2023 European Control Conference (ECC)
June 13-16, 2023. Bucharest, Romania

Learning Feasibility Constraints for Control Barrier Functions

Wei Xiao, Christos G. Cassandras, and Calin A. Belta

Abstract— **It has been shown that optimizing quadratic costs while stabilizing affine control systems to desired (sets of) states subject to state and control constraints can be reduced to a sequence of Quadratic Programs (QPs) by using Control Barrier Functions (CBFs) and Control Lyapunov Functions (CLFs). In this paper, we 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 required. To this end, we propose a sampling-based learning approach to learn a new feasibility constraint for CBFs; this constraint is then enforced by another HOCBF added to the QPs. The accuracy of the learned feasibility constraint is recursively improved by a recurrent training algorithm. We demonstrate the advantages of the proposed learning approach to constrained optimal control problems with specific focus on a robot control problem and on autonomous driving in an unknown environment.**

I. Introduction

With increased interest in autonomous systems, optimal control problems over long finite horizons 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].

BFs are Lyapunov-like functions [3], [4], whose use can be traced back to optimization problems [5]. More recently, they have been employed to prove set invariance [6], [7]. Control BFs (CBFs) are extensions of BFs for control systems that are used to map a constraint defined over system states onto a constraint on the control input. 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) [8] to form quadratic programs (QPs) [1], [9] that are solved in real time. These CBFs work for constraints that have relative degree one. A more general form [10] for arbitrarily high relative degree constraints, termed exponential CBF, employs input-output linearization and finds a pole placement controller with negative poles. The high order CBF (HOCBF) proposed in [2] is simpler and more general than the exponential CBF [10].

One of the remaining challenges to be addressed in the CBF-based method includes the feasibility of the associated QPs when both state constraints and control bounds are involved. Infeasibility can be avoided by precomputing feasible motion spaces [11] and using a receding horizon scheme as in Model Predictive Control (MPC) [12]. Optimal control methods can also avoid the infeasibility problem, but are hard to implement 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. As an example, for the adaptive cruise control (ACC) problem defined in [1], the infeasibility issue is addressed by considering the minimum braking distance. This approach does not scale well for high-dimensional systems. The penalty method proposed in [13] 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 [14] expressed themselves as CBFs; however, such conditions are usually hard to find for a general problem.

The use of machine learning techniques to obtain feasible solutions was recently proposed for legged robots. Feasibility constraints for probabilistic models are learned in [15] based on simplified models. Since the learned constraints are complex, they are simplified by expectation-maximization. Robot footstep limits are modeled as hyperplanes based on success and failure datasets in [16]. Reinforcement learning (RL) [17] has the potential to address the infeasibility issue for optimal control problems, but it is difficult to quantify infeasibility as a reward; moreover, the optimized parameters may drive the system to a local infeasible region where a feasible solution may never be found.

The main contribution of this paper is to address the CBF-based QP infeasibility in 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 a general-purpose optimal control problem. Unsafe sets are assumed to belong to a finite collection of sets whose geometries are known in advance, whereas their locations are only detected during system operation. Our approach proceeds by learning a new feasibility constraint that guarantees feasibility. Specifically, for each type of unsafe set, we sample the state space of the system in its proximity, check for feasibility of the QP

This work was supported in part by NSF under grants ECCS-1931600, DMS-1664644, CNS-1645681, IIS-1723995, and IIS-2024606, by ARPAE under grant DE-AR0001282, by AFOSR under grant FA9550-19-1-0158, and by the MathWorks.

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

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

one step forward for *regular* unsafe sets and multiple steps forward for *irregular* unsafe sets (precise definitions are provided in the sequel), and learn a differentiable classifier (for feasible and infeasible states) that is then added to the set of initial constraints. We validate the effectiveness of the proposed learning-based approach on a robot control problem in an unknown environment, as well as on an application in autonomous driving, in which moving obstacles (such as other vehicles) are usually involved.

II. PRELIMINARIES

We assume the reader is familiar with the definitions of a class $\mathcal{K}$ function, relative degree of a (sufficiently many times) differentiable function or constraint, and forward invariance of a set with respect to given dynamics; otherwise, please refer to [2] for details.

Consider an affine control system of the form

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

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 $\boldsymbol{u}_{min}, \boldsymbol{u}_{max} \in \mathbb{R}^q$ and the inequalities are interpreted componentwise.

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 functions $\psi_i : \mathbb{R}^n \to \mathbb{R}, i \in \{1, \dots, m\}$:

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

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:

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

Definition 1: (*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

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

for all $\boldsymbol{x} \in C_1 \cap \dots \cap C_m$. In (5), L_f^m (L_g) denotes Lie derivatives along f (g) m (one) times, 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 $C_1 \cap \dots \cap C_m$.

The HOCBF is a general form of the relative degree one CBF [1], [9] (setting $m = 1$ reduces the HOCBF to the common CBF form in [1], [9]), and it is also a general form of the exponential CBF [10]. We can define $\alpha_i(\cdot)$ in Def. 1 to be extended class $\mathcal{K}$ functions to ensure robustness of a HOCBF to perturbations [1].

Theorem 1: ([2]) Given a HOCBF $b(\boldsymbol{x})$ from Def. 1 with the associated sets $C_1 \dots C_m$ defined by (4), if $\boldsymbol{x}(0) \in C_1 \cap, \dots, \cap C_m$, then any Lipschitz continuous controller $\boldsymbol{u}(t)$ that satisfies the constraint in (5), $\forall t \geq 0$ renders $C_1 \cap \dots \cap C_m$ forward invariant for system (1).

Definition 2: (*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 $\forall \boldsymbol{x} \in \mathbb{R}^n$, $c_1||\boldsymbol{x}||^2 \leq V(\boldsymbol{x}) \leq c_2||\boldsymbol{x}||^2$,

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

Many existing works [1], [10], [18] 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 [18]. If convergence to a state is desired, then a CLF constraint of the form (6) is added, as in [1] [18]. 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:

$$\begin{aligned}&\min_{\boldsymbol{u}(\omega\Delta t), \delta(\omega\Delta t)} \boldsymbol{u}^T(\omega\Delta t) H \boldsymbol{u}(\omega\Delta t) + p_0\delta^2(\omega\Delta t)\\ &\text{s.t. } (5), \boldsymbol{u} \in U, L_f V(\boldsymbol{x}) + L_g V(\boldsymbol{x})\boldsymbol{u} + \epsilon V(\boldsymbol{x}) \leq \delta,\end{aligned} \tag{7}$$

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:

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

where $||\cdot||$ denotes the 2-norm of a vector; t_f denotes a given final time; and $\mathcal{C}$ is a strictly increasing function of its argument (usually 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:

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

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

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.

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.

Approach: The approach to Problem 1 proposed in [1] is based on the approach 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 3: (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*.

When there are multiple unsafe sets, whether each one is regular or not is 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 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; for example, a regular circular unsafe set can be used to over-approximate any shape of unsafe sets. In this paper, we limit ourselves to a set of known unsafe set types (a typical application is in autonomous driving where the vehicle types are known).

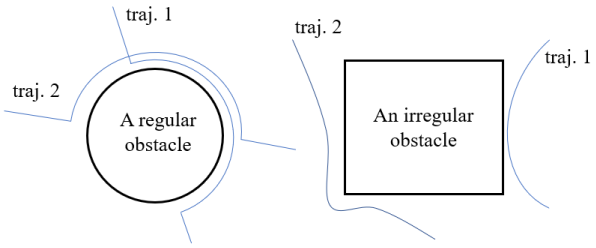


Fig. 1. Regular and irregular obstacle (unsafe set) examples for a robot with nonholonomic dynamics. The robot needs the same control input effort (from the CBF-based QPs) in order to avoid the regular circular obstacle, regardless of where the robot is initially located, as shown in example trajectories 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).

We propose a learning-based approach, specifically a classifier, to ensure the feasibility for a certain type of unsafe set.

IV. Sampling-Based Learning Approach

In this section, we show how we can deal with irregular unsafe sets as defined in Def. 3. This approach also works for regular unsafe sets, but tends to be conservative. Recall that the type of every unsafe set $i \in S$ is already known in an unknown environment, 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 (7) 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 (7), 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 (7) becomes infeasible. The main idea of the sampling-based learning approach is to partition the state space of system (1) into sets in which the QP (7) is feasible or infeasible after a certain number of time steps.

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 center 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 (7) at time 0 according to the geometry of the unsafe set:

- (i) **Regular unsafe set:** we solve the QP (7) at time 0 for 1 time step forward.
- (ii) **Irregular unsafe set:** we solve the QP (7) at time 0 for $H_t \in \mathbb{N} > 1$ time steps forward.

Remark 1: Unlike regular unsafe sets, when dealing with irregular unsafe sets, the system may get stuck at local traps. This is why we extend the solution of the QP (7) 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 (7) infeasible, especially when the system state approaches 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 (7) $H_t > 1$ 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)) (7) 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:

$$H_j(\boldsymbol{z}) : \mathbb{R}^n \to \mathbb{R}, \tag{11}$$

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 (11) is $\gamma \in \mathbb{N}$, we define the set of all control values that satisfy $H_j(\boldsymbol{z}(t)) \geq 0$ as:

$$\begin{aligned} K^j_{fea} = \{\boldsymbol{u} \in U : L_f^\gamma H_j(\boldsymbol{z}) + L_g L_f^{\gamma-1} H_j(\boldsymbol{z})\boldsymbol{u} \\ + O(H_j(\boldsymbol{z})) + \alpha_\gamma(\psi_{\gamma-1}(\boldsymbol{z})) \geq 0\} \end{aligned} \tag{12}$$

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 4: 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: 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.

Proof: By Theorem 5 in [2] and $H_j(\boldsymbol{z}(0)) \geq 0, \forall j \in S_t$, any control input that satisfies $\boldsymbol{u}(t) \in K^j_{fea}, \forall j \in S_t, \forall t \in [0, \infty]$ makes $H_j(\boldsymbol{z}(t)) \geq 0, \forall j \in S_t, \forall t \in [0, \infty]$. Since $H_j(\boldsymbol{z}), \forall j \in S_t$ classifies the state space of system (1) into feasible and infeasible spaces for Problem 1 with 100% accuracy, we have that Problem 1 is feasibility forward invariant for system (1). ■

Naturally, machine learning techniques cannot ensure 100% classification accuracy. We introduce an approach based on feedback training to improve the 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.

Similar to the QP (7), we have a feasible reformulated problem at $t = \omega \Delta t$ ($\omega = 0, 1, 2, \ldots, \frac{t_f}{\Delta t} - 1$):

$$\begin{aligned} \min_{\boldsymbol{u}(t), \delta(t)} \; & \boldsymbol{u}^T(t) H \boldsymbol{u}(t) + p_0 \delta^2(t) \\ \text{s.t. } & (5), \boldsymbol{u} \in U, L_f V(\boldsymbol{x}) + L_g V(\boldsymbol{x}) \boldsymbol{u} + \epsilon V(\boldsymbol{x}) \leq \delta, \\ & L_f^\gamma H_j(\boldsymbol{z}) + L_g L_f^{\gamma-1} H_j(\boldsymbol{z}) \boldsymbol{u} + O(H_j(\boldsymbol{z})) + \alpha_\gamma(\psi_{\gamma-1}(\boldsymbol{z})) \geq 0 \end{aligned} \tag{13}$$

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 (11). 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 (11) into the QP (7) (i.e., use (13)) 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. The workflow is shown in Fig. 2. The infeasibility rate is the ratio of the number of infeasible samples over the total times of solving the QP (7) or (13).

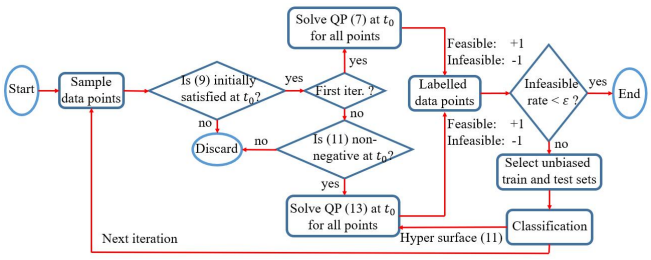


Fig. 2. Feedback training workflow for unsafe set $j \in S_t$ ($\varepsilon > 0$ denotes the termination threshold).

V. Implementation and Case Studies

We implemented the proposed learning approach in MATLAB and performed simulations for a robot control problem. 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: $\dot{x} = v\cos\theta, \dot{y} = v\sin\theta, \dot{\theta} = u_1, \dot{v} = u_2$, 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.

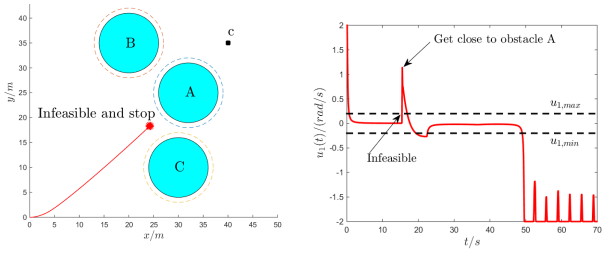
We instantiate cost (8) in the form: $\min_{\boldsymbol{u}(t)} \int_0^{t_f} \left[u_1^2(t) + u_2^2(t)\right] dt + p_0((x(t_f) - x_d)^2 + (y(t_f) - y_d)^2)$. In other words, we wish to minimize the energy consumption and drive the robot to a given destination $(x_d, y_d) \in \mathbb{R}^2$, i.e., drive $(x(t), y(t))$ to $(x_d, y_d), \forall t \in [t', t_f]$, for some $t' \in [0, t_f]$, as defined in (8).

The safe sets corresponding to (9) are defined as circular (regular) obstacles $\sqrt{(x(t) - x_i)^2 + (y(t) - y_i)^2} \geq r, \forall i \in S$, 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: $V_{min} \leq v(t) \leq V_{max}$, $u_{1,min} \leq u_1(t) \leq u_{1,max}, u_{2,min} \leq u_2(t) \leq u_{2,max}$, 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. Robot Control

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 (7) or (13). We obtained a hypersurface for each type of obstacle, and apply this hypersurface to the same type obstacles with unknown locations. We only present the case of irregular obstacles due to space limitation. With tight control bounds, the CBF-based QP can be easily infeasible, as shown in Fig. 3.



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

Fig. 3. The control profiles for the infeasible example with relaxation on both control limitations. The control bound for $u_2(t)$ is satisfied.

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 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:

$$k(\boldsymbol{y}, \boldsymbol{z}) = (k_1 + k_2 \boldsymbol{y}^T \boldsymbol{z})^2. \tag{14}$$

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

Now, 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. 4. 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 3rd training iteration is shown in Fig. 4.

TABLE I

Training results for the irregular obstacle

iter.	QP (13) 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[1]	0.0004			100k
gen.[2]	0	1.0000		100k

[1] denotes testing results within the training data sampling space.
[2] denotes testing results out of the training data sampling space.

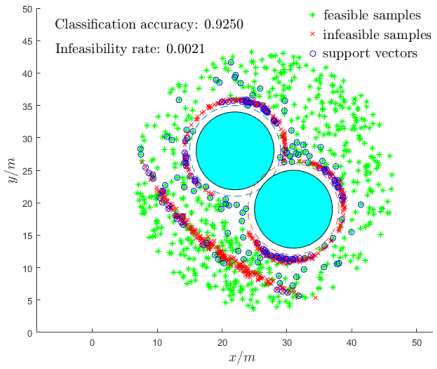
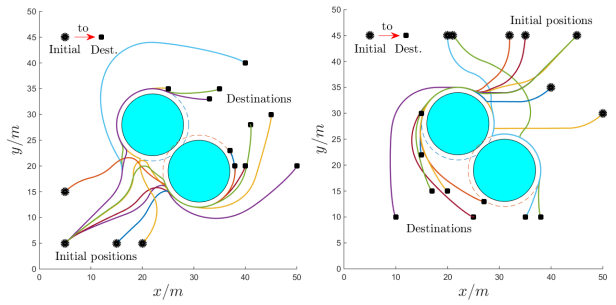


Fig. 4. Feedback training at the 3rd iteration for the irregular obstacle. All data are sampled around the obstacle (solid circle in all sub-figures). Each sample is a four dimensional point, but is visualized in $x - y$ plane.

As shown in Table I, the classification accuracy and the infeasibility rate change similarly to the regular obstacle case in each iteration. The training results in this case are shown in Table I. We also apply this hypersurface to the robot control problem, and test the feasibility and reachability. The QPs (13) 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. 5(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.



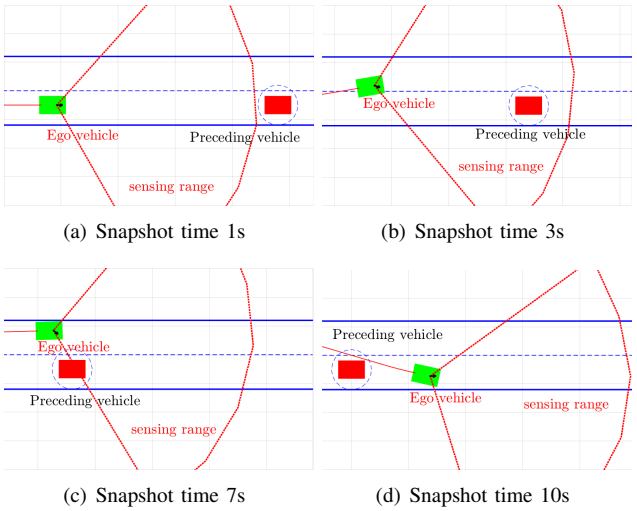
(a) Trajectories case 1 (b) Trajectories case 2

Fig. 5. Robot control problem feasibility and reachability test after learning in irregular obstacle case.

B. Autonomous Driving

In autonomous driving, the ego vehicle treats all the other vehicles as moving obstacles. We consider the same dynamics and constraints as in the last subsection, except relaxing the maximum speed limit to $28m/s$. We use the sampling-based 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 (14). The relative speed difference is sampled between $0m/s$ and $20m/s$. The feedback learning process is similar to Table I, but takes 6 iterations in order to achieve an infeasible rate that is smaller than ε. In the 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. Without the 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. 6.



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

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

Fig. 6. Sampling learning approach applied in autonomous driving for the overtaking of another moving vehicle.

VI. Conclusion

We proposed a machine learning technique for the CBF-based QPs corresponding to optimal control problems with safety constraints and control limitations. The learning approach can deal with both regular and irregular unsafe sets. The simulation 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.

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] J. P. Aubin, *Viability theory*. Springer, 2009.

[7] R. Wisniewski and C. Sloth, "Converse barrier certificate theorem," in *Proc. of 52nd IEEE Conference on Decision and Control*, Florence, Italy, 2013, pp. 4713–4718.

[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] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.