $\boldsymbol{p}, \boldsymbol{q}$ affect the feasible set for the decision variables of (13), as well as what time t_j^a the HOCBF constraint (5) will be active, i.e., we can rewrite $b_j(\boldsymbol{x}(t_j^a))$ as $b_j(\boldsymbol{x}(t_j^a))_{\boldsymbol{p},\boldsymbol{q}}$. Since t_j^a depends on $\boldsymbol{p}, \boldsymbol{q}$ and $b_j(\cdot)$ no longer explicitly depends on $\boldsymbol{x}(t_j^a)$ (i.e., $b_j(\boldsymbol{x}(t_j^a))_{\boldsymbol{p},\boldsymbol{q}}$ is fixed once $\boldsymbol{p}, \boldsymbol{q}$ are given), we define $\mathcal{D}_j(\boldsymbol{p}, \boldsymbol{q}) := b_j(\boldsymbol{x}(t_j^a))_{\boldsymbol{p},\boldsymbol{q}}$, and reformulate (12) so that the minimization is over $\boldsymbol{p}, \boldsymbol{q}$:

$$\min_{\boldsymbol{p},\boldsymbol{q}} \mathcal{D}_j(\boldsymbol{p}, \boldsymbol{q}), \quad j \in S_t. \tag{15}$$

We can, therefore, view the minimization of $D_j(\boldsymbol{p}, \boldsymbol{q})$ as the maximization of the feasibility robustness that depends on $\boldsymbol{p}, \boldsymbol{q}$. However, this optimization problem is hard to solve as t_j^a corresponds to a particular trajectory resulting from a sequence of QPs from the initial time to the final time. We will introduce a solution approach using machine learning techniques in the following section.

C. Offline Feasibility-Guided Optimization (sub-prob. (ii))

Note that subproblem (ii) depends on subproblem (i), and the feasibility of subproblem (i) depends on control bounds (2). Given an arbitrary $\boldsymbol{x}(0)$, one can generally expect most of the $\boldsymbol{p}, \boldsymbol{q}$ values to result into infeasible solutions of problem (13), which makes (15) difficult to solve. Therefore, we need to first solve the infeasiblity problem of sub-problem (i).

We randomly sample $\boldsymbol{p}, \boldsymbol{q}$ values over their domain (positive), and for each set of $\boldsymbol{p}, \boldsymbol{q}$ values, we solve problem (13) until the terminal state constraint is satisfied within some allowed error. If problem (13) is feasible at all times, then we label this particular set of $\boldsymbol{p}, \boldsymbol{q}$ values as $+1$, otherwise, we label it as -1. Eventually, we get sets of feasible and infeasible $\boldsymbol{p}, \boldsymbol{q}$ points. Note that the penalty method [32] guarantees that $+1$ data points exist given the control bounds (2) if certain conditions are satisfied. We assume that the control bounds

(2) are properly defined such that we can select balanced data sets for better classification, e.g., the same number of feasible and infeasible samplings with large enough data size, from the randomly sampled data. Then we can apply a classification method, e.g., a support vector machine (SVM) [36], to classify these two balanced sets and get a continuously differentiable hypersurface

$$\mathfrak{H}_j : \mathbb{R}^{2m} \to \mathbb{R}, \tag{16}$$

where

$$\mathfrak{H}_j(\boldsymbol{p}, \boldsymbol{q}) \geq 0 \tag{17}$$

denotes the set of $\boldsymbol{p}, \boldsymbol{q}$ values which leads to the feasible solution of QPs (13), i.e., it defines the feasibility constraint for the set of $\boldsymbol{p}, \boldsymbol{q}$ values associated with the QPs (13). With the assistance of the feasiblity classification hypersurface, we look further to optimize (15), i.e., we consider (15) subject to (17).

We wish to get the set of $\boldsymbol{p}, \boldsymbol{q}$ values such that (17) is satisfied, but (17) is usually complex as it is directly obtained from the classifier hypersurface, therefore, we view $\mathfrak{H}_j(\boldsymbol{p}, \boldsymbol{q}) \geq 0$ as a constraint applied to a dynamic system with states $\boldsymbol{p}, \boldsymbol{q}$ and formulate an associate HOCBF. In particular, just like $b(\boldsymbol{x})$ is associated with the dynamic system (1), we need to introduce an *auxiliary dynamic system* for $\mathfrak{H}_j(\boldsymbol{p}, \boldsymbol{q})$ and take $\boldsymbol{p}, \boldsymbol{q}$ as state variables, as shown in the sequel.

We have imposed the assumption that the control bounds (2) are properly defined such that the whole problem is well-posed to get a proper constraint (17) from the hypersurface. However, the learned hypersurface is still complex even with the HOCBF method, making this optimization problem (15) subject to (17) hard to solve. We use the following dynamic process to simplify this optimization problem.

We start at some feasible $\boldsymbol{p}_0 \in \mathbb{R}^m, \boldsymbol{q}_0 \in \mathbb{R}^m$ to search for the optimal $\boldsymbol{p}, \boldsymbol{q}$ values. Since the determination of the optimal $\boldsymbol{p}, \boldsymbol{q}$ is a dynamic process, we define the gradient (auxiliary dynamics) for $\boldsymbol{p}, \boldsymbol{q}$ as the variations of $\boldsymbol{p}, \boldsymbol{q}$ that are controlled, i.e., we have

$$\begin{aligned} (\dot{\boldsymbol{p}}(\mathfrak{t}), \dot{\boldsymbol{q}}(\mathfrak{t})) &= \boldsymbol{\nu}(\mathfrak{t}), \\ \text{s.t. } \mathfrak{H}_j(\boldsymbol{p}, \boldsymbol{q}) &\geq 0, \\ \boldsymbol{p}(\mathfrak{t}_0) = \boldsymbol{p}_0, \boldsymbol{q}(\mathfrak{t}_0) &= \boldsymbol{q}_0, \end{aligned} \tag{18}$$

where $\boldsymbol{\nu} \in \mathbb{R}^{2m}$ denotes an input vector in the dynamic process constructed in order to determine the optimal $\boldsymbol{p}, \boldsymbol{q}$. $\mathfrak{t}$ denotes the dynamic process time for the optimization of (15), which is different and independent from t in (1) and problem (13). $\mathfrak{t}_0 \in \mathbb{R}$ denotes the initial time.

We want to get a HOCBF corresponding to (17) treated as a state constraint analogous to $b_j(\boldsymbol{x}) \geq 0$ in (9) which leads to (5). Considering the feasibility of problem (13), the dynamic process that is controlled by $\boldsymbol{\nu}$ should be subjected to (18), as well as subjected to the HOCBF constraint for (17) since we define the hypersurface in (17) to be a HOCBF, as discussed in the last three paragraphs. Since we take all the state variables of the auxiliary dynamics (18) as the input for the classifier, the relative degree of the feasibility constraint (17) with respect to (18) is 1, i.e., we only need to differentiate $\mathfrak{H}_j(\boldsymbol{p}, \boldsymbol{q})$ along the dynamics (18) once to let $\boldsymbol{\nu}$ show up. A control $\boldsymbol{\nu}$ should satisfy the HOCBF constraint (5) which, given the system dynamics in (18), in this case is:

$$\frac{d\mathfrak{H}_j(\boldsymbol{p}, \boldsymbol{q})}{d(\boldsymbol{p}, \boldsymbol{q})}\boldsymbol{\nu} + \beta_1(\mathfrak{H}_j(\boldsymbol{p}, \boldsymbol{q})) \geq 0, \tag{19}$$

where $\beta_1(\cdot)$ is an extended class $\mathcal{K}$ function as $\mathfrak{H}_j(\boldsymbol{p}, \boldsymbol{q})$ could be negative due to the classification error. Any control $\boldsymbol{\nu}$ that satisfies (19) implies that the resulting $\boldsymbol{p}, \boldsymbol{q}$ (determined by $\boldsymbol{\nu}$) leads to a feasible solution of QPs (13) in the dynamic process.

We implement the feasibility constraint (17) by the HOCBF constraint (19) in which the control $\boldsymbol{\nu}$ explicitly shows. However, the cost function (15) is only defined over the state of the auxiliary dynamics (18), and we also wish the control $\boldsymbol{\nu}$ to show up in the cost function, which is required by the CBF-based optimization, as shown in Sec. IV-A. Therefore, we consider the derivative of the cost function (15) as our new cost to let $\boldsymbol{\nu}$ show up in the cost function. Recall that (15) is the robustness metric that we wish to minimize. As long as the derivative of (15) is negative, we can ensure that (15) is decreasing at each time step by discretizing $\mathfrak{t}$ similar to sub-problem (i). To modify (15), we proceed as follows.

By taking the derivative of (15) with respect to $\mathfrak{t}$, we have

$$\frac{d\mathcal{D}_j(\boldsymbol{p}(\mathfrak{t}), \boldsymbol{q}(\mathfrak{t}))}{d\mathfrak{t}} = \frac{d\mathcal{D}_j(\boldsymbol{p}(\mathfrak{t}), \boldsymbol{q}(\mathfrak{t}))}{d(\boldsymbol{p}(\mathfrak{t}), \boldsymbol{q}(\mathfrak{t}))}\boldsymbol{\nu}. \tag{20}$$

Then, we reformulate sub-problem (ii) through the dynamic process (18). The result is the **Feasibility-Guided Optimization** (FGO) algorithm that is implemented by the same approach as introduced in Sec. IV-A, i.e., we discretize $\mathfrak{t}$, and at each $\mathfrak{t} = \omega\Delta\mathfrak{t}, \omega \in \{0, 1, \dots\}$, where $\Delta\mathfrak{t} > 0$ denotes the discretization constant, and we solve

$$\begin{aligned} \min_{\boldsymbol{\nu}(\mathfrak{t})} & \frac{d\mathcal{D}_j(\boldsymbol{p}(\mathfrak{t}), \boldsymbol{q}(\mathfrak{t}))}{d(\boldsymbol{p}(\mathfrak{t}), \boldsymbol{q}(\mathfrak{t}))}\boldsymbol{\nu}(\mathfrak{t}), \\ \text{s.t. } & \frac{d\mathfrak{H}_j(\boldsymbol{p}, \boldsymbol{q})}{d(\boldsymbol{p}, \boldsymbol{q})}\boldsymbol{\nu} + \beta_1(\mathfrak{H}_j(\boldsymbol{p}, \boldsymbol{q})) \geq 0, \\ & \boldsymbol{\nu}_{min} \leq \boldsymbol{\nu} \leq \boldsymbol{\nu}_{max}. \end{aligned} \tag{21}$$

where $\boldsymbol{\nu}_{min} < \mathbf{0}, \boldsymbol{\nu}_{max} > \mathbf{0}$ (componentwise), $\mathbf{0} \in \mathbb{R}^{2m}$. Then, we update (18) for $\mathfrak{t} \in (\omega\Delta\mathfrak{t}, (\omega+1)\Delta\mathfrak{t})$ with $\boldsymbol{\nu}^*(\mathfrak{t})$. The optimization problem (21) is a linear program (LP) at each time step for each initial $\boldsymbol{p}, \boldsymbol{q}$ (we need to reset $\mathfrak{t}$ for each set of initial $\boldsymbol{p}, \boldsymbol{q}$ values). Without any constraint on $\boldsymbol{\nu}$, the LP (21) is ill-posed because it leads to unbounded solutions. In fact, the value of $\boldsymbol{\nu}$ determines the search step length of the FGO algorithm implemented through the LP (21), and we want to limit this step length. Therefore, we add limitations to $\boldsymbol{\nu}$ for the LP (21). Note that in the last equation, $\frac{d\mathcal{D}_j(\boldsymbol{p}(\mathfrak{t}), \boldsymbol{q}(\mathfrak{t}))}{d(\boldsymbol{p}(\mathfrak{t}), \boldsymbol{q}(\mathfrak{t}))}$ is a row vector of dimension $2m$, while $\boldsymbol{\nu}$ is a column vector of dimension $2m$. Therefore, the cost function in the last equation is a scalar function of $\boldsymbol{\nu}$.

After adding limitations to $\boldsymbol{\nu}$ in (21), the dynamic process's search step length will become bounded. Although there are control limitations on $\boldsymbol{\nu}$, the resulting LP from the optimization (21) is always feasible as the relative degree of (17) with respect to (18) is 1 [2]. We also need to evaluate $\frac{\partial\mathcal{D}_j}{\partial p_1}, \dots, \frac{\partial\mathcal{D}_j}{\partial p_m}, \frac{\partial\mathcal{D}_j}{\partial q_1}, \dots, \frac{\partial\mathcal{D}_j}{\partial q_m}$ at each time step, i.e., evaluate the coefficients of the cost function (21).

The resulting process is the FGO algorithm formulated from (21) to optimize $\boldsymbol{p}, \boldsymbol{q}$. For each step of the FGO algorithm, any one of the following four conditions may terminate it: (a) the problem (13) becomes infeasible (since the hypersurface from SVM cannot ensure 100% classification accuracy), (b) the evaluated values of $\frac{\partial \mathcal{D}_j}{\partial p_1}, \ldots, \frac{\partial \mathcal{D}_j}{\partial p_m}, \frac{\partial \mathcal{D}_j}{\partial q_1}, \ldots, \frac{\partial \mathcal{D}_j}{\partial q_m}$ are all 0, (c) the objective function value of (15) is greater than the current known minimum value. (d) the iteration time exceeds some $N \in \mathbb{N}$. We present the FGO algorithm in Algo. 1. This approach is more computationally efficient than an approach that directly incorporates the barrier function in the cost, as Algo. 1 only involves QPs.

If we consider (21) without constraint (19), then we have the commonly used gradient descent (GD) algorithm. The FGO algorithm is more efficient compared with GD since the solution searching path is guided by the feasibility of (13). However, the hypersurface in (17) cannot guarantee the correctness of the FGO method due to the classification error. We can apply GD one step forward whenever the FGO algorithm terminates to alleviate this limitation.

Note that we can update the training set and get a new classifier in (17) after running the FGO algorithm for a number of different initial samples $\boldsymbol{p}_0, \boldsymbol{q}_0$, i.e., re-initialize (18) for each FGO process. We will show how this may affect the performance of FGO in the case studies considered in Sec. VI. Once we have learned feasibility and robustness for some known types of unsafe sets with the FGO algorithm, we can use these unsafe sets to approximate other types of unsafe sets.

Remark 1. *The time complexity of subproblem (i), i.e., the QP (13), is $O(d^3)$, where $d = q + 1$ is the dimension of decision variables. Since the CBF method (after pre-training) does not need planning, it is more computationally efficient than path planning methods, such as Rapidly-exploring Randomized Trees (RRT) [37] and A* [38], as seen in Sec. VI.*

Remark 2. *The time complexity of subproblem (ii) is that of a LP [39], i.e., $O((d + c)^{1.5} dL)$, where d, c are the number of decision variables and constraints, respectively, and L is a given parameter. Thus, the complexity of the FGO is almost the same as the GD one as it just has one more constraint than the GD method, so the computational times are comparable.*

It is important to note that the parameter learning approach cannot work for irregular unsafe sets since the problem feasibility heavily depends on the initial condition, and system (1) may get stuck at the local traps formed by the irregular unsafe sets. Determining the optimal parameters that work for all initial conditions is non-trivial, and maybe infeasible. We show how we may consider irregular unsafe sets in the next section.

Algorithm 1: FGO algorithm

Input: Constraints (9), $\boldsymbol{K}$ in (8), system (1) with (2), N

Output: $\boldsymbol{p}^*, \boldsymbol{q}^*, \mathcal{D}_{min}$

1. Sample $\boldsymbol{p}, \boldsymbol{q}$ in the definition of the HOCBF;
2. Discard samples that do not meet the initial conditions of HOCBF constraint (5);
3. Solve (13) for each sample for $t \in [0, t_f]$ and label all samples;
4. Select balanced training and testing data sets;
5. Use machine learning techniques to find classifier (16);
6. Pick a feasible $\boldsymbol{p}_0, \boldsymbol{q}_0$, $\mathcal{D}_{min} = \mathcal{D}_j(\boldsymbol{p}_0, \boldsymbol{q}_0)$, iter. $= 1$;

while *iter.*++ $\leq N$ **do**
- Evaluate $\frac{\partial \mathcal{D}_j}{\partial p_1}, \ldots, \frac{\partial \mathcal{D}_j}{\partial p_m}, \frac{\partial \mathcal{D}_j}{\partial q_1}, \ldots, \frac{\partial \mathcal{D}_j}{\partial q_m}$ at $\boldsymbol{p}_0, \boldsymbol{q}_0$ with random perturbation to p_k or $q_k, k \in \{1, 2, \ldots, m\}$;
- **if** $\frac{\partial \mathcal{D}_j}{\partial p_k}, \frac{\partial \mathcal{D}_j}{\partial q_k}, \forall k \in \{1, 2, \ldots, m\}$ *is infeasible to evaluate over sub-problem* (i) **then**
  - Jump to the very begining of the loop;
- **else**
  - $\frac{\partial \mathcal{D}_j}{\partial p_k} = 0, \frac{\partial \mathcal{D}_j}{\partial q_k} = 0, \exists k \in \{1, 2, \ldots, m\}$ is infeasible to evaluate over sub-problem (i);
- **end**
- Solve the optimization (21) and get new $\boldsymbol{p}, \boldsymbol{q}$;
- Solve the problem (13) with $\boldsymbol{p}, \boldsymbol{q}$;
- **if** *(13) is feasible for all* $t \in [0, t_f]$ **then**
  - **if** $\mathcal{D}_{min} \geq \mathcal{D}_j(\boldsymbol{p}, \boldsymbol{q})$ **then**
    - $\mathcal{D}_{min} = \mathcal{D}_j(\boldsymbol{p}, \boldsymbol{q})$, $\boldsymbol{p}_0 = \boldsymbol{p}, \boldsymbol{q}_0 = \boldsymbol{q}$;
  - **else**
    - break;
  - **end**
- **else**
  - Solve the optimization (21) without (19) and get new $\boldsymbol{p}, \boldsymbol{q}$;
  - Solve the problem (13) with $\boldsymbol{p}, \boldsymbol{q}$;
  - **if** $\mathcal{D}_{min} \geq \mathcal{D}_j(\boldsymbol{p}, \boldsymbol{q})$ **then**
    - $\mathcal{D}_{min} = \mathcal{D}_j(\boldsymbol{p}, \boldsymbol{q})$, $\boldsymbol{p}_0 = \boldsymbol{p}, \boldsymbol{q}_0 = \boldsymbol{q}$;
  - **else**
    - break;
  - **end**
- **end**

end

$\boldsymbol{p}^* = \boldsymbol{p}_0, \boldsymbol{q}^* = \boldsymbol{q}_0$;

V. Sampling Learning Approach

In this section, we show how we can deal with irregular unsafe sets as defined in Def. 7. This approach also works for regular unsafe sets, but tends to be conservative. The dimension of the data is the same as the one of the system state, the training is done offline, and thus it works for rapidly dynamic systems as well. Recall that the type of every unsafe set $i \in S$ is already known in an unknown environment, and S_t denotes an index set for unsafe set types in an unknown environment, and $S_j \subseteq S, j \in S_t$ denotes the index set for unsafe sets of type j.

A. Feasible and Infeasible State Sets

The QP (13) may be infeasible at a given state $\boldsymbol{x}(t)$ at time t. The constraints in (9) form a constraint set for the state of system (1). Without control (i.e., $\boldsymbol{u}(t) = 0, \forall t \in [0, t_f]$), system (1) may escape from this constraint set for a given

initial state $\boldsymbol{x}(0)$. However, if the system is controlled with the optimal control $\boldsymbol{u}^*(t)$ from solving the QP (13), the system may also exit this constraint set since the limited control may not be able to prevent the system from leaving this set when the state approaches the set boundary, which typically happens in high relative degree systems. Then, QP (13) becomes infeasible.

An intuitive example is a robot control problem. Suppose a robot, with limited control input (deceleration), needs to arrive at a destination while avoiding an obstacle that is located between the robot's initial position and the destination. When the robot gets close to the obstacle with high speed, it may not be able to brake in time to avoid the obstacle since the control in QP (13) is limited by (2). However, when the speed is low, the robot can safely avoid the obstacle.

The main idea of the sampling learning approach is to partition the state space of system (1) into sets in which the QP (13) is feasible or infeasible after a certain number of time steps. This is a difficult problem, especially for high-dimensional systems with fast dynamics. We show how machine learning techniques can be used to address this problem.

B. Sampling and Classification

As mentioned before, the system is in an unknown environment such that it only knows the types of the unsafe sets the environment may include, but not their number and locations. In order to make the learned feasibility constraint independent from the location of an unsafe set, we choose the relative coordinate $\boldsymbol{z} \in \mathbb{R}^n$ between the system and unsafe set as one of the input features for machine learning techniques. For example, let the system state be $\boldsymbol{x} := (x_1, x_2, \ldots, x_n)$. If x_1, x_2 denote the 2-D position of an object in $\boldsymbol{x}$, then we define input features $\boldsymbol{z} := (x_1 - x_o, x_2 - y_o, x_3, \ldots, x_n)$ for the machine learning techniques, where $(x_o, y_o) \in \mathbb{R}^2$ denotes the 2-D location of the unsafe set. Along the same lines, we may also consider the relative speed and acceleration between the system and unsafe set as the input for the machine learning model in order to consider moving unsafe sets.

For each type of unsafe set $j \in S_t$, since we only consider the relative coordinates as the input for the learning model as discussed above, we arbitrarily assign a location and an orientation (if it exists) for j and randomly sample around j to find an initial state $\boldsymbol{z}(0)$ around the unsafe set. We then solve the QP (13) at time 0 according to the geometry of the unsafe set:

- (i) **Regular unsafe set:** we solve the QP (13) at time 0 for 1 time step forward.
- (ii) **Irregular unsafe set:** we solve the QP (13) at time 0 for $H_t \in \mathbb{N} > 1$ time steps forward.

Remark 3. *Unlike regular unsafe sets, when dealing with irregular unsafe sets, the system may get stuck at local traps, e.g., in Fig. 10. This is why we extend the solution of the QP (13) to $H_t > 1$ time steps. In this case, any one of the H_t-step QPs becoming infeasible will make the system fail. The local traps can easily make the QP (13) infeasible, especially when the system gets to their boundary. Therefore, it is more likely to make an inital state that is located around the local traps belong to the infeasible set when we solve the QP (13) $H_t > 1$ time steps forward. Then, the system may avoid the local traps if it avoids the infeasible set, and thus improve its reachability.*

If the QP (or all the QPs in case (ii)) (13) is feasible, we label the state $\boldsymbol{z}(0)$ as $+1$. Otherwise, it is labelled as -1. This procedure results in two labelled classes. We employ a machine learning technique (such as Support Vector Machine (SVM), Deep Neural Network (DNN), etc.) with $\boldsymbol{z}(0)$ as input to perform classification, and get a classification hypersurface for each $j \in S_t$ in the form:

$$H_j(\boldsymbol{z}) : \mathbb{R}^n \to \mathbb{R}, \tag{22}$$

where $H_j(\boldsymbol{z}(0)) \geq 0$ denotes that $\boldsymbol{z}(0)$ belongs to the feasible set. This inequality is called the **feasibility constraint**.

Assuming the relative degree of (22) is m, we define the set of all control values that satisfy $H_j(\boldsymbol{z}(t)) \geq 0$ as:

$$\begin{aligned} K^j_{fea} = \{\boldsymbol{u} \in U : L_f^m H_j(\boldsymbol{z}) + L_g L_f^{m-1} H_j(\boldsymbol{z})\boldsymbol{u} \\ + O(H_j(\boldsymbol{z})) + \alpha_m(\psi_{m-1}(\boldsymbol{z})) \geq 0\} \end{aligned} \tag{23}$$

where ψ_{m-1} is recursively defined as in (3) by H_j with extended class $\mathcal{K}$ functions.

We define feasibility forward invariance as follows:

Definition 8. *An optimal control problem is feasibility forward invariant for system (1) if its solutions starting at all feasible $\boldsymbol{x}(0)$ are feasible for all $t \geq 0$.*

Theorem 2. *([31]) Assume that the hypersurfaces $H_j(\boldsymbol{z}), \forall j \in S_t$ ensure 100% feasibility and infeasibility classification accuracy. If $H_j(\boldsymbol{z}(0)) \geq 0, \forall j \in S_t$, then any Lipschitz continuous controller $\boldsymbol{u}(t) \in K^j_{fea}, \forall j \in S_t$ renders Problem 1 feasibility forward invariant.*

Naturally, machine learning techniques cannot ensure 100% classfication accuracy. We introduce an approach based on feedback training to improve the classification accuracy in the following subsection. In fact, if the classification accuracy is high enough, Problem 1 may also be always feasible since system (1) may never reach the infeasible space.

Similarly to the QP (13), we have a feasible reformulated problem at $t = \omega\Delta t$ ($\omega = 0, 1, 2, \ldots, \frac{t_f}{\Delta t} - 1$):

$$\begin{aligned} \min_{\boldsymbol{u}(t), \delta(t)} \; & \mathcal{C}(||\boldsymbol{u}(t)||) + p_0\delta^2(t) \\ \text{s.t. } & \boldsymbol{u}_{min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{max} \\ & L_f V(\boldsymbol{x}) + L_g V(\boldsymbol{x})\boldsymbol{u} + \epsilon V(\boldsymbol{x}) \leq \delta, \\ & L_f^m b(\boldsymbol{x}) + [L_g L_f^{m-1} b(\boldsymbol{x})]\boldsymbol{u} + O(b(\boldsymbol{x})) + \alpha_m(\psi_{m-1}(\boldsymbol{x})) \geq 0, \\ & L_f^m H_j(\boldsymbol{z}) + L_g L_f^{m-1} H_j(\boldsymbol{z})\boldsymbol{u} + O(H_j(\boldsymbol{z})) + \alpha_m(\psi_{m-1}(\boldsymbol{z})) \geq 0 \end{aligned} \tag{24}$$

where $b(\boldsymbol{x}) = b_j(\boldsymbol{x})$, and every safety constraint of the same type j uses the same $H_j(\boldsymbol{z}(t)) \geq 0, \forall j \in S_t$.

C. Feedback Training

For each $j \in S_t$, we first sample the points without any hypersurface (22). After the first iteration, we obtain a hypersurface that classifies the state space of system (1) into feasible and infeasible sets, but with relatively low accuracy. Then we can add these hypersurfaces (22) into the QP (13)

(i.e., use (24)) and sample new data points to perform a new classification, and obtain another classification hypersurface that replaces the old one. Iteratively, the classification accuracy is improved and the infeasible set shrinks.

Since the CBF method requires the constraint to be initially satisfied, we discard the samples that do not meet this requirement. To ensure classification accuracy, we also need unbiased data samples.

D. Generalization

Since we sample data around the unsafe set, we also need to check the generalization of the hypersurface (22) in the area where we do not sample since system (1) may actually start from some state in the unsampled area. Problem 1 is usually feasible when system (1) is far away from the unsafe set, therefore, the unsampled area should be located at the positive side of the hypersurface (22), which can be viewed as the generalization (i.e., not overfitting) of this hypersurface, as usually appears in machine learning techniques.

Once we get a hypersurface (feasibility constraint) for a type of unsafe set in the pre-training process, we can also apply this feasibility constraint to other unsafe sets that are of the same type but with different locations since the hypersurface only depends on the relative location of the pre-training unsafe set. This is helpful for systems in which we do not know the number and locations of the unsafe set, but know the type of unsafe sets the environment has.

It is important to note that the optimal hypersurface is not unique given the training samples; this is due to the weight space symmetries [36] in neural networks.

Comparison between parameter and sampling learning: The parameter learning approach tries to learn the optimal parameters in the definition of a HOCBF such that the QP feasibility robustness is maximized. This can improve the adaptivity of a system in an unknown environment. However, a single set of parameters may not work for all possible initial conditions, and the system may also get stuck at local traps, so that the QP may still be infeasible. However, the sampling learning approach can deal with irregular unsafe sets as it learns a feasibility constraint by checking the feasibility of a longer (than 1) receding horizon control (multi-step QP) for each sampling state. The main drawback of this approach may be the conservativeness of the learned feasibility constraint as the feasibility robustness is not considered, which could lead to unreasonable system behaviors. We need to properly choose the learning model to alleviate the conservativeness.

VI. Implementation and Case Studies

We implemented the FGO algorithm in MATLAB and performed simulations for a robot control problem. The robot control problem is basically a reach and avoid problem, where the robot is required to reach a desired location while avoiding all the obstacles [40]. Suppose all the obstacles are of the same type but the obstacle number and their locations are unknown to the robot, and the robot is equipped with a sensor ($\frac{2}{3}\pi$ field of view (FOV) and $7m$ sensing distance with $1m$ uncertainty) to detect the obstacles.

With $\boldsymbol{x} := (x, y, \theta, v), \boldsymbol{u} = (u_1, u_2)$, the dynamics are defined as:

$$\begin{aligned} \dot{x} &= v\cos(\theta), \quad \dot{y} = v\sin(\theta), \\ \dot{\theta} &= u_1, \qquad\quad \dot{v} = u_2, \end{aligned} \tag{25}$$

where x, y denote the location along x, y axis, respectively, θ denotes the heading angle of the robot, v denotes the linear speed, and u_1, u_2 denote the two control inputs for turning and acceleration, respectively.

We instantiate cost (8) in the form:

$$\min_{\boldsymbol{u}(t)} \int_0^{t_f} \left[u_1^2(t) + u_2^2(t)\right] dt + p_0((x(t_f)-x_d)^2 + (y(t_f)-y_d)^2). \tag{26}$$

In other words, we wish to minimize the energy consumption and drive the robot to a given destination $(x_d, y_d) \in \mathbb{R}^2$, i.e., drive $(x(t), y(t))$ to $(x_d, y_d), \forall t \in [t', t_f]$, for some $t' \in [0, t_f]$, as defined in (8). The robot dynamics are not full state linearizable [33] and the relative degree of the position (output) is 2. Therefore, we cannot directly apply a CLF. However, the robot can arrive at the destination if its heading angle θ stabilizes to the desired direction and its speed v stabilizes to a desired speed $v_0 > 0$, i.e.,

$$\theta(t) \to \arctan(\frac{y_d - y(t)}{x_d - x(t)}), \quad v(t) \to v_0, \forall t \in [0, t_f]. \tag{27}$$

Now, we can apply the CLF method since the relative degrees of the heading angle and speed are 1.

The unsafe sets (9) are defined as circular (regular) obstacles:

$$\sqrt{(x(t) - x_i)^2 + (y(t) - y_i)^2} \geq r, \forall i \in S, \tag{28}$$

where (x_i, y_i) denotes the location of the obstacle $i \in S$, and $r > 0$ denotes the safe distance to the obstacle. Note that we may have irregular obstacles when there are overlapped circular obstacles.

The speed and control constraints (2) are defined as:

$$V_{min} \leq v(t) \leq V_{max}, \tag{29}$$

$$u_{1,min} \leq u_1(t) \leq u_{1,max}, u_{2,min} \leq u_2(t) \leq u_{2,max}, \tag{30}$$

where $V_{min} = 0m/s, V_{max} = 2m/s, u_{1,max} = -u_{1,min} = 0.2rad/s, u_{2,max} = -u_{2,min} = 0.5m/s^2$. Other parameters are $p_0 = 1, \Delta t = 0.1s, \epsilon = 10$.

A. Parameter Learning Case Studies

We set up the FGO algorithm training environment with the initial position of the robot, the location of the obstacle (with radius $6m$ and $r = 7m$) and the destination as $(5m, 25m)$, $(32m, 25m)$ and $(45m, (25+\varepsilon)m)$ where $\varepsilon \in \mathbb{R}$, respectively. The initial heading angle and speed of the robot are 0 deg and V_{max}, respectively. The parameters for the FGO algorithm are $\Delta \mathfrak{t} = 0.1, \boldsymbol{\nu}_{max} = -\boldsymbol{\nu}_{min} = (0.1, 0.1, 0.1, 0.1)$. The map for FGO training is shown in Fig. 4(a).

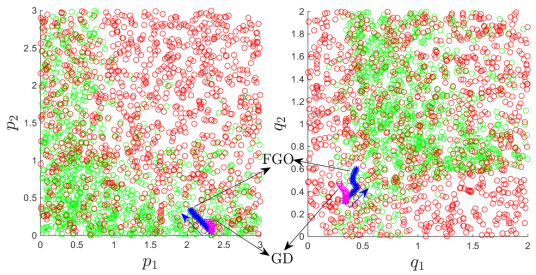
Note that the value of ε will affect the trajectory of the robot since we have a circular obstacle. If $\varepsilon = 0$, the robot will eventually stop at the equilibrium point shown in Fig. 4(a). If $\varepsilon > 0$, the robot goes left around the obstacle as shown in Fig. 4(a). Otherwise, the robot turns right.

We choose a very small $\varepsilon \neq 0$ in our FGO algorithm. Since the obstacle constraint (28) has relative degree 2 with respect to the dynamics, our HOCBF parameters are $\boldsymbol{p} = (p_1, p_2), \boldsymbol{q} = (q_1, q_2)$. We collect balanced data sets (the ratio of the samplings between $+1$ and -1 labelled data is 1:1 for both training and testing sets) from the random samplings of M training and 1000 testing samples for $\boldsymbol{p}$ and $\boldsymbol{q}$ over interval $(0, 3]$ and $(0, 2]$, respectively. Note that we allow $\boldsymbol{q}$ to sample in $(0, 1)$ as the robot will not get too close to the circular obstacles, although the class $\mathcal{K}$ function $p_i \psi_{i-1}^{q_i}(\boldsymbol{x})$ in (14) is not Lipschitz continuous when $\psi_{i-1}(\boldsymbol{x}) = 0$.

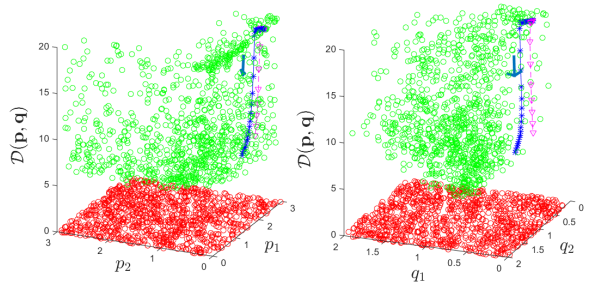
The classification model is a support vector machine (SVM) with polynomial kernel of degree 7, i.e., the kernel function $k(\boldsymbol{y}, \boldsymbol{z})$ is defined as

$$k(\boldsymbol{y}, \boldsymbol{z}) = (c_1 + c_2 \boldsymbol{y}^T \boldsymbol{z})^7, \tag{31}$$

where $\boldsymbol{y}, \boldsymbol{z}$ denote input vectors of SVM (i.e., $\boldsymbol{y} := (\boldsymbol{p}, \boldsymbol{q})$, as well as for $\boldsymbol{z}$). We set $c_1 = 0.8, c_2 = 0.5$, and the comparisons between FGO and GD are shown in Table I ("better/worse than GD percentage" denotes the percentage of data in the testing set that the FGO obtains a better objective value (12) of subproblem (ii) than the GD method).



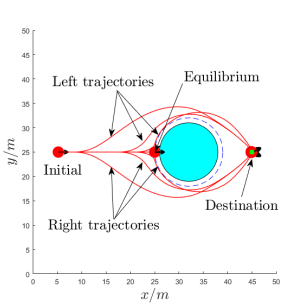
(a) FGO and GD algorithms search paths in 2D.



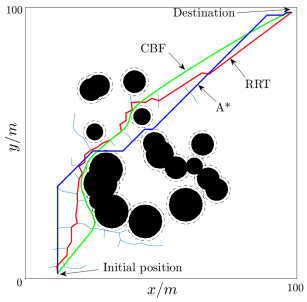
(b) FGO and GD algorithms search paths in 3D.

Fig. 3. FGO and GD comparison. The red and green circles denote infeasible and feasible points for $\boldsymbol{p}, \boldsymbol{q}$ in the training samples, respectively.

The FGO has better performance compared with GD in finding $\mathcal{D}_{min}$ when the number of training samples M for the hypersurface (19) is large enough, as shown in Table I. FGO and GD have almost the same computational cost, i.e., $< 0.01s$ for both. But this advantage decreases when the classification accuracy of the hypersurface (19) further increases, which may be due to over-fitting. One comparison example between FGO and GD search paths is shown in Fig. 3(a), 3(b). Note that we can combine them to get improved capability to search for $\boldsymbol{p}^*, \boldsymbol{q}^*$. If we apply the FGO method to the good results



(a) FGO pre-training map with feasible example trajectories.



(b) Comparison of robot paths between CBF, A* and RRT.

Fig. 4. Case study setup and planning frameworks comparison.

from GD, the additional improvement percentage is around 5% among all the testing samples.

We have implemented the learned optimal HOCBF parameters $(p_1^*, p_2^*, q_1^*, q_2^*) = (0.7426, 1.9745, 1.9148, 0.7024)$ in the definition of all the HOCBFs for all obstacles in a robot exploration problem in an unknown environment. We should also note that the optimal penalties and powers are not unique. All the circular obstacles are with different size to test the robustness of the penalty method with the learned optimal parameters, and are static but randomly distributed. The robot can safely avoid all the obstacles and arrive at its destination if the obstacles do not form traps such that the robot has no way to escape, as shown in Fig. 4(b).

We also compared the CBF-based robot exploration framework with the RRT [37] and A* [38] algorithms by considering the configuration shown in Fig. 4(b). The pre-training for the CBF-based method could be several hours. Both the RRT and A* algorithms have global environment information such that they tend to choose shorter-length trajectories compared with the CBF method. But this advantage disappears if the environment is changing fast, in which case the CBF method tends to be more robust and computationally efficient. Comparisons based on four different criteria are shown in Table II. In a dynamic environment, the RRT and A* algorithms need to replan their path at each time step, but the CBF method does not need to do this. Therefore, we can see that the CBF-based framework is able to better adjust to changes in the environment and is computationally efficient.

B. Sampling Learning Case Studies

We chose all class $\mathcal{K}$ functions in the definitions of all HOCBFs as linear functions, and used SVM to classify the feasible and infeasible sets for the QP (13) or (24). We obtained a hypersurface for each type of obstacle, and apply this hypersurface to the same type obstacles with unknown locations.

1) Regular Obstacles: We consider three regular obstacles (locations unknown to the robot, but detected by the on-board sensor with sensing range $> 6m$ in radius), and solve the QP (13) or (24) one time step forward to check the feasibility of the samples. Some other parameters are: $r = 7m, \varepsilon = 0.001, \xi = 1m, x_A = 32m, y_A = 25m, x_B = 20m, y_B =$

TABLE I
COMPARISONS BETWEEN THE GD AND FGO ALGORITHMS

items	GD	FGO							
Training sample number M		500	1000	1500	2000	2500	3000	3500	4000
Classification accuracy		0.879	0.927	0.939	0.953	0.960	0.963	0.966	0.970
Better than GD percentage		0.210	0.248	0.254	0.252	0.244	0.282	0.288	0.266
Worse than GD percentage		0.270	0.190	0.232	0.204	0.218	0.218	0.240	0.240
$\mathcal{D}_{min}/m$ (samples min.: 5.0)	4.6	4.6	4.6	4.6	4.8	4.6	4.6	4.6	4.6

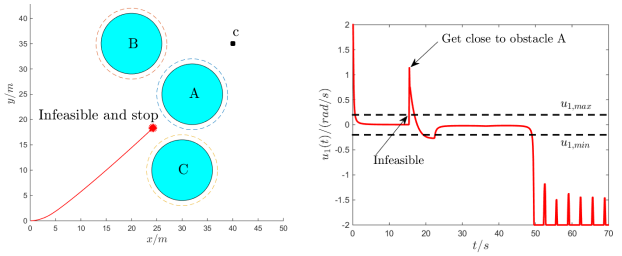
TABLE II
PERFORMANCE COMPARISON BETWEEN CBF, A* AND RRT

item	Compute time	safety guarantee	Environment knowledge	pre-training
CBF	$< 0.01s$	Yes	not required	required
A*	$1.3s$	No	required	not required
RRT	$0.3s$	No	required	not required

TABLE III
TRAINING RESULTS FOR THE REGULAR OBSTACLE

iter.	QP (24) inf. rate	classi. accu.	train	test
1	0.0665	0.8490	2k	1k(20k)
2	0.0528	0.9150	0.8k	0.2k (10k)
3	0.0058	0.9600	0.6k	0.2k (60k)
test	0.0004			25k
gen.	0	1.0000		20k

$35m, x_C = 30m, y_C = 10m$, and the map of the environment is shown in Fig. 5(a).



(a) Trajectories (b) Control $u_1(t)$

Fig. 5. The control profiles for the infeasible example with relaxation on both control limitations (30). The control bound for $u_2(t)$ is satisfied.

Assume the robot destination is $(40m, 35m)$. If the QP (13) is solved with all the objectives and constraints defined as (25)-(30), it will become infeasible when the robot gets close to the obstacle, as shown in Fig. 5(a). In order to show how these two control inputs may lead to the infeasibility of the QP (13), we relax both limitations (30) and show the control profile in Fig. 5(b).

To solve this infeasiblity problem, we apply the learning method introduced in Sec. IV. During pre-training, we arbitrarily assign the location $(x_o, y_o) := (20m, 35m)$ of an obstacle that is the same type (circle with the same radius) as $j \in S_t$. Then we define $\boldsymbol{z} := (x - x_o, y - y_o, \theta, v)$ as input for SVM with polynomial kernel of degree 2, i.e., the kernel $k(\boldsymbol{y}, \boldsymbol{z})$ is defined as:

$$k(\boldsymbol{y}, \boldsymbol{z}) = (k_1 + k_2 \boldsymbol{y}^T \boldsymbol{z})^2. \tag{32}$$

where $\boldsymbol{y}$ denotes an input vector similar to $\boldsymbol{z}$, $k_1 \in \mathbb{R}, k_2 \in \mathbb{R}$.

We set $k_1 = 0.9, k_2 = 0.4$ for the kernel (32) in the feedback training process. Each training iteration for the regular obstacle is shown in Fig. 6.

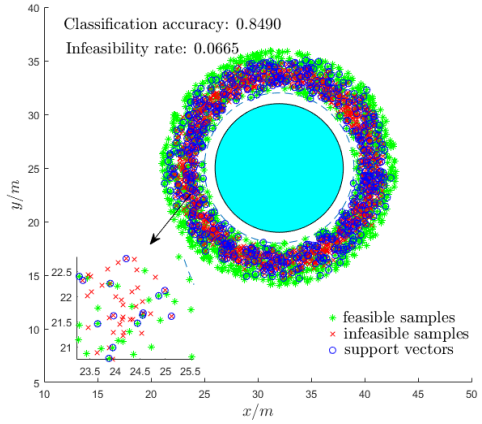
As shown in Fig. 6, the classification accuracy for the infeasible and feasible samples is improved after each iteration, and the infeasibility rate also decreases. It becomes inefficient to get infeasible samples after just three iterations since we want to get an unbiased training data set and the infeasibility rate is 0.0004 (improved 166 times compared to the first iteration). The training results are shown in Table III (the first number of test samples denotes the test number for the classification accuracy, and the number in the bracket denotes the test number for the infeasibility rate). Eventually, we get a hypersurface $H(\boldsymbol{z})$ from the kernel.

Hypersurface generalization: A natural question is what happens if the robot starts at points outside the sample area (we only sampled around the obstacle with location radius within $[7, 13]$, and achieved 96.00% classification accuracy over the test samples and 0.04% infeasibility rate for the QP (24)). Therefore, we tested the classification accuracy for samples with location radius within $[13, 32]$ (out of the train and test sets). The test accuracy is **100% (i.e., $H(\boldsymbol{z}) \geq 0$) and the infeasibility rate of the QP (24) is 0%** (over 20000 samples) for the obstacle (listed in the last row of Table III), which shows good generalization of this hypersurface.

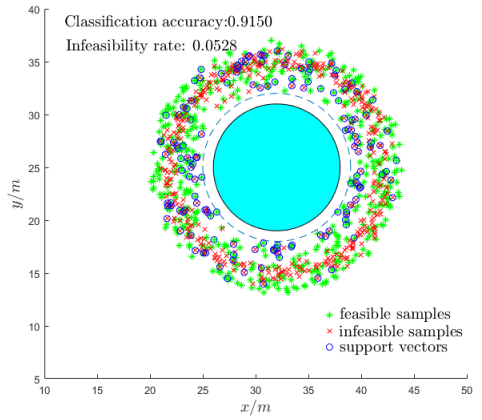
We apply this hypersurface to the robot control problem with the objective and constraints defined as (25)-(30) and test 8 different destinations a, b, c, d, e, f, g, h shown in Fig. 7. The initial conditions at time $0s$ are $\boldsymbol{x}(0) = (0m, 0m, 0rad, 1m/s)$ (out of the training and test sets), and $t_f = 70s$. The QPs (24) are always feasible during $[0s, 70s]$. The obstacles are safely avoided, and the robot eventually arrives at the destination. We present the two control input profiles in Fig. 7.

2) Irregular Obstacle: In this case, we consider an irregular obstacle that is formed by two overlapped disks (with locations $(22m, 28m)$ and $(31m, 19m)$ but unknown to the robot), as shown in Fig. 8. We apply the learning method introduced in Sec. IV to recursively improve the problem feasibility, and possibly to escape from local traps. We formulate a receding horizon control of $H_t = 60$, and check the feasibility of all these H_t step QPs. The other settings are the same as the ones from the regular case. Each training iteration is shown in Fig. 8.

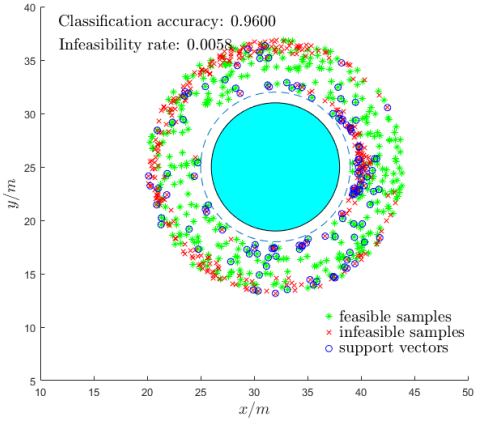
As shown in Fig. 8, the classification accuracy and the infeasibility rate change similarly to the regular obstacle case in each iteration. The training results for the irregular obstacle are shown in Table IV. We also apply this hypersurface to the



(a) 1st iter. (QP (13)).

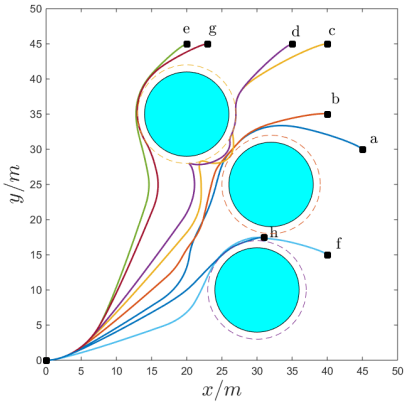


(b) 2nd iter. (QP (24)).

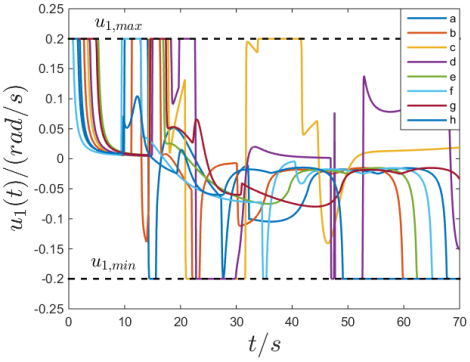


(c) 3rd iter. (QP (24)).

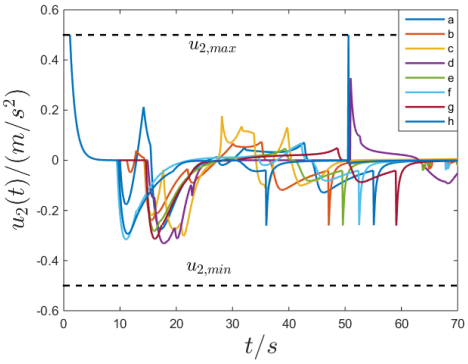
Fig. 6. Feedback training process for the regular obstacle. All data are sampled around the obstacle (solid circle in all sub-figures). Each sample is a four dimensional point, but is visualized in $x - y$ plane.



(a) Trajectories



(b) Control $u_1(t)$

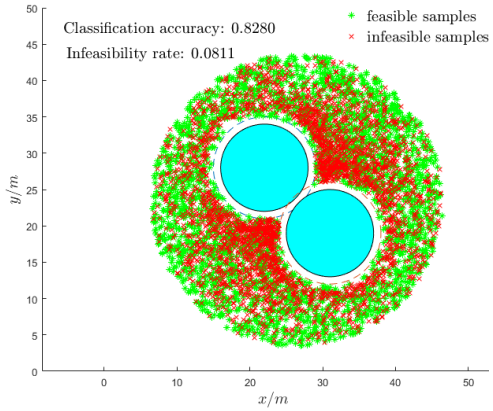


(c) Control $u_2(t)$

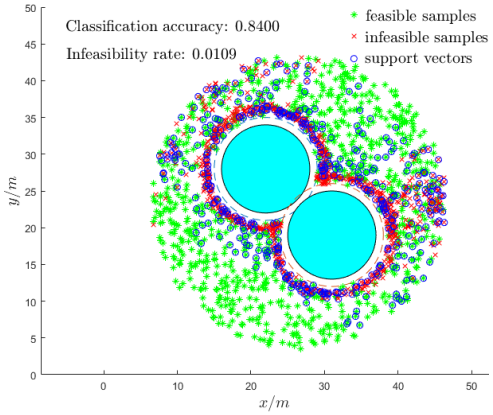
Fig. 7. Robot control problem feasibility test after learning in an environment with regular obstacles.

TABLE IV
TRAINING RESULTS FOR THE IRREGULAR OBSTACLE

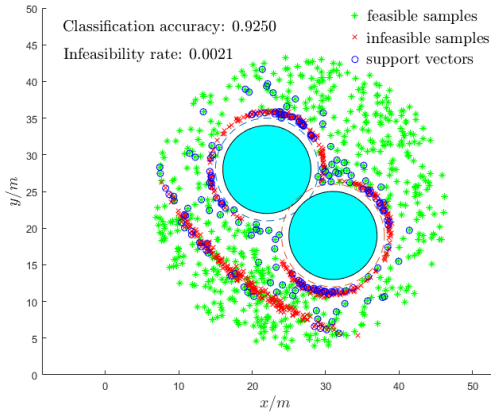
iter.	QP (24) inf. rate	classi. accu.	train	test
1	0.0811	0.8280	5k	1k(36k)
2	0.0109	0.8400	1.2k	0.8 (90k)
3	0.0021	0.9250	1k	0.2 (270k)
test	0.0004			100k
gen.	0	1.0000		100k



(a) 1st iter. (QP (13)).

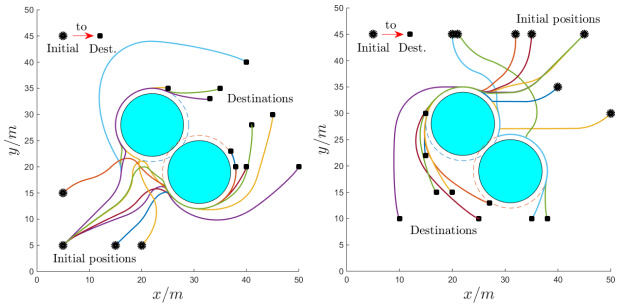


(b) 2nd iter. (QP (24)).



(c) 3rd iter. (QP (24)).

Fig. 8. Feedback training process for the irregular obstacle. All data are sampled around the obstacle (solid circle in all sub-figures). Each sample is a four dimensional point, but is visualized in $x-y$ plane.



(a) Trajectories case 1 (b) Trajectories case 2

Fig. 9. Robot control problem feasibility and reachability test after learning in irregular obstacle case.

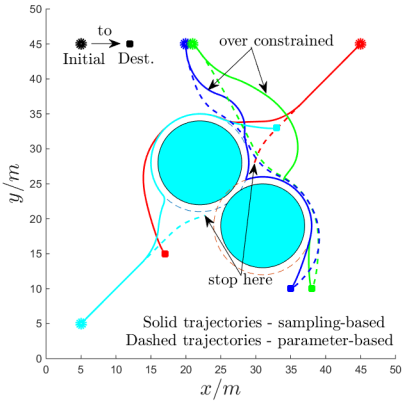


Fig. 10. Trajectory comparisons between parameter learning and sampling learning approaches.

robot control problem, and test the feasibility and reachability. The QPs (24) are always feasible on the path from the initial positions to destinations. The obstacles are safely avoided, and the robot can reach the destinations in most cases. We present the results in Fig. 9(a)-(b). The robot can safely avoid the local traps formed by these two circle obstacles, and thus, reachability is also improved in addition to feasibility.

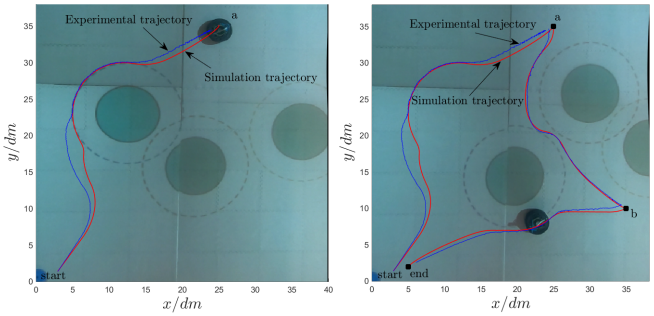
Comparison between parameter and sampling learning: The sampling learning method can easily get over-constrained such that the optimization (24) may generate unreasonable trajectories as shown in Fig. 9(b) (blue and green solid trajectories), while the parameter learning approach can avoid such problems. However, the parameter learning cannot escape from the local traps, and the QPs can easily become infeasible at these local traps, as the dashed red and cyan trajectories shown in Fig. 10. We can limit the applied region of the feasibility constraint to alleviate the over-constrained problem in the sampling learning approach.

C. Application to a Robot Control Problem

We implemented our approaches (parameter learning for regular obstacles and sampling learning for irregular obstacles) in a laboratory experimental setup using an *iRobot Create 2*. The experiment field has a size of 5×5 meters, and is equipped with an opti-track motion caption system to detect

the robot position and heading angle, as well as the location of the obstacles. The robot's linear speed is obtained by its own sensor. At the beginning of each control time interval ($\Delta t = 0.1s$), the controller receives the robot state from the opti-track and its own sensors, multiplies the position and speed by 10 (since the simulation is in a 50×50 meter field), and then solves the CBF-CLF based QP and gets an optimal control (rotation speed $u_1^*(t)$ and acceleration $u_2^*(t)$). Since *iRobot Create 2* takes the linear speed and the rotation speed as control inputs, we get the optimal speed as $v^*(t) = v(t) + u_2^*(t)\Delta t/10$, and then apply $v^*(t)$ and $u_1^*(t)$ to the robot in the following Δt.

The specification is to go to point $a := (25dm, 35dm)$ and $b := (35dm, 10dm)$, and then go back to the start point. When the robot arrives at point a, the map changes from 1 to 2, as shown in Fig. 11. The position and heading angle uncertainties (from the opti-track) are about $0.002m$ ($0.02m$ for the QP) and $0.005rad$, respectively, which are relatively small. However, the uncertainty of the linear speed from the robot sensors is about $0.03m/s$ ($0.3m/s$ for the QP). Therefore, we relax $u_{2,min}$ by $-0.2m/s^2$ in order to consider these noises, as well as system dynamics inaccuracy and other noise terms (such as the battery level of the robot). The comparison between the simulated trajectory and the experimental trajectory is shown in Fig. 11.



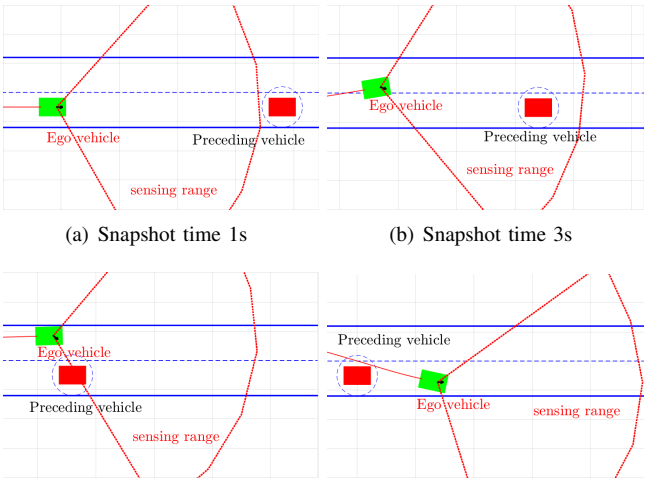
(a) Robot trajectory in map 1 (b) Robot trajectory from map 1 to 2

Fig. 11. Experimental results using an *iRobot Create 2*.

D. Application to Autonomous Driving

In autonomous driving, the ego vehicle treats all the other vehicles as moving obstacles [41]. We consider the same dynamics and constraints as in (25)-(30), except relaxing the maximum speed limit to $28m/s$. We use the sampling learning approach to recursively improve the QP feasibility with respect to moving obstacles. As all the vehicle types (such as size) are known, we can learn a feasibility constraint for each type of vehicle, and then apply this feasibility constraint to the QP. We fully cover the other vehicle by a disk, and only consider the distance between the center of the ego vehicle and the disk.

In the learning process, we take the relative position and relative speed between the ego vehicle and the other vehicle in both along-lane and lateral directions, and the heading of the ego vehicle as the inputs for the SVM model (32). The relative speed difference is sampled between $0m/s$ and $20m/s$. The feedback learning process is similar to Table III, but takes 6 iterations in order to achieve an infeasible rate that is smaller than ε. In the ego vehicle overtaking test, we set the initial and desired speeds for the ego vehicle as $28m/s$, while the other vehicle runs at a constant speed $16m/s$. Note that the control bounds for both u_1, u_2 are very tight, as shown in (30). Without the learned feasibility constraint, the QP will be infeasible at some time instant for the ego vehicle. However, the ego vehicle can successfully overtake the other vehicle and the QP is always feasible with the learned feasibility constraint, as the snapshot shown in Fig. 12.



(a) Snapshot time 1s (b) Snapshot time 3s

(c) Snapshot time 7s (d) Snapshot time 10s

Fig. 12. Sampling learning approach applied in autonomous driving for the overtaking of another moving vehicle.

VII. Conclusion

We presented a tutorial of a set of machine learning techniques for the CBF-CLF-based QPs corresponding to optimal control problems with safety constraints and control limitations. The parameter learning approach improves the feasibility of the QPs for regular unsafe sets (i.e., sets for which the feasibility does not depend on the initial condition). The sampling-based learning approach can deal with irregular (i.e., not regular) unsafe sets . The simulation and experimental results on a robot control problem and an autonomous driving case study show good feasibility performance with the proposed approaches. Future work will focus on dynamics and datasets affected by noise, addressing the inter-sampling effect of the QP-based approach, and on the combination of barrier-based methods with deep learning systems [42].

References

[1] 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.

[2] W. Xiao and C. Belta, "Control barrier functions for systems with high relative degree," in *Proc. of 58th IEEE Conference on Decision and Control*, Nice, France, 2019, pp. 474–479.

[3] K. P. Tee, S. S. Ge, and E. H. Tay, "Barrier lyapunov functions for the control of output-constrained nonlinear systems," *Automatica*, vol. 45, no. 4, pp. 918–927, 2009.

[4] P. Wieland and F. Allgower, "Constructive safety using control barrier functions," in *Proc. of 7th IFAC Symposium on Nonlinear Control System*, 2007.

[5] S. P. Boyd and L. Vandenberghe, *Convex optimization*. New York: Cambridge university press, 2004.

[6] E. Sontag, "A lyapunov-like stabilization of asymptotic controllability," *SIAM Journal of Control and Optimization*, vol. 21, no. 3, pp. 462–471, 1983.

[7] R. A. Freeman and P. V. Kokotovic, *Robust Nonlinear Control Design*. Birkhauser, 1996.

[8] A. D. Ames, K. Galloway, and J. W. Grizzle, "Control lyapunov functions and hybrid zero dynamics," in *Proc. of 51rd IEEE Conference on Decision and Control*, 2012, pp. 6837–6842.

[9] K. Galloway, K. Sreenath, A. D. Ames, and J. Grizzle, "Torque saturation in bipedal robotic walking through control lyapunov function based quadratic programs," *preprint arXiv:1302.7314*, 2013.

[10] P. Glotfelter, J. Cortes, and M. Egerstedt, "Nonsmooth barrier functions with applications to multi-robot systems," *IEEE control systems letters*, vol. 1, no. 2, pp. 310–315, 2017.

[11] Q. Nguyen and K. Sreenath, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *Proc. of the American Control Conference*, 2016, pp. 322–328.

[12] A. S. C. Armijos, A. Li, C. G. Cassandras, Y. K. Al-Nadawi, H. Araki, B. Chalaki, E. Moradi-Pari, H. N. Mahjoub, and V. Tadiparthi, "Cooperative energy and time-optimal lane change maneuvers with minimal highway traffic disruption," *Automatica*, vol. 165, p. 111651, 2024.

[13] A. Li, A. S. C. Armijos, and C. G. Cassandras, "Cooperative lane changing in mixed traffic can be robust to human driver behavior," in *2023 62nd IEEE Conference on Decision and Control (CDC)*. IEEE, 2023, pp. 5123–5128.

[14] A. Li, C. G. Cassandras, and W. Xiao, "Safe optimal interactions between automated and human-driven vehicles in mixed traffic with event-triggered control barrier functions," in *2024 American Control Conference (ACC)*. IEEE, 2024, pp. 1455–1460.

[15] Z. J. Patterson, W. Xiao, E. Sologuren, and D. Rus, "Safe control for soft-rigid robots with self-contact using control barrier functions," in *2024 IEEE 7th International Conference on Soft Robotics (RoboSoft)*. IEEE, 2024, pp. 151–156.

[16] S. Ceron, W. Xiao, and D. Rus, "Reciprocal and non-reciprocal swarmalators with programmable locomotion and formations for robot swarms," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 12 233–12 239.

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

[18] M. Srinivasan, A. Dabholkar, S. Coogan, and P. A. Vela, "Synthesis of control barrier functions using a supervised machine learning approach," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 7139–7145.

[19] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 708–717.

[20] B. T. Lopez, J. J. E. Slotine, and J. P. How, "Robust adaptive control barrier functions: An adaptive and data-driven approach to safety," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 1031–1036, 2020.

[21] S. Yaghoubi, G. Fainekos, and S. Sankaranarayanan, "Training neural network controllers using control barrier functions in the presence of disturbances," in *IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–6.

[22] A. Orthey and O. Stasse, "Towards reactive whole-body motion planning in cluttered environments by precomputing feasible motion spaces," in *In IEEE-RAS Int. Conf. on Humanoid Robotics*, 2013.

[23] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *In IEEE-RAS Int. Conf. on Robotics and Automation*, 2003.

[24] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, "Fast direct multiple shooting algorithms for optimal robot control," in *In Fast motions in biomechanics and robotics*, Springer, 2006, pp. 65–93.

[25] W. Xiao, C. Belta, and C. G. Cassandras, "Sufficient conditions for feasibility of optimal control problems using control barrier functions," *Automatica*, vol. 135, p. 109960, 2022.

[26] J. Carpentier, R. Budhiraja, and N. Mansard, "Learning feasibility constraints for multi-contact locomotion of legged robots," in *Robotics: Science and Systems*, Cambridge, MA, 2017.

[27] N. Perrin, O. Stasse, L. Baudouin, F. Lamiraux, and E. Yoshida, "Fast humanoid robot collision-free footstep planning using swept volume approximations," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 427–439, 2012.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, and A. A. R. et.al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, 2015.

[29] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *preprint arXiv:1602.01783*, 2016.

[30] W. Xiao, C. Belta, and C. G. Cassandras, "Feasibility guided learning for constrained optimal control problems," in *Proc. of 59th IEEE Conference on Decision and Control*, 2020, pp. 1896–1901.

[31] W. Xiao, C. G. Cassandras, and C. A. Belta, "Learning feasibility constraints for control barrier functions," in *2023 European Control Conference (ECC)*. IEEE, 2023, pp. 1–6.

[32] W. Xiao and C. Belta, "High-order control barrier functions," *IEEE Transactions on Automatic Control*, vol. 67, no. 7, pp. 3655–3662, 2021.

[33] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, third edition, 2002.

[34] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 96–101, 2019.

[35] G. Yang, C. Belta, and R. Tron, "Self-triggered control for safety critical systems using control barrier functions," in *Proc. of the American Control Conference*, 2019, pp. 4454–4459.

[36] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[37] S. M. LaValle, J. Kuffner, and J. James, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

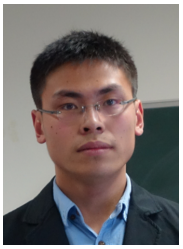
[38] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *EEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[39] P. M. Vaidya, "Speeding-up linear programming using fast matrix multiplication," in *30th Annual Symposium on Foundations of Computer Science*, 1989, pp. 332–337.

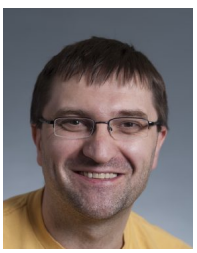
[40] S. Wang, Y. Wang, D. Li, and Q. Zhao, "Distributed relative localization algorithms for multi-robot networks: A survey," *Sensors*, vol. 23, no. 5, p. 2399, 2023.

[41] A. Li, A. S. C. Armijos, and C. G. Cassandras, "Robust optimal lane-changing control for connected autonomous vehicles in mixed traffic," *Automatica*, vol. 174, p. 112169, 2025.

[42] W. Xiao, T.-H. Wang, R. Hasani, M. Lechner, Y. Ban, C. Gan, and D. Rus, "On the forward invariance of neural odes," in *International conference on machine learning*. PMLR, 2023, pp. 38 100–38 124.



Wei Xiao (S'19) is currently a postdoctoral associate at Massachusetts Institute of Technology. He received a B.Sc. degree from the University of Science and Technology Beijing, China in 2013, a M.Sc. degree from the Chinese Academy of Sciences (Institute of Automation), China in 2016, and a Ph.D. degree from the Boston University, Brookline, MA, USA in 2021. His research interests include control theory and machine learning, with particular emphasis on robotics and traffic control. He received an Outstanding Student Paper Award at the 2020 IEEE Conference on Decision and Control.



Calin Belta (F'17) is the Brendan Iribe Endowed Professor of Electrical and Computer Engineering and Computer Science at the University of Maryland, College Park. His research focuses on dynamics and control theory, with particular emphasis on cyber–physical systems, formal methods, and applications to robotics and systems biology. Notable awards include the 2008 AFOSR Young Investigator Award, the 2005 National Science Foundation CAREER Award, and the 2017 IEEE TCNS Outstanding Paper Award. He is a Fellow of the IEEE and a Distinguished Lecturer of the IEEE CSS.

Christos G. Cassandras (F'96) is Distinguished Professor of Engineering at Boston University. He is Head of the Division of Systems Engineering, Professor of Electrical and Computer Engineering, and co-founder of Boston University's Center for Information and Systems Engineering (CISE). Hereceived degrees from Yale University, Stanford University, and Harvard University. In 1982-1984 he was with ITP Boston, Inc. where he worked on the design of automated manufacturing systems. In 1984-1996 he was a faculty member at the Department of Electrical and Computer Engineering, University of Massachusetts/Amherst. He specializes in the areas of discrete event and hybrid systems, cooperative control, stochastic optimization, and computer simulation, with applications to computer and sensor networks, manufacturing systems, and transportation systems. He has published over 500 refereed papers in these areas, and six books. He has guest-edited several technical journal issues and currently serves on several journal Editorial Boards, including Editor of Automatica. In addition to his academic activities, he has worked extensively with industrial organizations on various systems integration projects and the development of decision support software. He has most recently collaborated with The MathWorks, Inc. in the development of the discrete event and hybrid system simulator SimEvents.

Dr. Cassandras was Editor-in-Chief of the IEEE Transactions on Automatic Control from 1998 through 2009 and has also served as Editor for Technical Notes and Correspondence and Associate Editor. He was the 2012 President of the IEEE Control Systems Society (CSS). He has also served as Vice President for Publications and on the Board of Governors of the CSS, as well as on several IEEE committees, and has chaired several conferences. He has been a plenary/keynote speaker at numerous international conferences, including the 2017 IFAC World Congress, the American Control Conference in 2001 and the IEEE Conference on Decision and Control in 2002 and 2016, and has also been an IEEE Distinguished Lecturer.

He is the recipient of several awards, including the 2011 IEEE Control Systems Technology Award, the Distinguished Member Award of the IEEE Control Systems Society (2006), the 1999 Harold Chestnut Prize (IFAC Best Control Engineering Textbook) for Discrete Event Systems: Modeling and Performance Analysis, a 2011 prize and a 2014 prize for the IBM/IEEE Smarter Planet Challenge competition, the 2014 Engineering Distinguished Scholar Award at Boston University, several honorary professorships, a 1991 Lilly Fellowship and a 2012 Kern Fellowship. He is a member of Phi Beta Kappa and Tau Beta Pi. He is also a Fellow of the IEEE and a Fellow of the IFAC.