2020 59th IEEE Conference on Decision and Control (CDC)
Jeju Island, Republic of Korea, December 14-18, 2020

Feasibility-Guided Learning for Constrained Optimal Control Problems

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

Abstract— **Optimal control problems with constraints ensuring safety 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. In this paper, we 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); this is achieved by a proposed feasibility-guided learning approach using machine learning techniques. The effectiveness of the proposed feasibility-guided learning approach is demonstrated on a robot control problem.**

I. Introduction

Stabilizing a dynamical system while optimizing a cost and satisfying constraints is a fundamental and challenging problem in control theory. Typically, such problems include autonomous driving in traffic and robot safe exploration in unknown environments. When safety becomes critical, it is desired to prioritize the strict satisfaction of constraints over optimality. The barrier function method [1], [2], [3] has been proposed as an approach to this problem.

Barrier functions (BFs) are Lyapunov-like functions [4], whose use can be traced back to optimization problems [5]. More recently, they have been employed to prove set invariance [6], [7], [8], for multi-objective control [9], and in assisting Lyapunov functions [10]. 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) [11], [12], [13] as constraints to form quadratic programs (QPs) [1] 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 be infeasible in the presence of both safety constraints and tight control limitations, especially for high relative degree systems.

The CLF constraints are usually relaxed [13] such that they do not conflict with the CBF constraints in the QPs. Recent work showed that rich specifications given in signal temporal logic [2] and linear temporal logic [14] 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 two) defined in [1], the infeasibility issue is addressed by including the minimum braking distance in the safety constraint. In this case, an additional complex constraint needs to be added. Further, this approach does not scale well for high-dimensional systems. The penalty method [3] we recently developed can improve the feasibility of the QPs by penalizing the class $\mathcal{K}$ functions in the definition of a High Order CBF (HOCBF) [3] for an arbitrary relative degree constraint.

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. Feasibility robustness is defined in terms of maximally ensuring the feasibility of the CBF-associated QPs in the presence of time-varying and unknown unsafe sets. Based on our proposed penalty method from [3], we parameterize a HOCBF, and use the parameters to improve the feasibility of the CBF 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 a distance metric 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. 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. 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.

II. Preliminaries

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 $\boldsymbol{x}(0)$, $t \geq 0$, are forward complete.

Definition 1: 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 $\forall t \geq 0$.

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

The *relative degree* of a high order 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 $\psi_i : \mathbb{R}^n \to \mathbb{R}, i \in \{1, \dots, m\}$:

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

where $\alpha_i(\cdot), i \in \{1, \dots, m\}$ denote differentiable class $\mathcal{K}$ functions [15]. We further define a sequence of sets $C_1, \dots, C_m$ associated with (2) in the form:

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

Definition 2: (*High Order Control Barrier Function (HOCBF)* [3]) Let $C_1, \dots, C_m$ be defined by (3) and $\psi_1(\boldsymbol{x}), \dots, \psi_m(\boldsymbol{x})$ 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 $\alpha_1, \dots, \alpha_m$ such that

$$\sup_{\boldsymbol{u} \in U} [L_f^m b(\boldsymbol{x}) + L_g L_f^{m-1} b(\boldsymbol{x})\boldsymbol{u} + R(b(\boldsymbol{x})) + \alpha_m(\psi_{m-1}(\boldsymbol{x}))] \geq 0, \quad (4)$$

for all $\boldsymbol{x} \in C_1 \cap, \dots, \cap C_m$. In (4), L_f, L_g denote Lie derivatives along f and g, respectively, $R(\cdot)$ denotes the remaining Lie derivatives along f with degree less than or equal to $m-1$ (omitted for simplicity, see [3]).

Theorem 1: ([3]) Given a HOCBF $b(\boldsymbol{x})$ from Def. 2 with the associated sets $C_1, \dots, C_m$ defined by (3), if $\boldsymbol{x}(0) \in C_1 \cap, \dots, \cap C_m$, then any Lipschitz continuous controller $\boldsymbol{u}(t) \in U, \forall t \geq 0$ that satisfies (4) renders $C_1 \cap, \dots, \cap C_m$ forward invariant for system (1).

Definition 3: (*Control Lyapunov function (CLF)* [13]) 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, for $\forall \boldsymbol{x} \in \mathbb{R}^n$, $c_1||\boldsymbol{x}||^2 \leq V(\boldsymbol{x}) \leq c_2||\boldsymbol{x}||^2$ and

$$\inf_{u \in U} [L_f V(\boldsymbol{x}) + L_g V(\boldsymbol{x})\boldsymbol{u} + c_3 V(\boldsymbol{x})] \leq 0. \quad (5)$$

Note that (5) can be relaxed by replacing 0 with a relaxation variable $\delta \in \mathbb{R}$ at its right-hand side. However, this may not guarantee stability. Recent works [1], [2] 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. Therefore, the optimization problem is a sequence of quadratic programs (QPs).

III. Problem Formulation

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

$$\min_{\boldsymbol{u}(t)} \int_0^{t_f} \mathcal{C}(||\boldsymbol{u}(t)||)dt, \quad (6)$$

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

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

$$||\boldsymbol{x}(t) - \boldsymbol{K}|| \leq \xi, \forall t \in [t', t_f], \quad (7)$$

where $\xi > 0$ is a small number 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.,

$$b_j(\boldsymbol{x}(t)) \geq 0, \forall t \in [0, t_f]. \quad (8)$$

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

A HOCBF constraint for (8) becomes active when a control $\boldsymbol{u}$ makes (4) become an equality.

Feasibility robustness: The feasibility robustness of a controller with respect to a constraint (8) can be quantified by the value of $b_j(\boldsymbol{x}(t_a))$ when the HOCBF constraint (4) for (8) first becomes active at $t_a \in [0, t_f]$ and is active afterwards. The value of $b_j(\boldsymbol{x})$ usually denotes a distance metric to the unsafe set $j \in S_o$. In order to maximize the feasibility robustness, we need to minimize

$$\min_{t_a} b_j(\boldsymbol{x}(t_a)), j \in S_o. \quad (9)$$

Remark 1: There are three main advantages in maximizing the feasibility robustness of the controller: (i) The QPs are more likely to become 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 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 may fail to avoid these unsafe sets. (iii) There is higher probability to find a better solution (e.g., energy optimal) if the feasibility robustness is maximized since the QPs are less constrained. However, achieving (9) can increase the chance of collision in the presence of disturbances. This can be solved by relaxing the requirement in (9), i.e., by minimizing $||b_j(\boldsymbol{x}(t_a)) - d_0||$, for some $d_0 > 0$. This can also be solved by considering the noise bounds in (4) if the bounds are known [16].

Constraint 2 (Control limitations): Assume we have a set of constraints (componentwise) on control inputs of system (1) in the form:

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

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. The state constraints are similar to (10), and can be included in (8).

A control policy for system (1) is $\boldsymbol{feasible}$ if constraints (8) and (10) are satisfied. In this paper, we consider:

Problem: Find a feasible control policy for system (1) such that cost (6) is minimized, robustness is maximized (i.e., (9) is minimized), and state convergence (7) is satisfied with the smallest possible t'.

Approach: The robustness objective (9) depends on the time t_a, where t_a is determined once a HOCBF in the above problem is defined. Therefore, we need to consider objective (9) in the definition of a HOCBF. We decompose the above problem into two sub-problems: (i) objective (6) subject to (8), (10) and (7) that is solved with the QP-based method from [1]; (ii) objective (9) after solving sub-problem (i).

IV. Learning to Increase Feasibility Robustness

The learning objective is to maximize the feasibility robustness of the controller with respect to unknown unsafe sets. We define unsafe sets as being of the same "type" if they have the same geometry such that the feasibility of sub-problem (i) is the same, e.g., 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.

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 at $t = \omega\Delta t$, we solve

$$\min_{\boldsymbol{u}(t), \delta(t)} \mathcal{C}(||\boldsymbol{u}(t)||) + p_0\delta^2(t) \tag{11}$$

subject to (10), the CLF constraint (5) for (7) (by defining a CLF for (7) such that the CLF constraint is satisified) and the HOCBF constraints (4) corresponding to (8), where $p_0 > 0$ is a penalty on the relaxation $\delta(t) \in \mathbb{R}$, and $\delta(t)$ is a relaxation variable on the CLF constraint as discussed after Def. 3. The above optimization problem 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 a HOCBF such that the feasibility robustness is maximized; this is accomplished in the next subsection.

B. The Penalty Method

To improve the feasibility [3] of the problem (11), we add penalties on the class $\mathcal{K}$ functions $\alpha_1(\cdot), \alpha_2(\cdot), \dots, \alpha_m(\cdot)$, where m denotes the relative degree of the constraint $b(\boldsymbol{x}) \geq 0$ in the definition of a HOCBF $b(\boldsymbol{x})$. Let $\psi_0(\boldsymbol{x}) := b(\boldsymbol{x})$. In the set of class $\mathcal{K}$ functions that consist of power functions, we select the $\alpha_i(\cdot)$ functions in (2) as follows:

$$\psi_i(\boldsymbol{x}) := \dot{\psi}_{i-1}(\boldsymbol{x}) + p_i\psi_{i-1}^{q_i}(\boldsymbol{x}), \quad i \in \{1, \dots, m\} \tag{12}$$

where $p_i > 0, i \in \{1, \dots, m\}$ and $q_i > 0, i \in \{1, \dots, m\}$. Then, we can obtain the HOCBF constraint (4) when combining with dynamics (1), as shown in Def. 2.

For each type of unsafe set $j \in S_t$, we consider an arbitrary location for it and get an unsafe set constraint $b_j(\boldsymbol{x}(t)) \geq 0$, similar to (8). Let $\boldsymbol{p} := (p_1, \dots, p_m)$, $\boldsymbol{q} := (q_1, \dots, q_m)$. We know from [3] that the values of $q_1, \dots, q_m$ affect the feasibility region of (11), as well as what time the HOCBF constraint (4) will be active, i.e., we can rewrite $b_j(\boldsymbol{x}(t_a))$ as $b_j(\boldsymbol{x}(t_a), \boldsymbol{p}, \boldsymbol{q})$. Let $\mathcal{D}_j(\boldsymbol{p}, \boldsymbol{q}) := b_j(\boldsymbol{x}(t_a), \boldsymbol{p}, \boldsymbol{q})$. Since $b_j(\boldsymbol{x}(t_a), \boldsymbol{p}, \boldsymbol{q})$ is fixed once $\boldsymbol{p}, \boldsymbol{q}$ are given, $b_j(\cdot)$ no longer explicitly depends on $\boldsymbol{x}(t_a)$, therefore, we can reformulate (9) so that the minimization is over $\boldsymbol{p}, \boldsymbol{q}$:

$$\min_{\boldsymbol{p}, \boldsymbol{q}} \mathcal{D}_j(\boldsymbol{p}, \boldsymbol{q}), j \in S_t. \tag{13}$$

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

Given an arbitrary $\boldsymbol{x}(0)$, most of the $\boldsymbol{p}, \boldsymbol{q}$ values result in infeasible solutions of problem (11), which makes (13) 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 (11) until the state convergence (7) is achieved. If problem (11) 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 [3] guarantees that $+1$ data points exist given the control bounds (10). We assume that the control bounds (10) are properly defined such that we can select balanced data sets with large enough data size from the randomly sampled data. Then we can apply a classification method, e.g., a support vector machine (SVM), to classify these two balanced sets and get a continuously differentiable hypersurface $\mathfrak{H}_j : \mathbb{R}^{2m} \to \mathbb{R}$, where

$$\mathfrak{H}_j(\boldsymbol{p}, \boldsymbol{q}) \geq 0 \tag{14}$$

denotes the set of $\boldsymbol{p}, \boldsymbol{q}$ values which leads to the feasible solution of QPs (11), i.e., the feasibility constraint for the set of $\boldsymbol{p}, \boldsymbol{q}$ values associated with the QPs (11). We wish to get the set of $\boldsymbol{p}, \boldsymbol{q}$ values such that (14) is satisfied, but (14) is usually complex, therefore, we define $\mathfrak{H}_j(\boldsymbol{p}, \boldsymbol{q})$ to be a HOCBF. 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 later.

Note that subproblem (ii) depends on subproblem (i), and the feasibility of subproblem (i) depends on control bounds (10). We have imposed the assumption that the control bounds (10) are properly defined such that the whole problem is well-posed to get a proper constraint (14) from the hypersurface. With the assistance of the feasiblity classification hypersurface, we look further to optimize (13), i.e., we consider (13) subject to (14). However, the learned hypersurface 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 (auxiliary dynamics) for $\boldsymbol{p}, \boldsymbol{q}$ as the variations of $\boldsymbol{p}, \boldsymbol{q}$ that are controlled, i.e., we have

$$(\dot{\boldsymbol{p}}(\mathrm{t}), \dot{\boldsymbol{q}}(\mathrm{t})) = \boldsymbol{\nu}(\mathrm{t}), \qquad \boldsymbol{p}(\mathrm{t}_0) = \boldsymbol{p}_0, \boldsymbol{q}(\mathrm{t}_0) = \boldsymbol{q}_0, \tag{15}$$

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 (13), which is different and independent from t in (1) and problem (11). $\mathfrak{t}_0 \in \mathbb{R}$ denotes the initial time.

Considering feasibility of problem (11), the dynamic process that is determined by $\boldsymbol{\nu}$ should be subjected to (15), as well as subjected to the HOCBF constraint for (14) since we define the hypersurface in (14) to be a HOCBF to simplify the feasibility constraint (14), as discussed in the last three paragraphs. Note that we can get the feasibility constraint (14) from the classifier hypersurface $\mathfrak{H}_j(\boldsymbol{p}, \boldsymbol{q})$. Since we take all the state variables of the auxiliary dynamics (15) as the input for the classifier, the relative degree of the feasibility constraint (14) with respect to (15) is 1, i.e., we only need to differentiate $\mathfrak{H}_j(\boldsymbol{p}, \boldsymbol{q})$ along the dynamics (15) once to let $\boldsymbol{\nu}$ show up. We then define the hypersurface in (14) as a HOCBF with $m = 1$ for the auxiliary system (15), and a control $\boldsymbol{\nu}$ should satisfy the HOCBF constraint (4) which in this case is:

$$\frac{d\mathfrak{H}_j(\boldsymbol{p}, \boldsymbol{q})}{d(\boldsymbol{p}, \boldsymbol{q})}\boldsymbol{\nu} + \alpha_1(\mathfrak{H}_j(\boldsymbol{p}, \boldsymbol{q})) \geq 0, \tag{16}$$

where $\alpha_1(\cdot)$ is a class $\mathcal{K}$ function. Any control $\boldsymbol{\nu}$ that satisfies (16) implies that the resulting $\boldsymbol{p}, \boldsymbol{q}$ (determined by $\boldsymbol{\nu}$) leads to a feasible solution of QPs (11) in the dynamic process.

We implement the feasibility constraint (14) by the HOCBF constraint (16) in which the control $\boldsymbol{\nu}$ explicitly shows. However, the cost function (13) is only defined over the state of the auxiliary dynamics (15), 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 (13) as our new cost to let $\boldsymbol{\nu}$ show up in the cost function. As long as the derivative of (13) is negative, we make sure that (13) is decreasing in each time step by discretizing $\mathfrak{t}$ similar to sub-problem (i). The dynamic process to simplify the solution of problem (13) is as follows.

By taking the derivative of (13) with respect to $\mathfrak{t}$, we have

$$\frac{d\mathcal{D}_j(\boldsymbol{p}(\mathfrak{t}), \boldsymbol{q}(\mathfrak{t}))}{d\mathfrak{t}} = \frac{d\mathcal{D}_j(\boldsymbol{p}(\mathfrak{t}), \boldsymbol{q}(\mathfrak{t}))}{d(\boldsymbol{p}(\mathfrak{t}), \boldsymbol{q}(\mathfrak{t}))}\boldsymbol{\nu}. \tag{17}$$

Then, we reformulate sub-problem (ii) through the dynamic process (15). 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, we solve

$$\min_{\boldsymbol{\nu}(\mathfrak{t})} \frac{d\mathcal{D}_j(\boldsymbol{p}(\mathfrak{t}), \boldsymbol{q}(\mathfrak{t}))}{d(\boldsymbol{p}(\mathfrak{t}), \boldsymbol{q}(\mathfrak{t}))}\boldsymbol{\nu}(\mathfrak{t}), \qquad \text{s.t. } (16), (15). \tag{18}$$

Then update (15) 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 (18) 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 (18) 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 (18), and we want to limit this step length. Therefore, we add limitations to $\boldsymbol{\nu}$ for the LP (18):

$$\boldsymbol{\nu}_{min} \leq \boldsymbol{\nu} \leq \boldsymbol{\nu}_{max}. \tag{19}$$

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

After adding (19) to (18), the dynamic process search step length will become bounded. Although there are control limitations on $\boldsymbol{\nu}$, the resulting LP from the optimization (18) is always feasible as the relative degree of (14) with respect to (15) is 1 [3]. 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 (18).

The resulting process is the FGO algorithm formulated from (18) to optimize $\boldsymbol{p}, \boldsymbol{q}$. For each step of the FGO algorithm, the following four conditions may terminate it: (i) the problem (11) becomes infeasible (since the hypersurface from SVM 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 (13) is greater than the current known minimum value. (iv) the iteration time exceeds some $N \in \mathbb{N}$. The FGO algorithm can be found in [17].

If we consider (18) without the constraint (16), 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 (11). Meanwhile, the hypersurface in (14) 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 (14) after running the FGO algorithm for a number of different initial samples $\boldsymbol{p}_0, \boldsymbol{q}_0$, i.e., re-initialize (15) for each FGO process. We will show how this may affect the performance of FGO in the case studies considered in Sec. V. Once we have learned feasibility and robustness for some type-known unsafe sets with the FGO algorithm, we can use these unsafe sets to approximate other types of unsafe sets.

Remark 2: *The time complexity of subproblem* (i), i.e., the QP (11), 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) [18] and A* [19], as seen in Sec. V.

Remark 3: *The time complexity of subproblem* (ii) is that of a LP [20], 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.

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 $1m$ uncertainty) to detect the obstacles.

The robot 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 consider cost (6) as the energy consumption: $\min_{\boldsymbol{u}(t)} \int_0^{t_f} \left[u_1^2(t) + u_2^2(t)\right] dt$. 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 (7). The robot dynamics are not full state linearizable [15] 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., $\theta(t) \to \arctan(\frac{y_d - y(t)}{x_d - x(t)})$, $v(t) \to v_0, \forall t \in [0, t_f]$. Now, we can apply the CLF method since the relative degrees of the heading angle and speed are 1.

The unsafe sets (8) are defined as circular obstacles:

$$\sqrt{(x(t) - x_i)^2 + (y(t) - y_i)^2} \geq r, \forall i \in S, \qquad (20)$$

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

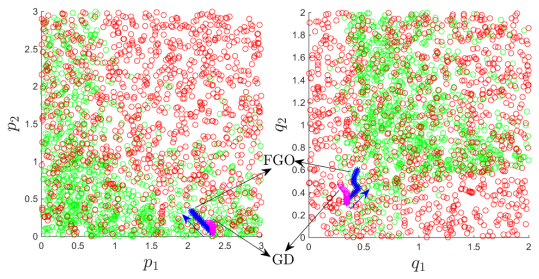
The speed and control constraints (10) are defined as: $V_{min} \leq v(t) \leq V_{max}, u_{1,min} \leq u_1(t) \leq u_{1,max}, u_{2,min} \leq u_2(t) \leq u_{2,max}$,, where $V_{min} = 0m/s, V_{max} = 2m/s, u_{1,max} = -u_{1,min} = 0.2rad/s, u_{2,max} = -u_{2,min} = 0.5m/s^2$. Other parameters are $p_0 = 1, \Delta t = 0.1s$. $\Delta \mathrm{t} = 0.1, \boldsymbol{\nu}_{max} = -\boldsymbol{\nu}_{min} = (0.1, 0.1, 0.1, 0.1)$.

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 map for FGO training is shown in Fig. 2(a).

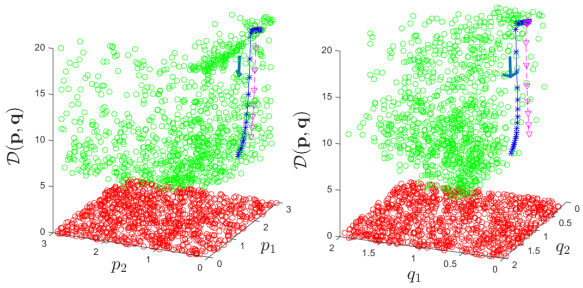
Note that the value of ε will affect the trajectory of the robot since we have a circular obstacle. If $\varepsilon = 0$, the robot will eventually stop at the equilibrium point shown in Fig. 2(a). If $\varepsilon > 0$, the robot goes left around the obstacle as shown in Fig. 2(a). Otherwise, the robot turns right.

We choose a very small $\varepsilon \neq 0$ in our FGO algorithm. Since the obstacle constraint (20) is with relative degree 2 with respect to the dynamics, we have $\boldsymbol{p} = (p_1, p_2), \boldsymbol{q} = (q_1, q_2)$. We get 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 with M training and 1000 testing samples 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 $k(\boldsymbol{y}, \boldsymbol{z}) = (c_1 + c_2 \boldsymbol{y}^T \boldsymbol{z})^7$, 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 vaule (9) of subproblem (ii) than the GD method).



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



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

Fig. 1. 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 (16) 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 (16) further increases, which may be due to over-fitting. One comparison example between FGO and GD search paths is shown in Fig. 1(a), 1(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 from GD, the additional improvement percentage is around 5% among all the testing samples.

We have implemented the learned optimal penalties and powers $(p_1^*, p_2^*, q_1^*, q_2^*) = (0.7426, 1.9745, 1.9148, 0.7024)$ 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.

We also compared the CBF-based robot exploration framework with the RRT [18] and A* [19] algorithms by consider-

TABLE I

COMPARISONS BETWEEN THE GD AND FGO ALGORITHMS

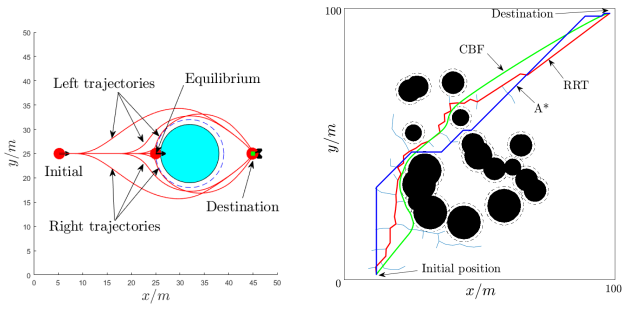
items	GD	FGO							
Training sample number M		500	1000	1500	2000	2500	3000	3500	4000
Classification accuracy		0.879	0.927	0.939	0.953	0.960	0.963	0.966	0.970
Better than GD percentage		0.210	0.248	0.254	0.252	0.244	0.282	0.288	0.266
Worse than GD percentage		0.270	0.190	0.232	0.204	0.218	0.218	0.240	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	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

ing the configuration shown in Fig. 2(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 re-plan 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 the change of environment and computationally efficient.



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

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

Fig. 2. Case study setup and planning frameworks comparison.

VI. CONCLUSIONS

The proposed feasibility-guided learning approach has shown an improved ability to determine the optimal parameters of a HOCBF compared with the gradient-descent method in terms of feasibility robustness. 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 methods. Although the improvement of the FGO in the studied example is not impressive compared with GD, the proposed FGO offers an effective approach to optimize system parameters, and has the potential to have better performance on other problems. Future work will focus on how to deal with traps formed by obstacles, and apply the proposed method to moving obstacles.

REFERENCES

[1] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *Proc. of IEEE Conf. on Decision and Control*, 2014, pp. 6271–6278.

[2] 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.

[3] 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.

[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] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE Trans. on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, 2007.

[8] 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.

[9] 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*, Florence, Italy, 2013, pp. 1478–1483.

[10] M. Hosseinzadeh and E. Garone, "An explicit reference governor for the intersection of concave constraints," *IEEE Transactions on Automatic Control*, vol. 65, no. 1, pp. 1–11, 2020.

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

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

[13] 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.

[14] M. Srinivasan, S. Coogan, and M. Egerstedt, "Feasibility envelopes for metric temporal logic specifications," in *Proc. of 57th IEEE Conference on Decision and Control*, 2018, pp. 1991–1996.

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

[16] W. Xiao, C. G. Cassandras, and C. Belta, "Bridging the gap between optimal trajectory planning and safety-critical control with applications to autonomous vehicles," *Automatica (provisionally accepted, available in arXiv: 2008.07632)*, 2020.

[17] W. Xiao, C. Belta, and C. G. Cassandras, "Feasibility-guided learning for robust control in constrained optimal control problems," *preprint arXiv:1912.04066*, 2020.

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

[19] 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.

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