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-descent method 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 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 $K$ 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) [10] [11] 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 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 $K$ functions in the definition of a High Order CBF (HOCBF) for an arbitrary relative degree constraint.
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 feasibility 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 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. We further define a sequence of sets $C_1, C_2, \ldots, C_m$ associated with (2) in the form:

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

**Definition 4:** (High Order Control Barrier Function (HOCBF) [21]) Let $C_1, C_2, \ldots, C_m$ be defined by (3) and $\psi_1(x), \psi_2(x), \ldots, \psi_m(x)$ be defined by (2). A function $b : \mathbb{R}^n \rightarrow \mathbb{R}$ is a high order control barrier function (HOCBF) of relative degree $m$ for system (1) if there exist differentiable class $K$ functions $\alpha_1, \alpha_2, \ldots, \alpha_m$ such that

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

for all $x \in C_1 \cap C_2 \cap \ldots \cap C_m$. In (4), $L_i, L_g$ 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:

$$K_{cbf} = \{u \in U : L_i^m b(x) + L_g L_i^{m-1} b(x) u + O(b(x)) + \alpha_m(\psi_{m-1}(x)) \geq 0\} \quad (5)$$

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

**Definition 5:** (Control Lyapunov function (CLF) [1]) A continuously differentiable function $V : \mathbb{R}^n \rightarrow \mathbb{R}$ 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

$$c_1 \|x\|^2 \leq V(x) \leq c_2 \|x\|^2 \quad (6)$$

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

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

**Theorem 2:** ([11]) Given an exponentially stabilizing CLF $V$ as in Def. 5 any Lipschitz continuous controller $u \in K_{clf}(x)$, with

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

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 \( \mathbf{f} \) 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:

\[
\min_{\mathbf{u}(t)} J(\mathbf{u}(t)) = \int_0^{t_f} C(\|\mathbf{u}(t)\|) \, dt, \tag{8}
\]

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

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

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

where \( \xi > 0 \) denotes arbitrarily small and \( t' \in [0, t_f] \).

**Constraint 1 (Unsafe Sets):** Let \( S_o \) denote a set of unsafe regions (obstacles) \( j \in S_o \), i.e.,

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

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 \( \mathbf{u} \) makes inequality (10) 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 \( h_j(\mathbf{x}(t_a)) \) (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 (active afterwards) at \( t_a \in [0, t_f] \). In order to maximize the feasibility robustness, we need to minimize

\[
\min_{t_a} h_j(\mathbf{x}(t_a)), j \in S_o. \tag{11}
\]

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 \( h(\mathbf{x}) := z(t) - \delta \) (\( m = 2 \) 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 \( h(\mathbf{x}(t_a)) \) 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 \( \min_{t_a} h(\mathbf{x}(t_a)) \).

**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:

\[
x_{\min} \leq \mathbf{x}(t) \leq x_{\max}, \forall t \in [0, t_f], \tag{12}
\]

where \( x_{\max} := (x_{\max,1}, x_{\max,2}, \ldots, x_{\max,n}) \in \mathbb{R}^n \) and \( x_{\min} := (x_{\min,1}, x_{\min,2}, \ldots, x_{\min,n}) \in \mathbb{R}^n \) 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:

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

where \( u_{\min} \in \mathbb{R}^q \) and \( u_{\max} \in \mathbb{R}^q \) 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 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 \( \xi \) 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 \( S_i \subseteq S_o \) denote the index set of all the unsafe set types in \( S_o \).
A. HO-CBF 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 \(\{0, \Delta t, \lfloor 2\Delta t\}, \ldots\) where \(\Delta t > 0\). In each interval \(\omega \Delta t, (\omega + 1)\Delta t\) \(\omega = 0, 1, 2, \ldots\), we assume the control is constant (i.e., the overall control will be piece-wise constant). Then at \(t = \omega \Delta t\), we solve

\[
\min_{u(t), \delta(t)} C(|u(t)|) + p\delta^2(t) \tag{14}
\]

subject to \(\|\|\) the CLF constraint \(\mathcal{J}_1\) for \(\mathcal{J}_2\) (by defining a CLF for \(\mathcal{J}_2\) such that a CLF constraint similar to \(\mathcal{J}_1\) is satisfied) and the HO-CBF constraints \(\mathcal{J}_3\) corresponding to \(\mathcal{J}_4\) and \(\mathcal{J}_5\), where \(p > 0\) is a penalty on the relaxation \(\delta(t)\) of the relaxation \(\delta(t)\) is a relaxation variable that replaces \(0\) on the right-hand side of \(\mathcal{J}_1\). 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 HO-CBF such that the feasibility robustness is maximized.

B. The Penalty Method

To improve the feasibility [21] of the problem \(\mathcal{J}_4\), we add penalties on the class \(K\) functions \(\alpha_1(t), \alpha_2(t), \ldots, \alpha_m(t)\) in \(\mathcal{J}_3\) in the definition of a HO-CBF \(b(x)\). In the set of class \(K\) functions that consist of power functions, we explicitly rewrite \(\mathcal{J}_3\) as

\[
\psi_0(x) := b(x), \quad \psi_1(x) := \psi_0(x) + p_1\psi_0^1(x), \quad \psi_m(x) := \psi_{m-1}(x) + p_m\psi_{m-1}^m(x),
\]

\[
\forall m \geq 1 \quad \text{as } \psi_m(x) := \psi_{m-1}(x) + p_m\psi_{m-1}^m(x),
\]

where \(p_1 > 0, p_2 > 0, \ldots, p_m > 0\) and \(q_1 > 0, q_2 > 0, \ldots, q_m > 0\).

For each type of unsafe set \(\mathcal{J}_t \in \mathcal{S}_t\), we consider an arbitrary location for it and get an unsafe set constraint \(h_j(x(t)) \geq 0\) (similar to \(\mathcal{J}_6\)). Let \(p := (p_1, p_2, \ldots, p_m), q := (q_1, q_2, \ldots, q_m)\). We know from [21] that the values of \(p_1, q_1, q_2, \ldots, q_m\) affect the feasibility region of \(\mathcal{J}_6\), as well as what time the HO-CBF constraint \(\mathcal{J}_3\) will be active, i.e., we can rewrite \(h_j(x(t))\) as \(h_j(x(t), p, q)\). Let \(D_j(p, q) := h_j(x(t), p, q)\) since \(h_j(x(t), p, q)\) is fixed once \(p, q\) are given. \(h_j(\cdot)\) does not actually depend on \(x(t)\).

We reformulate \(\mathcal{J}_6\) as

\[
\min_{p, q} D_j(p, q), j \in \mathcal{S}_t. \tag{16}
\]

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

Then, we need to find the optimal \(p\) and \(q\) that minimize \(\mathcal{J}_6\) for each unsafe set \(j \in \mathcal{S}_t\). 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 \(\mathcal{J}_6\) is a typical problem that can be solved with reinforcement learning approaches. However, most of the \(p, q\) values result in infeasible solutions of problem \(\mathcal{J}_4\), which makes \(\mathcal{J}_6\) difficult to solve. Therefore, we need to first solve the infeasibility problem of sub-problem (i). We randomly sample \(p, q\) values over their domain, and for each set of \(p, q\) values, we solve problem \(\mathcal{J}_6\) until the state convergence \(\mathcal{J}_5\) is achieved. If problem \(\mathcal{J}_4\) (the QPs) is feasible at all times, then we label this set of \(p, q\) values (as a whole) as +1, otherwise, we label it as −1. Eventually, we get sets of feasible and infeasible \(p, q\) 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:

\[
\hat{\mathcal{J}}_j : \mathbb{R}^{2m} \rightarrow \mathbb{R}, \tag{17}
\]

where

\[
\hat{\mathcal{J}}_j(p, q) \geq 0 \tag{18}
\]

denotes the set of \(p, q\) values (as a whole) which leads to the feasible solution of QPs \(\mathcal{J}_4\), i.e., the feasibility constraint for the set of \(p, q\) values associated with the QPs \(\mathcal{J}_4\). We can use a HO-CBF to enforce \(\mathcal{J}_6\) if \(p, q\) are state variables of a system, which motivates us to define dynamics for \(p, q\), as shown later.

Based on the feasibility classification hypersurface \(\mathcal{J}_6\), we look further to optimize \(\mathcal{J}_6\), i.e., we consider \(\mathcal{J}_6\) subject to \(\mathcal{J}_6\). However, the learned hypersurface \(\mathcal{J}_6\) 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 \(p_0, q_0 \in \mathbb{R}^m\), and then search for the optimal \(p, q\) values. Since the determination of the optimal \(p, q\) is a dynamic process, we define the gradient (dynamics) for \(p, q\) (as the variations of \(p, q\) that are controlled), i.e., we have

\[
(p(t), q(t)) = \nu(t), \hat{p}(t_0) = p_0, \hat{q}(t_0) = q_0. \tag{19}
\]

where \(\nu \in \mathbb{R}^{2m}\) denotes a controllable input vector in the dynamic process constructed in order to determine the optimal \(p, q\). \(t\) denotes the dynamic process time for the optimization of \(\mathcal{J}_6\), which is different and independent from \(t\) in the system \(\mathcal{J}_4\) and problem \(\mathcal{J}_6\). \(t_0 \in \mathbb{R}\) denotes the initial time.

Considering feasibility of the problem \(\mathcal{J}_6\), the dynamic process (determined by \(\nu\)) should be subjected to \(\mathcal{J}_6\) and \(\mathcal{J}_6\). Since we want to find the control \(\nu\) such that the resulting \(p, q\) (determined by \(\nu\)) always lead to the feasible solution of QPs \(\mathcal{J}_4\) with the CBF method, i.e., we need to take the derivative of \(\mathcal{J}_6\), we minimize the derivative of \(\mathcal{J}_6\) (the fastest decreasing direction of the value of \(\mathcal{J}_6\)) in the dynamic process to make \(\nu\) also show up in the cost function. As long as the derivative of \(\mathcal{J}_6\) is negative, we make sure that \(\mathcal{J}_6\) is decreasing in each time step (by discretizing \(t\) similar to sub-problem (i)).
By taking the derivative of \( (16) \) with respect to \( t \), we have
\[
\frac{dD_j(p(t), q(t))}{dt} = \frac{dD_j(p(t), q(t))}{dp(t), q(t)} \frac{dp(t), q(t)}{dt} - \frac{dD_j(p(t), q(t))}{d(t)}
\]
(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 \( \nu \) that can satisfy \( (18) \) in the dynamic process:
\[
\frac{d\delta \nu_j(p, q)}{d\nu_j(p, q)} + \alpha_1(\delta \nu_j(p, q)) \geq 0,
\]
(21)
where \( \alpha_1(\cdot) \) is a class \( K \) function as in Def. 4 (the definition of \( \psi_1(\cdot) \) in (2)). Any control input \( \nu \) that satisfies (21) implies that the resulting \( p, q \) (determined by \( \nu \)) 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. V-A to solve the dynamic process, i.e., we discretize \( t \), at each \( t = \omega \Delta t, \omega \in \{0, 1, \ldots\} \) (\( \Delta t > 0 \) denotes the discretization constant), and we solve
\[
\min_{\nu(t)} \frac{dD_j(p(t), q(t))}{d(p(t), q(t))} \nu(t)
\]
(22)
superscripted to \( (21), (19) \). Then update \( (19) \) for \( t \in \{\omega \Delta t, (\omega + 1) \Delta t\} \) with \( \nu(t) \). Note that in the last equation, \( \frac{dD_j(p(t), q(t))}{d(p(t), q(t))} \) is a vector of dimension \( 1 \times 2m \), while \( \nu \) is a vector of dimension \( 2m \times 1 \). Therefore, the cost function in the last equation is a scalar function of \( \nu \).

The optimization problem (22) is a linear program (LP) (to determine \( \nu \)) at each time step for each initial \( p, q \) (we need to reset \( t \) for each set of initial \( p, q \) values). Without any constraint on \( \nu \), the LP (22) is ill-posed because it leads to unbounded solutions. In fact, the value of \( \nu \) 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 \( \nu \) for the LP (22):
\[
\nu_{\min} \leq \nu \leq \nu_{\max},
\]
(23)
where \( \nu_{\min} < 0, \nu_{\max} > 0 \) (interpreted componentwise), \( 0 \in \mathbb{R}^{2m} \).

After adding (23) to (22), the dynamic process search step length will become bounded. Although there are control limitations on \( \nu \), 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
\[
\frac{dD_j(p, q)}{dp_m, \ldots, dp_{m+q_m}}
\]
and we also need to evaluate \( \frac{dD_j}{dp_{m+1}, \ldots, dp_{m+q_m}} \) 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 \( \frac{\partial D_j, \ldots, \partial D_j}{\partial p_k), \ldots, \partial p_m} \) are all 0, (iii) the objective function value of (16) is greater than the current known minimum value. (iv) the iteration time exceeds some \( \mathcal{N} \in \mathbb{N} \).

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: \( p^*, q^*, D_{\min} \)

Sample \( p, q \) 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 \( p_0, q_0 \), \( D_{\min} := D_j(p_0, q_0) \), iter. = 1;  while iter.++ \( \leq N \) do

- Evaluate \( \frac{\partial D_j}{\partial p_k}, \ldots, \frac{\partial D_j}{\partial q_k} \), where \( k \in \{1, 2, \ldots, m\} \) is infeasible then
  - Jump to the very beginning of the loop;
- Solve optimization (22) and get new \( p, q \);

- if \( (14) \) is feasible then
  - if \( D_{\min} \geq D_j(p, q) \) then
    - \( D_{\min} = D_j(p, q) \), \( p_0 = p \), \( q_0 = q \);
  - else
    - break;
  - end
- else
  - Solve optimization (22) without (21) and get new \( p, q \);
- Solve problem (14) with \( p, q \);
  - if \( D_{\min} \geq D_j(p, q) \) then
    - \( D_{\min} = D_j(p, q) \), \( p_0 = p \), \( q_0 = q \);
  - else
    - break;
  - end
- end

end

\( p^* = p_0, q^* = q_0 \).
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 $p^*, q^*$ with the FGO algorithm, then the optimal $p^*, q^*$ 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 $p^*, q^*$ under the worst initial conditions (e.g., with maximum obstacle-approaching speed for a robot), and then these optimal $p^*, q^*$ 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 $p^*, q^*$ are found under this condition, 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 ($\frac{3}{4}\pi$ field of view (FOV)) and 7 m sensing distance with 1 m sensing uncertainty) to detect the obstacles.

The robot dynamics are defined in the form:

$$
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{\theta} \\
\dot{v}
\end{bmatrix} =
\begin{bmatrix}
v \cos(\theta) \\
v \sin(\theta) \\
0 \\
f(x) \\
g(x)
\end{bmatrix}
+ \begin{bmatrix}
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2
\end{bmatrix},
$$

where $x, y$ denote the location along $x, y$ axis, respectively, $\theta$ 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:

$$J(u(t)) = \int_{0}^{\tau_f} \eta \max\{u_2^2, u_1^2\} + \frac{\max\{u_2^2, u_1^2\}}{\max\{u_{1,\min}, u_{1,\max}\}^2} u_1^2(t) + (1-\eta) u_2^2(t) dt$$

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 $\theta$ stabilizes to the desired direction and its speed $v$ stabilizes to a desired speed $v_0 > 0$, i.e.,

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

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:

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

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:

$$v_{\min} \leq v(t) \leq v_{\max}, \forall t \in [0, t_f],$$

$$u_{1,\min} \leq u_1(t) \leq u_{1,\max}, \forall t \in [0, t_f],$$

$$u_{2,\min} \leq u_2(t) \leq u_{2,\max}, \forall t \in [0, t_f],$$

$$v_{\min} \geq 0, v_{\max} > 0$$

denote the minimum and maximum speed, respectively. The simulation parameters are listed in Table I.

| Parameter | Value | Unit | Parameter | Value | Unit |
|-----------|-------|------|-----------|-------|------|
| $p$       | 1     | unitless | $r$ | 7 | m |
| $\epsilon$ | 10   | unitless | $\Delta t$ | 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/s² | $u_{2,\max}$ | 0.5 | m/s² |

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. $\Delta t = 0.1, v_{\max} = -v_{\min} = (0.1, 0.1, 0.1, 0.1)$. The map for FGO training is shown in Fig. 1.

Note that the value of $\varepsilon$ 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
with respect to system (24), we have trajectories.

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

\[
k(y, z) = (c_1 + c_2 y^T z)^7.
\]

where \( y, z \) denote input vectors of SVM (i.e., \( y := (p, q) \)), as well as for \( z \). 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 \( D_{\text{min}} \) when the number of training samples \( M \) for the hypersurface (21) is large enough, as shown in Table [II]

The implementation on a robot safe exploration 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. Budhiraia, 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 | 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 |
| $D_{\text{min}}/m$ (samples min.: 5.0) | 4.6 | 4.6 |
|                        | 4.6 | 4.6 |
|                        | 4.6 | 4.6 |
|                        | 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 $p, q$ 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  | $< 0.01$ s        | 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. Nilsson 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.