2020-12-14

Feasibility-guided learning for constrained optimal control problems

Feasibility-Guided Learning for Robust Control in Constrained Optimal Control Problems

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

Abstract— **Optimal control problems with constraints ensuring safety and convergence to desired states can be mapped onto a sequence of real time optimization problems through the use of Control Barrier Functions (CBFs) and Control Lyapunov Functions (CLFs). One of the main challenges in these approaches is ensuring the feasibility of the resulting quadratic programs (QPs) if the system is affine in controls. The recently proposed penalty method has the potential to improve the existence of feasible solutions to such problems. In this paper, we further improve the feasibility robustness (i.e., feasibility maintenance in the presence of time-varying and unknown unsafe sets) through the definition of a High Order CBF (HOCBF) that works for arbitrary relative degree constraints; this is achieved by a proposed feasibility-guided learning approach. Specifically, we apply machine learning techniques to classify the parameter space of a HOCBF into feasible and infeasible sets, and get a differentiable classifier that is then added to the learning process. The proposed feasibility-guided learning approach is compared with the gradient-descent method on a robot control problem. The simulation results show an improved ability of the feasibility-guided learning approach over the gradient-decent method to determine the optimal parameters in the definition of a HOCBF for the feasibility robustness, as well as show the potential of the CBF method for robot safe navigation in an unknown environment.**

I. Introduction

Stabilizing a dynamical system while optimizing a cost function and satisfying safety constraints is a fundamental and challenging problem in control. Typically, such problems include autonomous driving in road traffic and robot safe exploration in unknown environments. When safety becomes critical, it is desired to prioritize the strict satisfaction of constraints instead of optimality. The barrier function method [2], [12], [9], [21] has been proposed as an approach to this problem.

Barrier functions (BFs) are Lyapunov-like functions [19], whose use can be traced back to optimization problems [4]. More recently, they have been employed in verification and control, e.g., to prove set invariance [3], [16], [20] and for multi-objective control [14]. Control BFs (CBFs) are extensions of BFs for control systems. Recently, it has been shown that CBFs can be combined with control Lyapunov functions (CLFs) [17], [6], [1] as constraints to form quadratic programs (QPs) [7] for nonlinear control systems that are affine in controls, and these QPs can be solved in real time. While computationally efficient, the CBF and CLF-based QPs can easily become infeasible in the presence of both stringent safety constraints and tight control limitations, especially for high relative degree systems in a highly dynamical environment.

The CLF constraints are usually relaxed [2] such that they do not conflict with the CBF constraints in the QPs. Recent work showed that rich specifications given in signal temporal logic [9] and linear temporal logic [13], [18] can be translated to constraints and implemented by the CBF method with good solution feasibility if the constraints are with relative degree one. Several approaches to improve feasibility for the CBF and CLF-based QPs on specific applications have been proposed. For the adaptive cruise control (ACC) problem (the system is with relative degree 2) defined in [2], the infeasibility issue is addressed by including the minimum braking distance in the safety constraint. An approximation of the braking distance was used in [22] for a cooperative optimization control problem with non-linear dynamics. In both cases, an additional complex safety constraint needs to be added. Further, this approach does not scale well for high-dimensional systems. We recently developed the penalty method [21], which can improve the feasibility of the QPs by penalizing the class $\mathcal{K}$ functions in the definition of a High Order CBF (HOCBF) for an arbitrary relative degree constraint.

The use of machine learning techniques to improve feasibility was recently proposed for legged robots. Feasibility constraints for probabilistic models are learned in [5] based on simplified models. Since the learned constraints are complex, they are simplified by expectation-maximization (EM). Robot footstep limits are modeled as hyper-planes based on success and failure datasets in [15]. Reinforcement learning (RL) [11] [10] has the potential to address the infeasibility issue for optimal control problems, but it is difficult to quantify feasibility as a reward and the optimized parameters may also go to a local infeasible region where a feasible solution could never be found.

In this paper, we adopt the CBF method to improve the feasibility and feasibility robustness of optimal control problems with stringent safety constraints (usually with high relative degree) and tight control limitations in an unknown environment. The feasibility robustness is defined by the QP feasibility maintenance in the presence of a number of time-varying and unknown unsafe sets. Based on our proposed penalty method from [21], we parameterize a HOCBF, and use the parameters to improve the feasibility of the CBF

This work was supported in part by the NSF under grants IIS-1723995, CPS-1446151, ECCS-1931600, DMS-1664644, and CNS-1645681, by ARPA-Es NEXTCAR program under grant de-ar0000796, by AFOSR under grant FA9550-19-1-0158, and by the MathWorks. The authors are with the Division of Systems Engineering and Center for Information and Systems Engineering, Boston University, Brookline, MA, 02446, USA {xiaowei, cbelta, cgc}@bu.edu

and CLF-based QPs. Since trajectories of a system may be required to avoid a number of unsafe sets at the same time, we propose the idea of minimizing the value of a HOCBF (usually the distance to an unsafe set) when the corresponding HOCBF constraint first becomes active. In other words, we want the HOCBF constraint to become active as late as possible in the QPs. In this way, the feasibility robustness of the controller with respect to unknown unsafe sets is maximized. The main benefits of maximizing the robustness lie in the fact that the QP feasiblity can be maintained when the unsafe sets are unknown and with detection noise, as will be shown later. Another contribution of this paper 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 is maximized. We compare the proposed feasibility-guided method with the gradient-descent method with results showing improved controller robustness in a robot control problem.

This paper is structured as follows. In Sec. II, we give preliminaries on HOCBFs and CLFs. In Sec. III, we formulate an optimal control problem with safety constraints and control limitations. The framework of learning the optimal penalties and powers for a specific type of unsafe set is given in Sec. IV. We provide simulations and comparisons in Sec. V, and conclude with final remarks and directions for future work in Sec. VI.

II. Preliminaries

Definition 1: (*Class $\mathcal{K}$ function* [8]) 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 $\alpha(0) = 0$.

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 globally Lipschitz, and $\boldsymbol{u} \in U \subset \mathbb{R}^q$ (U denotes the control constraint set). Solutions $\boldsymbol{x}(t)$ of (1), starting at $\boldsymbol{x}(0)$ (we set the initial time to 0 without loss of generality), $t \geq 0$, are forward complete.

Definition 2: A set $C \subset \mathbb{R}^n$ is forward invariant for system (1) if its solutions starting at any $\boldsymbol{x}(0) \in C$ satisfy $\boldsymbol{x}(t) \in C$ for $\forall t \geq 0$.

Definition 3: (*Relative degree*) The relative degree of a continuously differentiable function $b : \mathbb{R}^n \to \mathbb{R}$ with respect to system (1) is the number of times we need to differentiate it along its dynamics until the control $\boldsymbol{u}$ explicitly shows in the corresponding derivative.

In this paper, since function b is used to define a constraint $b(\boldsymbol{x}) \geq 0$, we will also refer to the relative degree of b as the relative degree of 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 functions $\psi_1 : \mathbb{R}^n \to \mathbb{R}, \psi_2 : \mathbb{R}^n \to \mathbb{R}, \dots, \psi_m : \mathbb{R}^n \to \mathbb{R}$:

$$\psi_i(\boldsymbol{x}) := \dot{\psi}_{i-1}(\boldsymbol{x}) + \alpha_i(\psi_{i-1}(\boldsymbol{x})), i \in \{1, 2, \dots, m\}, \tag{2}$$

where $\alpha_i(\cdot), i \in \{1, 2, \dots, m\}$ denotes a differentiable class $\mathcal{K}$ function.

We further define a sequence of sets $C_1, C_2, \dots, C_m$ associated with (2) in the form:

$$C_i := \{\boldsymbol{x} \in \mathbb{R}^n : \psi_{i-1}(\boldsymbol{x}) \geq 0\}, i \in \{1, 2, \dots, m\}. \tag{3}$$

Definition 4: (*High Order Control Barrier Function (HOCBF)* [21]) Let $C_1, C_2, \dots, C_m$ be defined by (3) and $\psi_1(\boldsymbol{x}), \psi_2(\boldsymbol{x}), \dots, \psi_m(\boldsymbol{x})$ be defined by (2). 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 differentiable class $\mathcal{K}$ functions $\alpha_1, \alpha_2, \dots, \alpha_m$ such that

$$L_f^m b(\boldsymbol{x}) + L_g L_f^{m-1} b(\boldsymbol{x})\boldsymbol{u} + O(b(\boldsymbol{x})) + \alpha_m(\psi_{m-1}(\boldsymbol{x})) \geq 0, \tag{4}$$

for all $\boldsymbol{x} \in C_1 \cap C_2 \cap, \dots, \cap C_m$. In (4), L_f, L_g denote Lie derivatives along f and g, respectively, $O(\cdot)$ denotes the remaining Lie derivatives along f with degree less than or equal to $m - 1$ (omitted for simplicity, see [21]).

Given a HOCBF b, we define the set of all control values that satisfy (4) as:

$$\begin{aligned} K_{cbf} = \{\boldsymbol{u} \in U : L_f^m b(\boldsymbol{x}) + L_g L_f^{m-1} b(\boldsymbol{x})\boldsymbol{u} \\ + O(b(\boldsymbol{x})) + \alpha_m(\psi_{m-1}(\boldsymbol{x})) \geq 0\} \end{aligned} \tag{5}$$

Theorem 1: ([21]) Given a HOCBF $b(\boldsymbol{x})$ from Def. 4 with the associated sets $C_1, C_2, \dots, C_m$ defined by (3), if $\boldsymbol{x}(0) \in C_1 \cap C_2 \cap, \dots, \cap C_m$, then any Lipschitz continuous controller $\boldsymbol{u}(t) \in K_{cbf}, \forall t \geq 0$ renders $C_1 \cap C_2 \cap, \dots, \cap C_m$ forward invariant for system (1).

Definition 5: (*Control Lyapunov function (CLF)* [1]) A continuously differentiable function $V : \mathbb{R}^n \to \mathbb{R}$ is a globally and exponentially stabilizing control Lyapunov function (CLF) for system (1) if there exist constants $c_1 > 0, c_2 > 0, c_3 > 0$ such that

$$c_1||\boldsymbol{x}||^2 \leq V(\boldsymbol{x}) \leq c_2||\boldsymbol{x}||^2 \tag{6}$$

$$\inf_{u\in U}[L_f V(\boldsymbol{x}) + L_g V(\boldsymbol{x})\boldsymbol{u} + c_3 V(\boldsymbol{x})] \leq 0. \tag{7}$$

for $\forall \boldsymbol{x} \in \mathbb{R}^n$.

Theorem 2: ([1]) Given an exponentially stabilizing CLF V as in Def. 5, any Lipschitz continuous controller $\boldsymbol{u} \in K_{clf}(\boldsymbol{x})$, with

$$K_{clf}(\boldsymbol{x}) := \{\boldsymbol{u} \in U : L_f V(\boldsymbol{x}) + L_g V(\boldsymbol{x})\boldsymbol{u} + c_3 V(\boldsymbol{x}) \leq 0\},$$

exponentially stabilizes system (1) to its zero dynamics (defined by the dynamics of the internal part if we transform the system to standard form and set the output to zero [8]). Note that (7) can be relaxed by adding a relaxation at its right-hand side [1].

Recent works [2], [9], [12] combine CBFs and CLFs with quadratic costs to form optimization problems. Time is discretized and an optimization problem with constraints given by CBFs and CLFs is solved at each time step. Note that these constraints are linear in control since the state is fixed at the value at the beginning of the interval, and therefore the optimization problem is a quadratic program

(QP). The optimal control obtained by solving the QP is applied at the current time step and held constant for the whole interval. The dynamics (1) are updated, and the procedure is repeated. This method works conditioned on the fact that the QP is always feasible. We will show how we can further improve the QP feasibility by maximizing the feasibility robustness (will be formally defined in the next section) of the controller with respect to unknown unsafe sets in this paper.

III. Problem Formulation

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

$$\min_{\boldsymbol{u}(t)} J(\boldsymbol{u}(t)) = \int_0^{t_f} \mathcal{C}(||\boldsymbol{u}(t)||)dt, \tag{8}$$

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

State convergence: We want the state of system (1) to converge to a point $\boldsymbol{K} \in \mathbb{R}^n$, i.e.,

$$||\boldsymbol{x}(t) - \boldsymbol{K}|| \leq \xi, \forall t \in [t', t_f], \tag{9}$$

where $\xi > 0$ is arbitrarily small and $t' \in [0, t_f]$.

Constraint 1 (Unsafe Sets): Let S_o denote a set of unsafe sets. System (1) should always avoid all unsafe regions (obstacles) $j \in S_o$, i.e.,

$$h_j(\boldsymbol{x}(t)) \geq 0, \forall t \in [0, t_f]. \tag{10}$$

where $h_j : \mathbb{R}^n \to \mathbb{R}, \forall j \in S_o$ is continuously differentiable.

A HOCBF constraint for (10) becomes active when a control $\boldsymbol{u}$ makes inquality (4) become an equality for $b = h_j$.

Feasibility robustness: The feasibility robustness of a controller with respect to a constraint (10) can be quantified by the value of $h_j(\boldsymbol{x}(t_a))$ (the value of $h_j(\cdot)$ usually denotes the distance to the unsafe set $j \in S_o$) when the HOCBF constraint (4) for (10) first becomes active (and active afterwards) at $t_a \in [0, t_f]$. In order to maximize the feasibility robustness, we need to minimize

$$\min_{t_a} h_j(\boldsymbol{x}(t_a)), j \in S_o. \tag{11}$$

As an example, consider the adaptive cruise control problem [21]. The distance $z(t)$ between the controlled vehicle and the vehicle in front (both vehicles have double integrator dynamics and control constraints) should be greater than a constant $\delta > 0$, i.e., $z(t) \geq \delta, \forall t \geq 0$. Then we can define a HOCBF $h(\boldsymbol{x}) := z(t) - \delta$ ($m = 2$ in Def. 4 since the relative degree of $h(\cdot)$ is 2 for double integrator dynamics) for this safety constraint, and any control input should satisfy the HOCBF constraint (4). If the HOCBF constraint is first active at t_a and the value of $h(\boldsymbol{x}(t_a))$ is very small (while the control constraints should be always satisfied), i.e., the distance between these two vehicles is small, then the controller (from the QPs) is less constrained (before the HOCBF constraint becomes active) and robust to perturbations (such as noise). Thus, the feasibility robustness of the controller is improved and we wish to solve $\min_{t_a} h(\boldsymbol{x}(t_a))$.

Remark 1: There are three main advantages of maximizing the feasibility robustness of the controller. (i) The QPs are more likely to become feasible since fewer (HOCBF) 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 the change of environment and the detection of unknown unsafe sets since the corresponding HOCBF constraints only work (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 will fail to avoid these unsafe sets (i.e., QPs become infeasible). (iii) There is higher probability to find a better solution (e.g., energy optimal) if the feasibility robustness is maximized since the QP solutions are less constrained.

Constraint 2 (State Limitations): Assume we have a set of constraints on the state of system (1) in the form:

$$\boldsymbol{x}_{min} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{max}, \forall t \in [0, t_f], \tag{12}$$

where $\boldsymbol{x}_{max} := (x_{max,1}, x_{max,2}, \ldots, x_{max,n}) \in \mathbb{R}^n$ and $\boldsymbol{x}_{min} := (x_{min,1}, x_{min,2}, \ldots, x_{min,n}) \in \mathbb{R}^n$ denote the maximum and minimum state vectors, respectively, and the inequality is interpreted componentwise.

Constraint 3 (Control limitations): Assume we have a set of constraints on control inputs of system (1) in the form:

$$\boldsymbol{u}_{min} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{max}, \forall t \in [0, t_f]. \tag{13}$$

where $\boldsymbol{u}_{min} \in \mathbb{R}^q$ and $\boldsymbol{u}_{max} \in \mathbb{R}^q$ denote the minimum and maximum control input vectors, respectively (i.e., the constraint set U in (1) is rectangular).

A control policy for system (1) is $\boldsymbol{feasible}$ if constraints (10), (12) and (13) are satisfied. In this paper, we consider the following problem:

Problem: Find a feasible control policy for system (1) such that cost (8) is minimized, robustness is maximized (i.e., (11) is minimized), constraints 1, 2, 3 ((10), (12) and (13)) are strictly satisfied, and state convergence (9) is satisfied with the smallest possible ξ and t'.

Approach: The robustness objective (11) depends on the time t_a, while t_a is determined once a HOCBF in the above problem is defined. Therefore, we need to consider objective (11) in the definition of a HOCBF. We break the above problem into two sub-problems: (i) objective (8) subject to (10), (12), (13) and (9) that is solved with the QP-based method introduced at the end of Sec. II; (ii) objective (11) after solving sub-problem (i) $\forall t \in [0, t_f]$.

IV. Learning to Increase Feasibility Robustness

In this section, we introduce how to learn the optimal parameters in the definition of a HOCBF such that the feasibility robustness of the controller with respect to unknown unsafe sets is maximized, i.e., how to reformulate sub-problem (i), (ii) introduced at the end of the last section. We define unsafe sets as being of the same "type" if they have the same geometry. For example, circular unsafe sets are the same type if they have the same radius but different locations. Let $S_t \subseteq S_o$ denote the index set of all the unsafe set types in S_o.

A. HOCBF and CLF-based QP (sub-problem (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 at $t = \omega\Delta t$, we solve

$$\min_{\boldsymbol{u}(t), \delta(t)} \mathcal{C}(||\boldsymbol{u}(t)||) + p\delta^2(t) \tag{14}$$

subject to (13), the CLF constraint (7) for (9) (by defining a CLF for (9) such that a CLF constraint similar to (7 is satisified) and the HOCBF constraints (4) corresponding to (10) and (12), where $p > 0$ is a penalty on the relaxation $\delta(t)$ ($\delta(t)$ is a relaxation variable that replaces 0 on the right-hand side of (7)). Since the state is kept constant at its value at the beginning of the interval, the above optimization problem is a QP, which can easily become infeasible. In the rest of the paper, we show how we can use machine learning techniques in finding the optimal parameters in the definition of a HOCBF such that the feasibility robustness is maximized.

B. The Penalty Method

To improve the feasibility [21] of the problem (14), we add penalties on the class $\mathcal{K}$ functions $\alpha_1(\cdot), \alpha_2(\cdot), \dots, \alpha_m(\cdot)$ in (2) in the definition of a HOCBF $b(\boldsymbol{x})$. In the set of class $\mathcal{K}$ functions that consist of power functions, we explictly rewrite (2) as

$$\begin{aligned} \psi_0(\boldsymbol{x}) :=& b(\boldsymbol{x}) \\ \psi_1(\boldsymbol{x}) :=& \dot{\psi}_0(\boldsymbol{x}) + p_1\psi_0^{q_1}(\boldsymbol{x}), \\ &\vdots \\ \psi_m(\boldsymbol{x}) :=& \dot{\psi}_{m-1}(\boldsymbol{x}) + p_m\psi_{m-1}^{q_m}(\boldsymbol{x}), \end{aligned} \tag{15}$$

where $p_1 > 0, p_2 > 0, \dots, p_m > 0$ and $q_1 > 0, q_2 > 0, \dots, q_m > 0$.

For each type of unsafe set $j \in S_t$, we consider an arbitrary location for it and get an unsafe set constraint $h_j(\boldsymbol{x}(t)) \geq 0$ (similar to (10)). Let $\boldsymbol{p} := (p_1, p_2, \dots, p_m)$, $\boldsymbol{q} := (q_1, q_2, \dots, q_m)$. We know from [21] that the values of $q_1, q_2, \dots, q_m$ affect the feasibility region of (14), as well as what time the HOCBF constraint (4) will be active, i.e., we can rewrite $h_j(\boldsymbol{x}(t_a))$ as $h_j(\boldsymbol{x}(t_a), \boldsymbol{p}, \boldsymbol{q})$. Let $\mathcal{D}_j(\boldsymbol{p}, \boldsymbol{q}) := h_j(\boldsymbol{x}(t_a), \boldsymbol{p}, \boldsymbol{q})$ (since $h_j(\boldsymbol{x}(t_a), \boldsymbol{p}, \boldsymbol{q})$ is fixed once $\boldsymbol{p}, \boldsymbol{q}$ are given, $h_j(\cdot)$ does not actually depend on $\boldsymbol{x}(t_a)$). We reformulate (11) as

$$\min_{\boldsymbol{p}, \boldsymbol{q}} \mathcal{D}_j(\boldsymbol{p}, \boldsymbol{q}), j \in S_t. \tag{16}$$

We can view the minimization of $D_j(\boldsymbol{p}, \boldsymbol{q})$ as the maximization of the feasibility robustness that depends on $\boldsymbol{p}, \boldsymbol{q}$.

Then, we need to find the optimal $\boldsymbol{p}$ and $\boldsymbol{q}$ that minimize (16) for each unsafe set type $j \in S_t$. However, this optimization problem is hard to solve. We will introduce an approach using machine learning techniques in the following section.

C. Feasibility-Guided Optimization (sub-problem (ii))

The optimization problem (16) is a typical problem that can be solved with reinforcement learning approaches. However, most of the $\boldsymbol{p}, \boldsymbol{q}$ values result in infeasible solutions of problem (14), which makes (16) 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, and for each set of $\boldsymbol{p}, \boldsymbol{q}$ values, we solve problem (14) until the state convergence (9) is achieved. If problem (14) (the QPs) is feasible at all times, then we label this set of $\boldsymbol{p}, \boldsymbol{q}$ values (as a whole) as $+1$, otherwise, we label it as -1. Eventually, we get sets of feasible and infeasible $\boldsymbol{p}, \boldsymbol{q}$ points. Then we can apply a machine learning technique (such as support vector machine, deep neural network etc.) to classify these two sets and get a continuously differentiable hypersurface:

$$\mathfrak{H}_j : \mathbb{R}^{2m} \to \mathbb{R}, \tag{17}$$

where

$$\mathfrak{H}_j(\boldsymbol{p}, \boldsymbol{q}) \geq 0 \tag{18}$$

denotes the set of $\boldsymbol{p}, \boldsymbol{q}$ values (as a whole) which leads to the feasible solution of QPs (14), i.e., the feasibility constraint for the set of $\boldsymbol{p}, \boldsymbol{q}$ values associated with the QPs (14). We can use a HOCBF to enforce (18) if $\boldsymbol{p}, \boldsymbol{q}$ are state variables of a system, which motivates us to define dynamics for $\boldsymbol{p}, \boldsymbol{q}$, as shown later.

Based on the feasiblity classification hypersurface (17), we look further to optimize (16), i.e., we consider (16) subject to (18). However, the learned hypersurface (17) is generally complex, and thus makes this optimization problem very hard to solve. We use the following approach 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 (dynamics) for $\boldsymbol{p}, \boldsymbol{q}$ (as the variations of $\boldsymbol{p}, \boldsymbol{q}$ that are controlled), i.e., we have

$$(\dot{\boldsymbol{p}}(\mathfrak{t}), \dot{\boldsymbol{q}}(\mathfrak{t})) = \boldsymbol{\nu}(\mathfrak{t}), \dot{\boldsymbol{p}}(\mathfrak{t}_0) = \boldsymbol{p}_0, \dot{\boldsymbol{q}}(\mathfrak{t}_0) = \boldsymbol{q}_0, \tag{19}$$

where $\boldsymbol{\nu} \in \mathbb{R}^{2m}$ denotes a controllable 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 (16), which is different and independent from t in the system (1) and problem (14). $\mathfrak{t}_0 \in \mathbb{R}$ denotes the initial time.

Considering feasibility of the problem (14), the dynamic process (determined by $\boldsymbol{\nu}$) should be subjected to (19) and (18). Since we want to find the control $\boldsymbol{\nu}$ such that the resulting $\boldsymbol{p}, \boldsymbol{q}$ (determined by $\boldsymbol{\nu}$) always lead to the feasible solution of QPs (14) with the CBF method, i.e., we need to take the derivative of (17), we minimize the derivative of (16) (the fastest decreasing direction of the value of $\mathcal{D}_j(\boldsymbol{p}, \boldsymbol{q})$ in (16)) in the dynamic process to make $\boldsymbol{\nu}$ also show up in the cost function. As long as the derivative of (16) is negative, we make sure that (16) is decreasing in each time step (by discretizing $\mathfrak{t}$ similar to sub-problem (i)).

By taking the derivative of (16) with respect to $\mathfrak{t}$, we have

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

The relative degree of the feasibility constraint (18) with respect to (19) is 1. We then use a HOCBF with $m = 1$ (as in Def. 4) to enforce (18) and find a control $\boldsymbol{\nu}$ that can satisfy (18) in the dynamic process:

$$\frac{d\mathfrak{H}_j(\boldsymbol{p},\boldsymbol{q})}{d(\boldsymbol{p},\boldsymbol{q})}\boldsymbol{\nu} + \alpha_1(\mathfrak{H}_j(\boldsymbol{p},\boldsymbol{q})) \geq 0, \tag{21}$$

where $\alpha_1(\cdot)$ is a class $\mathcal{K}$ function as in Def. 4 (the definition of $\psi_1(\cdot)$ in (2)). Any control input $\boldsymbol{\nu}$ that satisfies (21) implies that the resulting $\boldsymbol{p}, \boldsymbol{q}$ (determined by $\boldsymbol{\nu}$) lead to a feasible solution of QPs (14) in the dynamic process.

Then, we reformulate sub-problem (ii) by the dynamic process (**feasibility-guided optimization** (FGO)). We use the approach introduced as in Sec. IV-A to solve the dynamic process, i.e., we discretize $\mathfrak{t}$, at each $\mathfrak{t} = \omega\Delta\mathfrak{t}, \omega \in \{0, 1, \dots\}$ ($\Delta\mathfrak{t} > 0$ denotes the discretization constant), and we solve

$$\min_{\boldsymbol{\nu}(\mathfrak{t})} \frac{d\mathcal{D}_j(\boldsymbol{p}(\mathfrak{t}),\boldsymbol{q}(\mathfrak{t}))}{d(\boldsymbol{p}(\mathfrak{t}),\boldsymbol{q}(\mathfrak{t}))}\boldsymbol{\nu}(\mathfrak{t}) \tag{22}$$

subject to (21), (19). Then update (19) for $\mathfrak{t} \in (\omega\Delta\mathfrak{t}, (\omega + 1)\Delta\mathfrak{t})$ with $\boldsymbol{\nu}^*(\mathfrak{t})$. 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 vector of dimension $1 \times 2m$, while $\boldsymbol{\nu}$ is a vector of dimension $2m \times 1$. Therefore, the cost function in the last equation is a scalar function of $\boldsymbol{\nu}$.

The optimization problem (22) is a linear program (LP) (to determine $\boldsymbol{\nu}$) 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 (22) is ill-posed because it leads to unbounded solutions. In fact, the value of $\boldsymbol{\nu}$ determines the search step length of the dynamic process, and we want to limit this step length. Otherwise, the solution of the LP at each step is infinity (i.e., the dynamic process search step length is infinity, and fails to work). Therefore, we add limitations to $\boldsymbol{\nu}$ for the LP (22):

$$\boldsymbol{\nu}_{min} \leq \boldsymbol{\nu} \leq \boldsymbol{\nu}_{max}. \tag{23}$$

where $\boldsymbol{\nu}_{min} < \mathbf{0}, \boldsymbol{\nu}_{max} > \mathbf{0}$ (interpreted componentwise), $\mathbf{0} \in \mathbb{R}^{2m}$.

After adding (23) to (22), the dynamic process search step length will become bounded. Although there are control limitations on $\boldsymbol{\nu}$, the resulting LP from the optimization (22) is always feasible since the relative degree of (18) with respect to (19) is 1.

Note that in (22), we have

$$\frac{d\mathcal{D}_j}{d(\boldsymbol{p},\boldsymbol{q})} = (\frac{\partial\mathcal{D}_j}{\partial p_1}, \dots, \frac{\partial\mathcal{D}_j}{\partial p_m}, \frac{\partial\mathcal{D}_j}{\partial q_1}, \dots, \frac{\partial\mathcal{D}_j}{\partial q_m})$$

and 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 (22)).

We present the FGO algorithm in Alg. 1. For each step of the FGO algorithm from Alg. 1, the following four conditions may terminate the algorithm: (i) the problem (14) becomes infeasible (since the hypersurface (17) from the machine learning techniques cannot ensure 100% classification accuracy), (ii) the evaluated values of $\frac{\partial\mathcal{D}_j}{\partial p_1}, \dots, \frac{\partial\mathcal{D}_j}{\partial p_m}, \frac{\partial\mathcal{D}_j}{\partial q_1}, \dots, \frac{\partial\mathcal{D}_j}{\partial q_m}$ are all 0, (iii) the objective function value of (16) is greater than the current known minimum value. (iv) the iteration time exceeds some $N \in \mathbb{N}$.

If we consider Alg. 1 without the constraint (21), then we have the commonly used gradient descent (GD) algorithm. The FGO algorithm is more conservative compared with GD since the solution searching path is guided by the feasibility of (14). We can apply GD one step forward whenever the FGO algorithm terminates to alleviate this limitation, which is shown in the last part of Alg. 1.

Algorithm 1: FGO algorithm

Input: Constraints (10) (9), system (1) with (13), N
Output: $\boldsymbol{p}^*, \boldsymbol{q}^*, \mathcal{D}_{min}$
Sample $\boldsymbol{p}, \boldsymbol{q}$ in the definition of the HOCBF;
Discard samples that do not meet the initial conditions of HOCBF constraint (4);
Solve (14) for each sample for $t \in [0, t_f]$ and label all samples;
Use machine learning to find classifier (17);
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 $\frac{\partial\mathcal{D}_j}{\partial p_1}, \dots, \frac{\partial\mathcal{D}_j}{\partial p_m}, \frac{\partial\mathcal{D}_j}{\partial q_1}, \dots, \frac{\partial\mathcal{D}_j}{\partial q_m}$ at $\boldsymbol{p}_0, \boldsymbol{q}_0$;
  if $\frac{\partial\mathcal{D}_j}{\partial p_k}, \frac{\partial\mathcal{D}_j}{\partial q_k}, \forall k \in \{1, 2, \dots, m\}$ *is infeasible* **then**
    Jump to the very begining of the loop;
  else
    $\frac{\partial\mathcal{D}_j}{\partial p_k} = 0, \frac{\partial\mathcal{D}_j}{\partial q_k} = 0$ if $\exists k \in \{1, 2, \dots, m\}$ such that $\frac{\partial\mathcal{D}_j}{\partial p_k}, \frac{\partial\mathcal{D}_j}{\partial q_k}$ is infeasible to evaluate;
  end
  Solve optimization (22) and get new $\boldsymbol{p}, \boldsymbol{q}$;
  Solve problem (14) with $\boldsymbol{p}, \boldsymbol{q}$;
  if *(14) is feasible* **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 optimization (22) without (21) and get new $\boldsymbol{p}, \boldsymbol{q}$;
    Solve problem (14) 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
$\boldsymbol{p}^* = \boldsymbol{p}_0, \boldsymbol{q}^* = \boldsymbol{q}_0$;

D. Feasibility Generalization

The feasibility and feasibility robustness of the controller for problem (14) is sensitive to the "shape" of the unsafe sets, but not to the "location". For example, the location of a circular obstacle does not affect the feasibility and feasibility robustness of the controller for a robot, but the geometry of this circular obstacle does. In this case, we do not need to know the exact location of the obstacle. If we have learned feasibility for a specific location obstacle and get optimal $\boldsymbol{p}^*, \boldsymbol{q}^*$ with the FGO algorithm, then the optimal $\boldsymbol{p}^*, \boldsymbol{q}^*$ apply to other located obstacles of the same geometry.

In the case that we know the type of unsafe sets but not the locations, we can learn feasibility and robustness for each type of unsafe set given an arbitrary location with the FGO algorithm. Since the initial system condition may also affect the feasibility of problem (14), we may learn the optimal $\boldsymbol{p}^*, \boldsymbol{q}^*$ under the worst initial conditions (e.g., with maximum obstacle-approaching speed for a robot), and then these optimal $\boldsymbol{p}^*, \boldsymbol{q}^*$ may also apply to other initial conditions. For example, we may set the initial heading angle (as well as the target heading angle) of a robot so as to initially pass through the center of the circle obstacle and set the speed to its maximum speed. Once the optimal $\boldsymbol{p}^*, \boldsymbol{q}^*$ are found under this conditon, they may also be applied to other conditions.

In an unknown environment, system (1) may even not know the type of the unsafe sets, i.e., the formulation of (10). We can learn feasibility and robustness for some type-known unsafe sets with the FGO algorithm, and then use these unsafe sets to approximate any other types of unsafe sets.

V. Implementation and Case Studies

We implemented the FGO algorithm 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 $1m$ sensing uncertainty) to detect the obstacles.

The robot dynamics are defined in the form:

$$\underbrace{\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\\ \dot{v}\end{bmatrix}}_{\dot{\boldsymbol{x}}} = \underbrace{\begin{bmatrix}v\cos(\theta)\\ v\sin(\theta)\\ 0\\ 0\end{bmatrix}}_{f(\boldsymbol{x})} + \underbrace{\begin{bmatrix}0 & 0\\ 0 & 0\\ 1 & 0\\ 0 & 1\end{bmatrix}}_{g(\boldsymbol{x})} \underbrace{\begin{bmatrix}u_1\\ u_2\end{bmatrix}}_{\boldsymbol{u}} \tag{24}$$

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 consider cost (8) as the energy consumption in the form:

$$J(\boldsymbol{u}(t)) = \int_0^{t_f} \eta \frac{\max\{u_{2,min}^2, u_{2,max}^2\}}{\max\{u_{1,min}^2, u_{1,max}^2\}} u_1^2(t) + (1-\eta) u_2^2(t) dt \tag{25}$$

TABLE I
Simulation Parameters

Name	value	unit	Name	value	unit
p	1	unitless	r	7	m
ϵ	10	unitless	Δt	0.1	s
v_{min}	0	m/s	v_{max}	2	m/s
$u_{1,min}$	-0.2	rad/s	$u_{1,max}$	0.2	rad/s
$u_{2,min}$	-0.5	m/s^2	$u_{2,max}$	0.5	m/s^2

where $u_{1,min} < 0, u_{1,max} > 0$, $u_{2,min} < 0, u_{2,max} > 0$ denote the minimum and maximum turning control and acceleration control, respectively. $\eta \in [0, 1]$ denotes a weight factor which captures the tradeoff between the two components.

We also want the robot to arrive at a destination $(x_d, y_d) \in \mathbb{R}^2$, i.e., drive $(x(t), y(t))$ to $(x_d, y_d), \forall t \in [t', t_f], t' \in [0, t_f]$, as defined in (9). The dynamics (24) are not full state linearizable [8] 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.,

$$\theta(t) \to \arctan(\frac{y_d - y(t)}{x_d - x(t)}), \quad v(t) \to v_0, \forall t \in [0, t_f]. \tag{26}$$

Now, we can apply the CLF method to (26) (as introduced in Def. 5) since the relative degrees of the heading angle and speed are 1.

The unsafe sets (10) are defined as circular obstacles:

$$\sqrt{(x(t) - x_i)^2 + (y(t) - y_i)^2} \geq r, \forall t \in [0, t_f], \forall i \in S, \tag{27}$$

where (x_i, y_i) denotes the location of the obstacle i, and $r > 0$ denotes the safe distance to the obstacle.

The speed and control constraints (13) are defined as:

$$\begin{aligned} v_{min} \leq v(t) \leq v_{max}, \forall t \in [0, t_f], \\ u_{1,min} \leq u_1(t) \leq u_{1,max}, \forall t \in [0, t_f], \\ u_{2,min} \leq u_2(t) \leq u_{2,max}, \forall t \in [0, t_f]. \end{aligned} \tag{28}$$

$v_{min} \geq 0, v_{max} > 0$ denote the minimum and maximum speed, respectively. The simulation parameters are listed in Table I.

We set up the FGO algorithm training environment with the initial position of the robot, the location of the obstacle 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, $\Delta \mathfrak{t} = 0.1, \boldsymbol{\nu}_{max} = -\boldsymbol{\nu}_{min} = (0.1, 0.1, 0.1, 0.1)$. The map for FGO training is shown in Fig. 1.

Note that the value of ε in this example 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. 1 since the desired heading angle (26) in the CLF exactly passes through the origin of the obstacle. If $\varepsilon > 0$, the robot goes left around the obstacle as shown in Fig. 1. Otherwise, the robot turns right and then goes to the destination.

We choose a very small $\varepsilon \neq 0$ in our FGO algorithm. Since the obstacle constraint (27) is with relative degree 2

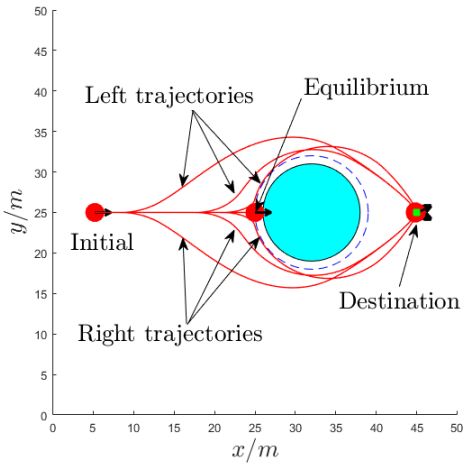


Fig. 1. FGO algorithm pre-training map with some feasible example trajectories.

with respect to system (24), we have $\boldsymbol{p} = (p_1, p_2), \boldsymbol{q} = (q_1, q_2)$. We randomly sample M (M is a positive integer) training and 1000 testing samples (around half of them are feasible for this robot path planning problem) for $\boldsymbol{p}$ and $\boldsymbol{q}$ over interval $(0, 3]$ and $(0, 2]$, respectively.

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

$$k(\boldsymbol{y}, \boldsymbol{z}) = (c_1 + c_2 \boldsymbol{y}^T \boldsymbol{z})^7. \tag{29}$$

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 II.

The FGO has better performance compared with GD in finding $\mathcal{D}_{min}$ when the number of training samples M for the hypersurface (21) is large enough, as shown in Table II. But this advantage decreases when the classification accuracy of the hyper surface (21) further increases, which may be due to over-fitting. One comparison example between FGO and GD search paths is shown in Fig. 2(a), 2(b). We can combine the FGO and GD algorithms, i.e., we choose the best result from the FGO and GD algorithms by implementing both of them. If we continue to apply the FGO algorithm to the good results from GD, the further improvement percentage is around 5% among all the testing samples.

We have also implemented the learned optimal penalties and powers ($(p_1^*, p_2^*, q_1^*, q_2^*) = (0.7426, 1.9745, 1.9148, 0.7024)$, not unique) in the definition of all the HOCBFs for all obstacles in a robot exploration problem in an unknown environment. All the circular obstacles (with different size to test the robustness of the penalty method with the learned optimal penalties and powers) move randomly. It is assumed that any obstacles that are detected by the robot will stop moving until the robot moves away. This is to ensure that the robot will not collide with obstacles passively (i.e., collisions happen due to the movement of obstacles). 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.

We also compared the CBF-based robot exploration framework with the RRT and A* algorithms by picking one frame from the last simulation and fixing the location of all the obstacles, as shown in Fig. 3. 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 may disappear 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 III. The computation time for the CBF method is only shown for the solution of one-step QPs in Table III since it does not need a receding horizon, while the computation time for the RRT and A* algorithms are for the path planning time. In a dynamic environment, the RRT and A* algorithms need to re-plan their path at each time step, which may take less time than the ones shown in Table III but is more demanding for these two algorithms. Therefore, we can see that the CBF-based framework is more adaptive in robot safe exploration.

VI. Conclusions

We improved the constrained optimal control problem feasibility by maximizing the feasibility robustness through the learning of optimal parameters in the definition of a high order control barrier function that works for arbitrary relative degree constraints. This is achieved by a feasibility-guided learning approach. The proposed feasibility-guided learning approach has shown an improved ability to determine the optimal parameters compared with the gradient-descent method. The implementation on a robot safe exploration problem has shown good potential and adaptivity of the proposed framework for planning with safety guarantees compared with other path planning algorithms. Future work will focus on how to deal with traps formed by obstacles, including environment and system noise.

References

[1] 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*, pages 6837–6842, 2012.

[2] A. D. Ames, J. W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *Proc. of 53rd IEEE Conference on Decision and Control*, pages 6271–6278, 2014.

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

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

[5] 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.

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

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

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

[9] L. Lindemann and D. V. Dimarogonas. Control barrier functions for signal temporal logic tasks. *IEEE Control Systems Letters*, 3(1):96–101, 2019.

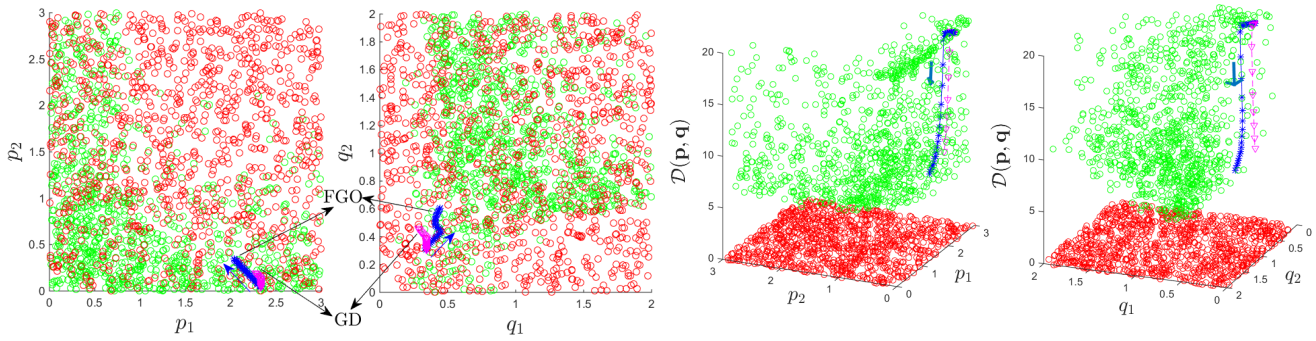
[10] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *preprint arXiv:1602.01783*, 2016.

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

TABLE II

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



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

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

Fig. 2. FGO and GD algorithm search implementation. The red circles denote infeasible points and the green circles denote feasible points for $\boldsymbol{p}, \boldsymbol{q}$ in the training samples.

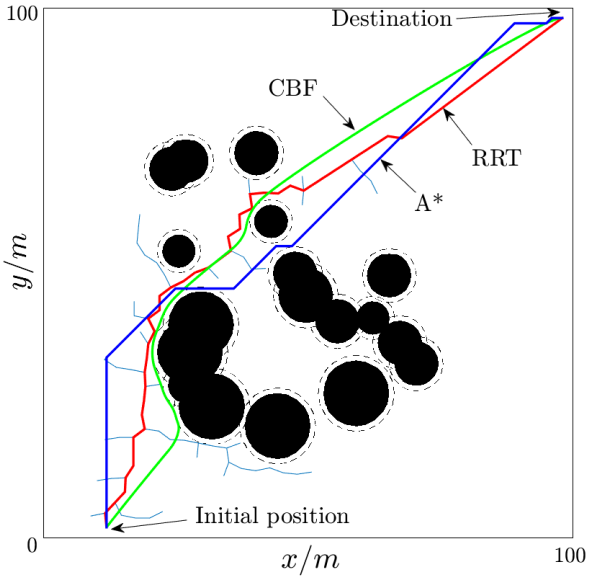


Fig. 3. Comparison of robot paths between CBF, A* and RRT.

TABLE III

PERFORMANCE COMPARISON BETWEEN CBF, A* AND RRT IN HIGHLY DYNAMIC UNKNOWN ENVIRONMENT

item	R.T. 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

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

[13] P. Nillson and A. D. Ames. Barrier functions: Bridging the gap between planning from specifications and safety-critical control. In *Proc. of 57th IEEE Conference on Decision and Control*, pages 765–772, Miami, 2018.

[14] D. Panagou, D. M. Stipanovic, and P. G. Voulgaris. Multi-objective control for multi-agent systems using lyapunov-like barrier functions. In *Proc. of 52nd IEEE Conference on Decision and Control*, pages 1478–1483, Florence, Italy, 2013.

[15] 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*, 28(2):427–439, 2012.

[16] S. Prajna, A. Jadbabaie, and G. J. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Transactions on Automatic Control*, 52(8):1415–1428, 2007.

[17] E. Sontag. A lyapunov-like stabilization of asymptotic controllability. *SIAM Journal of Control and Optimization*, 21(3):462–471, 1983.

[18] M. Srinivasan, S. Coogan, and M. Egerstedt. Feasibility envelopes for metric temporal logic specifications. In *Proc. of 57th IEEE Conference on Decision and Control*, pages 1991–1996, Miami Beach, FL, 2018.

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

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

[21] W. Xiao and C. Belta. Control barrier functions for systems with high relative degree. In *Proc. of 58th IEEE Conference on Decision and Control*, 2019. available in arXiv:1903.04706.

[22] W. Xiao, C. Belta, and C. G. Cassandras. Decentralized merging control in traffic networks: A control barrier function approach. In *Proc. ACM/IEEE International Conference on Cyber-Physical Systems*, pages 270–279, Montreal, Canada, 2019.