Learning Differentiable Safety-Critical Control using Control Barrier Functions for Generalization to Novel Environments

Hengbo Ma*, Bike Zhang*, Masayoshi Tomizuka, and Koushil Sreenath

***Abstract*— Control barrier functions (CBFs) have become a popular tool to enforce safety of a control system. CBFs are commonly utilized in a quadratic program formulation (CBF-QP) as safety-critical constraints. A class $\mathcal{K}$ function in CBFs usually needs to be tuned manually in order to balance the trade-off between performance and safety for each environment. However, this process is often heuristic and can become intractable for high relative-degree systems. Moreover, it prevents the CBF-QP from generalizing to different environments in the real world. By embedding the optimization procedure of the exponential control barrier function based quadratic program (ECBF-QP) as a differentiable layer within a deep learning architecture, we propose a differentiable safety-critical control framework that enables generalization to new environments for high relative-degree systems with forward invariance guarantees. Finally, we validate the proposed control design with 2D double and quadruple integrator systems in various environments.**

I. Introduction

Safety plays a critical role in autonomous systems that interact with people, such as autonomous driving and robotics. There are several approaches to developing a safe control strategy, e.g., Hamilton-Jacobi reachability analysis [7] and model predictive control [18]. However, such methods may have high computational costs in real-time applications. Control barrier functions (CBFs) [4] have gained more attention recently since these methods only depend on the current state and do not require heavy computation. CBFs are usually encoded as constraints in a quadratic program (CBF-QP) for safety-critical tasks [3]. With a properly chosen class $\mathcal{K}$ function in CBFs, a system can avoid unsafe sets. Meanwhile, it does not reduce the stabilizing performance from a high-level controller [4]. However, the performance of the overall controller, which consists of a high-level controller and CBF-QP, can be easily undermined if the environment changes. In other words, each safe set in CBFs necessitates a unique class $\mathcal{K}$ function that maximizes the overall control performance for a specific environment. In the real world, the environment information for safety-critical tasks is usually not fully known a priori, and a system might also face different environments during its deployment. Thus, it is hard to tune a class $\mathcal{K}$ function for each environment beforehand to reconcile performance and safety. Moreover, the tuning process for choosing a class $\mathcal{K}$ function becomes tedious when there are multiple control barrier function constraints in the CBF-QP [31], or some are with high relative-degree in the ECBF-QP [23]. This challenge impedes the progress towards deploying CBF-based safety-critical controllers in the real world.

To address this challenge, we investigate how to model the relation between environment information and safety-critical control. We propose a learning safety-critical control framework using an environment-dependent neural network which satisfies the forward invariance condition. Thanks to the development of differentiable convex optimization [1], we can enable the learning procedure in an end-to-end style. After offline training, we can directly deploy the proposed safety-critical control framework in different environments without any adaption.

A. Related Work

1) Safe environment generalization: Cluttered environments have been considered in the safe control literature. A provably approximately correct-bayes framework is proposed to synthesize controllers that provably generalize to novel environments in [21]. A control Lyapunov function and control barrier function based quadratic program (CLF-CBF-QP) is utilized with a high-level path plan to navigate through obstacle-scattered environments in [8]. Moreover, for hostile environments with adversarial agents, a probabilistic tree logic method is proposed in [11] to assure safety. Safe generalization problem with control barrier functions is considered with a weighted mixture of existing controllers in [28]. Yet the generalization ability is largely limited by the number of existing controllers. With the help of reinforcement learning, safe environment adaptation is studied through a risk-averse approach in [36]. However, this approach can only provide relative safety instead of safety guarantee. For robotic applications, bipedal robot walking on stepping stones is addressed in [22] using a robust control barrier function method, where the distances between adjacent stones are different at each step. Predictive control with CBFs tackles the safe car overtaking problems in [35], where different leading cars serve as novel environments. Our approach adopts a different way using class $\mathcal{K}$ function to generalize a controller to enforce safety under different environments.

2) Safe learning control: A safe reinforcement learning (RL) framework under constrained Markov decision process is proposed in [10] using a Lyapunov based method. A learning-based control barrier function from expert demonstration is proposed in [25] to ensure safety. In [27], a CBF is created using RL for risk mitigation in adversarial environments. In [9] and [29], they address the model uncertainty

*The authors contributed equally to this work and names are in alphabetical order.

H. Ma, B. Zhang, M. Tomizuka and K. Sreenath are with University of California, Berkeley, CA 94720, USA (e-mail:hengbo_ma, bikezhang, tomizuka, koushils@berkeley.edu).

problem by learning CBF constraints. In [12], the authors design a learning robust control Lyapunov barrier function that can generalize despite model uncertainty. A model-free safe reinforcement learning is studied by synthesizing a barrier certificate and querying a black-box dynamic function in [37]. A game theoretic approach is adopted in [30] to reduce conservatism while maintaining robustness during human robot interaction. Differentiable optimization layers have emerged as a new approach for safe learning control recently. In [24], a differentiable layer is applied to control barrier function based quadratic program in order to enhance the recursive feasibility, where the parameters are adapted online. In [15], safety is framed as a differentiable robust CBF layer in model-based RL. We also utilize the differentiable optimization layer as a tool. However, we focus on generalizing the safety-critical control to novel environments.

B. Contributions

The contribution of this paper is as follows:

- We present an approach to generalizing safety-critical control to novel environments by integrating control barrier functions and differentiable optimization.
- We introduce a neural network based ECBF-QP and formulate the safety-critical control as a differentiable optimization layer.
- We show that the proposed neural network module based on the exponential control barrier function assures the forward invariance of a safe set.
- We numerically validate the proposed learning control design using systems with different relative-degrees and novel environments with randomly generated obstacles.

C. Organization

This paper is organized as follows: in Sec. II, we introduce the background of control barrier functions and differentiable optimization. The problem formulation is illustrated in Sec. III, where we motivate the formulation with a simple case study. Then, in Sec. IV, we present the methodology of learning differentiable safety-critical control using control barrier functions. In Sec. V, we test the proposed control logic on 2D double and quadruple integrator systems with different environment settings. Secs. VI and VII provides discussion and concluding remarks.

II. Background

Throughout this paper, we will consider a nonlinear control affine system:

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

where $x \in \mathcal{X} \subset \mathbb{R}^n$ represents the state of the system, $u \in \mathbb{R}^m$ is the control input, and $f : \mathcal{X} \to \mathbb{R}^n$ and $g : \mathcal{X} \to \mathbb{R}^m$ are locally Lipschitz continuous.

A. Control Barrier Functions

Definition 1. *[19] A Lipschitz continuous function $\alpha : [0, a) \to [0, \infty), a > 0$ is said to belong to class $\mathcal{K}$ if it is strictly increasing and $\alpha(0) = 0$. Moreover, α is said to belong to class $\mathcal{K}_\infty$ if it belongs to class $\mathcal{K}$, $a = \infty$, and $\lim_{r\to\infty} \alpha(r) = \infty$.*

Definition 2. *[2, Def. 2] Consider a continuously differentiable function $h : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$ and a set $\mathcal{C}$ defined as the superlevel set of h, $\mathcal{C} = \{x \in \mathcal{X} : h(x) \geq 0\}$, then h is a control barrier function (CBF) if there exists an extended class $\mathcal{K}_\infty$ function α such that for the control system* (1)*:*

$$\sup_{u\in\mathbb{R}^m} [L_f h(x) + L_g h(x)u] \geq -\alpha(h(x)). \tag{2}$$

If h is a control barrier function on $\mathcal{X}$ and $\frac{\partial h}{\partial x} \neq 0$ for all $x \in \partial\mathcal{C}$, any Lipschitz continuous controller satisfying (2) renders the set $\mathcal{C}$ forward invariant [2, Thm.2]. By incorporating (2) as a constraint, a quadratic program based safety-critical controller is proposed in [3]:

CBF-QP:

$$u^*(x) = \quad \underset{u\in\mathbb{R}^m}{\arg\min} \quad \|u - u_{\text{perf}}\|^2 \tag{3a}$$

$$\text{s.t.} \quad L_f h(x) + L_g h(x)u \geq -\alpha(h(x)), \tag{3b}$$

where u_{perf} is the reference control input that can be from a high-level performance controller, which is expected to achieve the control objective. For instance, model predictive control is a popular choice as a high-level performance controller [26]. In the context of safety-critical control, a control Lyapunov function is often used in a quadratic program formulation (CLF-QP) to realize stability.

Remark 1. *In Definition 2, an extended class $\mathcal{K}_\infty$ function is required for CBF. Here, we restrict ourselves to a subclass: class $\mathcal{K}$ function, which can facilitate our learning algorithm. Typically, $\alpha(x)$ is simplified as αx, with α being a positive constant, which we term as a linear class $\mathcal{K}$ function. Previous work [24], [33], [34] have investigated how to adjust the class $\mathcal{K}$ function in order to improve the feasibility. In this work, we focus on learning a neural network based class $\mathcal{K}$ function to safely generalize to different environments.*

The CBF constraint in (3b) has been so far assumed to be relative-degree one, which typically does not held for most safety-critical constraints in robotic systems [16]. A special type of CBFs called exponential control barrier functions (ECBFs) has been introduced to enforce arbitrarily high relative-degree CBF constraints in [23].

Definition 3. *[23, Def. 1] Consider a r-times continuously differentiable function $h : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$ and a set $\mathcal{C}$ defined as the superlevel set of h, $\mathcal{C} = \{x \in \mathcal{X} : h(x) \geq 0\}$, then h is an exponential control barrier function (ECBF) if there exists a row vector $K_\alpha \in \mathbb{R}^r$ such that for the system* (1)*:*

$$\sup_{u\in\mathbb{R}^m} [L_f^r h(x) + L_g L_f^{r-1} h(x)u] \geq -K_\alpha \eta_b(x), \tag{4}$$

for $\forall x \in \{x \in \mathbb{R}^n | h(x) \geq 0\}$, *with*

$$\eta_b(x) = \begin{bmatrix} h(x) \\ \dot{h}(x) \\ \ddot{h}(x) \\ \vdots \\ h^{(r-1)}(x) \end{bmatrix} = \begin{bmatrix} h(x) \\ L_f h(x) \\ L_f^2 h(x) \\ \vdots \\ L_f^{r-1} h(x) \end{bmatrix}. \tag{5}$$

We define $\mu = L_f^r h(x) + L_g L_f^{r-1} h(x) u$, then the above dynamics of $h(x)$ can be written as the linear system

$$\begin{aligned} \dot{\eta_b}(x) &= F\eta_b(x) + G\mu, \\ h(x) &= C\eta_b(x), \end{aligned} \tag{6}$$

where

$$F = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \tag{7}$$
$$C = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}.$$

If $\mu \geq -K_\alpha \eta_b(x)$, with $(F - GK_\alpha)$ being Hurwitz and total negative, then we can guarantee that $h(x_0) \geq 0 \implies h(x(t)) \geq 0, \forall t \geq 0$ where x_0 is the initial condition.

Let $-p_i$ be the negative real eigenvalues of $(F - GK_\alpha)$. We can then define a family of functions $v_i : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$ with corresponding superlevel sets $\mathcal{C}_i$,

$$\begin{aligned} v_0(x) &= h(x), & \mathcal{C}_0 &= \{x : v_0(x) \geq 0\}, \\ v_1(x) &= \dot{v}_0 + p_1 v_0(x), & \mathcal{C}_1 &= \{x : v_1(x) \geq 0\}, \\ &\vdots & &\vdots \\ v_r(x) &= \dot{v}_{r-1} + p_r v_{r-1}(x), & \mathcal{C}_r &= \{x : v_r(x) \geq 0\}, \end{aligned} \tag{8}$$

where $\mathcal{C}_0$ plays the role of the safe set $\mathcal{C}$ as defined in Definition 2 for a relative-degree one CBF. Then, we have:

Theorem 1. *[23, Thm.2] A valid exponential CBF should satisfy two conditions: suppose K_α is chosen such that $p_i > 0$ and the eigenvalues $-p_i$ satisfy $p_i \geq -\frac{\dot{v}_{i-1}(x_0)}{v_{i-1}(x_0)}$, then* (9b) *guarantees $h(x)$ is an exponential control barrier function.*

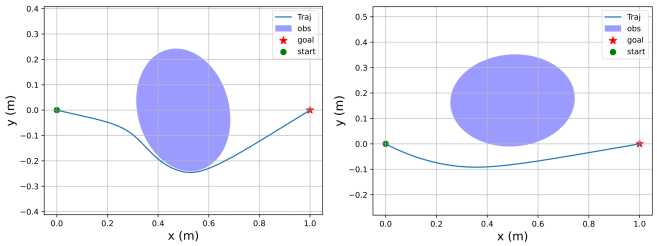
Given an ECBF, we can extend the CBF-QP in (3) to enforce high relative-degree safety-critical constraints:

ECBF-QP:

$$u^*(x) = \underset{u \in \mathbb{R}^m}{\arg\min} \quad \|u - u_{\text{perf}}\|^2 \tag{9a}$$
$$\text{s.t.} \quad L_f^r h(x) + L_g L_f^{r-1} h(x) u \geq -K_\alpha \eta_b(x), \tag{9b}$$

where u_{perf} is the reference control input. Note that the control barrier function constraint (9b) can be extended to multiple constraints in order to account for different safety criteria. Furthermore, a general formulation of high order control barrier functions can be seen in [32].

Remark 2. *In ECBF-QP* (9b)*, due to the relation between the coeeficients in K_α and the eigenvalues $-p_i$, $K_\alpha \eta_b$ can be reformulated as $\Pi_{i=1}^r [L_f + p_i] \circ h(x) - L_f^r h(x)$. This will be used to develop our differentiable safety-critical control formulation. Note that L_f here is the Lie derivative operator s.t. $L_f \circ h(x) = L_f h(x) = \frac{\partial}{\partial x} h(x) f(x)$.*



(a) Environment 1 (b) Environment 2

Fig. 1: Motivating example for safety-critical control for generalization to novel environments using a 2D double integrator. A hand-tuned K_α for Environment 1 in (a) is used in the novel Environment 2 in (b). As can be seen, this results in a trajectory with a larger deviation from the obstacle in Environment 2. Thus, a well-tuned K_α for one environment does not necessarily generalize to a different environment.

B. Differentiable Optimization

A differentiable optimization problem is a class of optimization problems whose solutions can be backpropagated through. This functionality enables an optimization problem to serve as layers within deep learning architectures, which can encode constraints and complex dependencies through optimization that traditional convolutional and fully-connected layers usually cannot capture [6]. Some successful differentiable layer examples include differentiable model predictive control [5], [14], Pontryagin's maximum principle [17], robust control [13], and meta learning [20], etc. In this work, we utilize the differentiable optimization layer presented in [1] for the ECBF-QP.

Remark 3. *While CBFs are continuously differentiable functions [2], here a differentiable safety-critical control using CBF does not mean the CBF itself is differentiable but rather that the backpropagation can go through the CBF-QP differentiable optimization layer.*

III. Problem Statement

Having established the background of CBFs and differentiable optimizations, we now present our problem formulation for generalizing to novel environments.

A. Motivating Example

Using a 2D double integrator as an illustrative example, we design a linear quadratic regulator (LQR) to drive the system to a goal location while avoiding different obstacles using the ECBF-QP in (9). The LQR controller serves as the reference performance controller u_{perf}. The simulation results are demonstrated in Fig. 1. Note that the K_α for ECBF-QP is tuned manually in Fig. 1(a), which leads to a short and smooth trajectory, i.e., a smooth trajectory that goes around the obstacle with minimal detour from a straight-line

trajectory from start to goal. However, the ECBF-QP with the same K_α results in a large detour in Fig. 1 (b) in a different environment. While larger detours are conservative, they potentially require more control effort, and result in energy inefficient motions. The motion in Environment 2 could be shorter by getting closer to the obstacle. This example demonstrates that the K_α plays an important role in generating desirable trajectories in different environments.

Remark 4. *In order to get a trajectory with desired properties, e.g., smoothness and minimum distance, it is necessary to choose a proper K_α using ECBF-QP. Moreover, certain fixed K_α that works in a particular environment could actually fail in a different environment, resulting in violation of the safety constraint $h(x) \geq 0$.*

B. Problem Formulation

Building upon the motivating example, we are motivated to optimize the class $\mathcal{K}$ function in CBF-QP or K_α in ECBF-QP with respect to different environments, which can result in a safe trajectory satisfying a user-defined metric.

To this end, we represent the $K_\alpha \eta_b(x)$ in (4) with a neural network parameterized with θ. $\Pi_\theta(u_{\text{perf}}, e, x_0)$ represents the solution of ECBF-QP mentioned in (9b). Such an ECBF-QP can be embedded as a layer in a deep learning pipeline by using differentiable convex optimization technique. Then we minimize a performance cost $\mathcal{L}$ in an episodic setting. The formulation is given as follows:

$$\begin{aligned} &\arg\min_{\theta} \quad \mathbb{E}_{x_0 \sim P_0, e \sim P_e}[\mathcal{L}(\tau, e, x_0)] \\ \text{s.t.} \quad &u = \Pi_\theta(u_{\text{perf}}(x), e, x_0), \\ &\dot{x} = f(x) + g(x)u, \end{aligned} \tag{10}$$

where e is an environment sampled from a distribution of environments P_e, e.g., e consists of center and radius of the obstacle. x is the state where we evaluate cost at each time step, and x_0 is the initial state which is sampled from a distribution P_0. Note that $\mathcal{L}$ is the cost along a trajectory instead of the cost at each time step, and τ represents the trajectory with time horizon T. u_{perf} is the performance control input provided by a high-level performance controller. Once the training procedure is done offline, we can deploy the neural network based controller Π_θ to novel environments.

IV. Methodology

Having seen the problem formulation, we will next introduce how to enable the generalization to novel environments via learning differentiable ECBF-QP.

The overall control architecture is shown in Fig. 2, which basically includes two parts: a performance controller and a differentiable ECBF-QP safety filter. The performance controller is mainly responsible for achieving the control objective, and the differentiable ECBF-QP serves as a safety filter, which will be explained in detail in this section.

A. Differentiable Safety-Critical Control using CBFs

We formulate our differentiable safety-critical control based on exponential control barrier functions in (9). Differentiable CBFs have been used in [24] and [15]. However, they used systems with relative-degree one or solved a relative-degree two system using a cascaded approach. Here, we extend it to a general formulation as follows:

Differentiable ECBF-QP:

$$\begin{aligned} &\Pi_\theta(u_{\text{perf}}, e, x_0) = \arg\min_{u \in \mathbb{R}^m, \delta \in \mathbb{R}} \ \|u - u_{\text{perf}}\|^2 + \zeta \delta^2 \\ \text{s.t.} \quad &L_f^r h(x) + L_g L_f^{r-1} h(x) u \geq -\alpha_\theta(e, x_0, \eta_b(x)) - \delta^2, \end{aligned} \tag{11}$$

where, $\Pi_\theta(u_{\text{perf}}, e, x_0)$ is the safe policy filtered by the ECBF-QP and conditioned on the high-level performance control input u_{perf} and the environment information e. $\alpha_\theta(e, x_0, \eta_b(x))$ is denoted by α-net, where θ represents parameters of the network.

As we will see next, the α function is a linear function (of η_b) that encodes an ECBF constraint within the differentiable ECBF-QP so as to deal with high relative-degree safety constraints, which are common in many robotic applications. We include a slack variable δ which guarantees that such an optimization is feasible during the training procedure, and ζ is a hyperparameter. Note that we do not use δ as in (11) during the test time.

B. The Structure of α-net

Exponential control barrier function provides a formal structure to guarantee safety with a vector parameter K_α. In general, it is not easy and probably time-consuming to find the best K_α directly in order to generalize to novel environments. We thus encode the structure of exponential CBF into our neural network. As noted in Remark 2, our formulation is shown as follows

$$\alpha_\theta(e, x_0, \eta_b(x)) = \prod_{i=1}^{r} [L_f + p_i(e, x_0; \theta)] \circ h(x) - L_f^r h(x), \tag{12}$$

where L_f is the lie derivative operator as mentioned in Remark 2. Notice that for the ECBF, the right side of (9b) only includes the lie derivative with respect to f. The function $\mathbf{p}(e, x_0; \theta) \in \mathbb{R}^r$ outputs $[p_1, p_2, \ldots, p_r]$, and $p_i(e, x_0; \theta)$ represents p_i as defined in (8). We use the following neural network structure:

$$\begin{aligned} &\mathbf{p}(e, x_0; \theta) = \text{ReLU}(\prod_{k=0}^{m} \sigma(W_k l_k) - b(x_0)) + b(x_0), \\ &b_i(x_0) = \text{ReLU}(-\frac{\dot{v}^{i-1}(x_0)}{v^{i-1}(x_0)} - \epsilon) + \epsilon, i = 1, \ldots, r, \\ &l_0 = [e, x_0], \end{aligned} \tag{13}$$

where $b_i(x_0)$ is the i-th element of $b(x_0) \in \mathbb{R}^r$, and it represents the bounds of $p_{i=1\ldots r}$ in Thm.1. The parameter θ represents the weights $\{W_0, W_1, \ldots, W_m\}$. l_k represents the outputs of the k-th layer of the neural network. We concatenate the environment information e and initial state

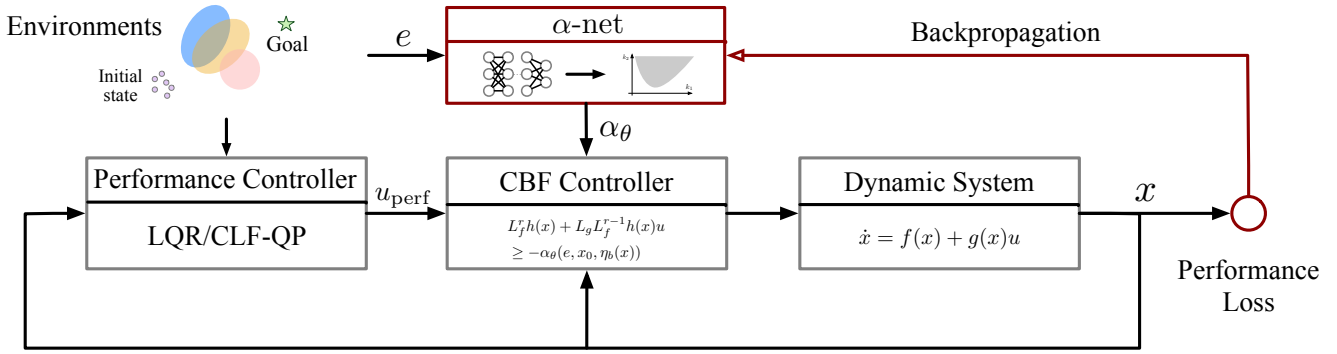


Fig. 2: The overall framework of the proposed approach, which includes two main components: a performance controller and a differentiable CBF-QP. The novel environment information, e, is an input to the performance controller and α net. The performance loss computed along a trajectory will be backpropagated through the α net, then the α net outputs the parameters to construct the class $\mathcal{K}$ function.

x_0 together as the input l_0 and choose the ReLU function as the activation, with $\sigma(\cdot)$ being any activation function. Then, we randomly initialize the neural network parameters with positive numbers.

Theorem 2. *Given the function* $\boldsymbol{p}(e, x_0; \theta)$ *defined in* (13)*,* p_i *satisfies the conditions in Thm.1. Thus,* $h(x)$ *is an exponential CBF and* $\mathcal{C}_0$ *in* (8) *is forward invariant.*

Proof. Since $b_i(x) = \max\{-\frac{\dot{v}^{i-1}(x_0)}{v^{i-1}(x_0)}, \epsilon\}$ and $p_i(e, x_0, \theta) \geq b_i(x_0)$, we have $p_i(e, x_0, \theta) \geq \max\{-\frac{\dot{v}^{i-1}(x_0)}{v^{i-1}(x_0)}, \epsilon\}$. It follows that $p_i(e, x_0, \theta)$ satisfies i) $p_i(e, x_0, \theta) \geq -\frac{\dot{v}^{i-1}(x_0)}{v^{i-1}(x_0)}$ and ii) $p_i(e, x_0, \theta) \geq \epsilon > 0$, which are the conditions in Thm.1. From [23, Thm.1], the set $\mathcal{C}_0$ is forward invariant given $h(x)$ is a valid exponential CBF. □

C. Loss Function

In general, the loss function $\mathcal{L}(\tau, e, x_0)$ can be designed with any performance evaluation metric $\mathcal{L}_{\text{perf}}$. In our work, we propose a loss function which includes two components:

$$\mathcal{L}(\tau, e, x_0) = \mathcal{L}_{\text{perf}}(\tau, e, x_0) + \lambda_\delta \sum_{t=1}^{T} \delta_t^2, \tag{14}$$

where T is the number of simulation time steps with a fixed simulation interval Δt, τ is the simulated trajectory represented by $[x_0, x_1, \ldots, x_T]$. The coefficient λ_δ is for slack variable penalty. Notice that we use a slack variable δ in (11) to make sure that the optimization program will not be interrupted by the infeasibility issue of solving the ECBF-QP. Here, δ_t represents the value of the slack variable δ at each time step. However, the gradient descent of the neural network α_θ may lead to a solution such that δ_t is large. Hence, we include the penalty of δ_t in the loss function. The ideal situation is that δ_t is zero.

D. Algorithm

The training algorithm is shown in Algorithm 1. The input is a distribution of environments P_e, and the output is the network weights θ of the α-net. For each iteration, we sample n environments and rollout trajectories τ, then the weights of the neural network are updated after each iteration.

Algorithm 1: Training algorithm

Input: Environment distribution P_e, initial state distribution P_0, simulation time interval Δt, simulation time horizon T.

Output: The network weights θ.

while $t \leq$ *number of iteration* **do**
  for *i=1:n* **do**
    Sample $e_i \sim P_e$, $x_{0,i} \sim P_0$
    Collect the trajectory τ_i by using the designed controller.
  Update $\theta_t \to \theta_{t-1} - \lambda \frac{1}{n} \nabla_\theta \sum_{e_i} \mathcal{L}_\theta(\tau_i, e_i, x_{0,i})$

When the task is obstacle avoidance, we can iteratively use the learned policy for m obstacles during the test time. The differentiable ECBF-QP in (11) becomes

$$\begin{aligned} &\Pi_\theta(u_{\text{perf}}, e, x_0) = \arg\min_{u \in \mathbb{R}^m} \|u - u_{\text{perf}}\|^2 \\ &\text{s.t.} \quad L_f^r h_j(x) + L_g L_f^{r-1} h_j(x) u \geq -\alpha_\theta(e_j, x_0, \eta_{b,j}(x)), \\ &\quad j = 1, \ldots, m, \end{aligned} \tag{15}$$

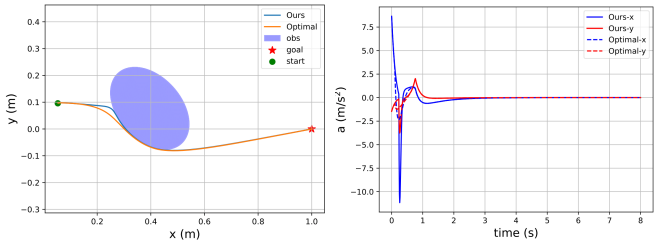
where h_j represents the j-th exponential CBF. The environment e can have multiple obstacles and each of them e_j can be captured by an ECBF constraint.

V. Results

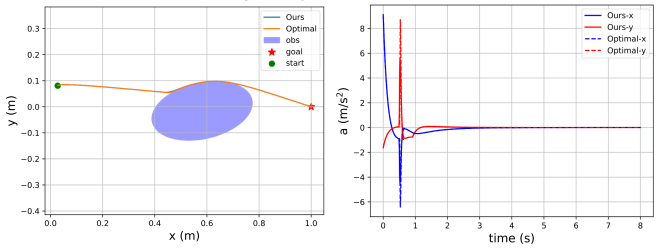
After developing our methodology for learning differentiable safety-critical control using CBFs, we now present the simulation results of our proposed framework using 2D double and quadruple integrator systems.

A. Simulation Setup

We focus on the collision avoidance problem. We set up different environments with different obstacles, which are

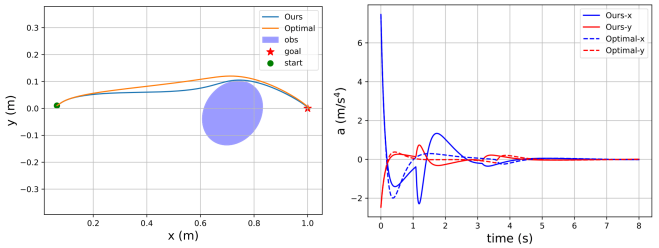


(a) Environment 1: Trajectory (b) Environment 1: Control input

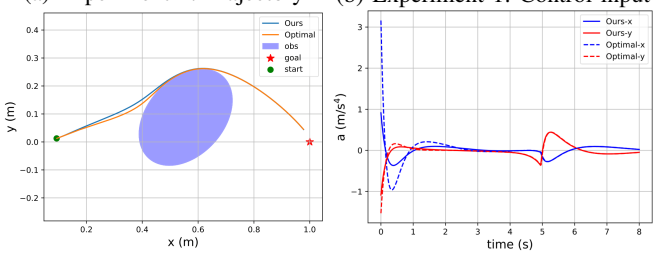


(c) Environment 2: Trajectory (d) Environment 2: Control input

Fig. 3: 2D double integrator (n=4, r=2) avoids a randomly generated obstacle in two different environments. The blue trajectory uses the proposed method, and the orange trajectory is the optimal performance reference that was generated by learning specifically for that environment. The corresponding control inputs are shown on the right side.



(a) Experiment 1: Trajectory (b) Experiment 1: Control input



(c) Experiment 2: Trajectory (d) Experiment 2: Control input

Fig. 4: 2D quadruple integrator (n=8, r=4) avoids a randomly generated obstacle in two different environments. The blue trajectory uses the proposed method, and the orange trajectory is the optimal performance reference that was generated by learning specifically for that environment. The corresponding control inputs are shown on the right side.

represented by ellipses:

$$\begin{aligned} h(y) &= (y - y_c)^\top Q(y - y_c) - 1, \\ y &= Cx, Q = R(\theta)^\top \Lambda R(\theta), \end{aligned} \tag{16}$$

where y is the measurement variable, i.e., the position in Cartesian space. $R(\theta)$ is the rotation matrix defined by the orientation θ of the ellipse. y_c is the center of the obstacle. Λ is a diagonal matrix which represents the size of the obstacle. We define the environment information as $e = [y_c, diag(\Lambda)^\top, \theta]^\top \in \mathbb{R}^5$. For different environments, we randomly sample e from a Gaussian distribution $\mathcal{P}_e$.

We use a linear quadratic regulator as the performance controller and a differentiable ECBF-QP as the safety filter. For the α-net in the differentiable ECBF-QP, we use a feedforward neural network with two hidden layers. Each hidden layer size is 100. Based on Algorithm 1, we train each system with 100 iterations. In each iteration, we sample 30 environments, and for each rollout, the simulation time is 8s. Futhermore, the initial condition of each system is selected randomly within a predefined region. We use the same loss function for both systems, which is the sum of the distance between each point and the goal location.

$$\mathcal{L}_{\text{perf}}(\tau, e, x_0) = \sum_{t=0}^{T} \|x_t - x_{\text{goal}}\|^2. \tag{17}$$

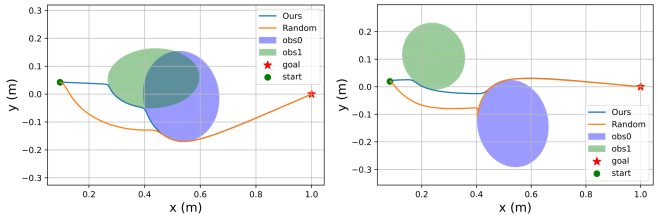
The numerical values of this loss will serve as means to compare performance of different controllers. Moreover, we train a α-net only conditioned on a specific environment to serve as the optimal solution for that specific environment.

B. *Double Integrator Experiment*

Two representative validation results for 2D double integrator avoiding an obstacle with the proposed approach are shown in Fig. 3, including trajectory and control input. The start point is chosen randomly, the goal location is at $(1.0, 0.0)$, and the obstacle is colored as blue. Moreover, the proposed method is compared with the optimal performance solution, which is obtained by finding the best K_α based on the current environment, i.e., the environment in Fig. 3. In Environment 1, the losses defined in (17) for our method and optimal performance solution are 26.88 and 26.41. In Environment 2, the losses are 25.88 and 25.86 for ours and optimal performance solution, respectively. The simulation result shows that the performance of our proposed method is close to the optimal performance solution in terms of the loss function in (17). Also, our control inputs (solid lines) is similar to the optimal ones (dashed lines) as shown in Fig. 3 (b) and (d).

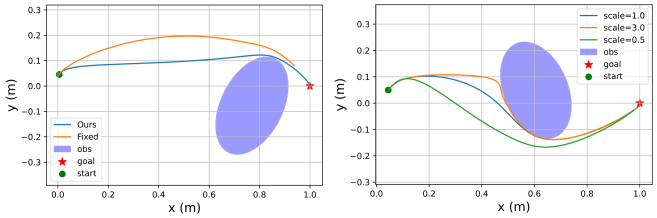
C. *Quadruple Integrator Experiment*

In Fig. 4, we show that our approach can cope with a system with relative-degree four for generalization to novel environments. In both environments, the losses for the optimal performance solution is 26.71 and 35.03, whereas the losses for our method is 28.35 and 35.36, respectively. We also observe that the control inputs of the proposed method is similar to the optimal performance solution as shown in Fig. 4 (b) and (d). The results imply that our approach can determine a proper K_α given the environment information without any manually tuning process for high relative-degree systems.



(a) Experiment 1 (b) Experiment 2

Fig. 5: 2D double integrator is able to generalize to novel environments with two randomly generated obstacles, which are not experienced during training. The blue trajectory utilizes the proposed framework, and the orange trajectory uses a random valid K_α.



(a) Ablation study 1 (b) Ablation study 2

Fig. 6: Ablation study: (a) obstacle information is necessary for environment generation. The blue curve uses the proposed method, and the orange curve is the baseline with fixed obstacle information; (b) a larger or smaller valid K_α does not lead to a better performance. Different colors represent different scales.

D. Multiple Obstacles Experiment

To further validate the generalization ability, we extend the simulation setup of 2D double integrator from one obstacle to multiple obstacles. We randomly generate two obstacles and formulate one ECBF constraint for each object accordingly in (15). For each constraint, we use the same learned α-net. In this scenario, the proposed method needs to be able to generalize to more complex environments. The simulation results of two examples are shown in Fig. 5. In Experiment 1, the losses for our method and random valid K_α are 28.11 and 32.50, respectively, and in Experiment 2, the corresponding losses are 23.96 and 35.51. It shows that our approach successfully generalizes to multiple obstacle scenarios and outperforms the baseline by a large margin.

E. Ablation Study

We conduct two ablation studies using the 2D quadruple integrator to validate that our proposed design is necessary for generalizing to novel environments.

1) Is the obstacle information indeed useful?: The first ablation study is to evaluate whether the obstacle information is necessary. We train an α-net with only one fixed obstacle during training as a baseline and then test it with novel environments. The resulting trajectories are shown in Fig. 6(a). Our proposed method has a loss of 30.26, while the loss for baseline is 36.79. This indicates that the obstacle information is necessary as an input to our neural network.

2) Is a larger or smaller valid K_α better?: In Fig. 6(b), we investigate whether a larger or smaller valid K_α can achieve a better performance with respect to our loss function. We scale K_α by multiplying the learned eigenvalues $\{p_i\}_{i=1,\dots,r}$ with coefficients 3.0 and 0.5. The losses for our method, 3.0 scale, and 0.5 scale are 33.90, 65.90, and 37.85, respectively. The resulting trajectories demonstrate that the proposed method outperforms the cases with the scales.

VI. Discussion

We next provide analysis of the proposed framework and a discussion on the limits and thoughts on future work.

We carry out a performance benchmark in three different scenarios: 2D double integrator with one obstacle (Double Integrator), 2D quadruple integrator with one obstacle (Quadruple Integrator), and 2D double integrator with two obstacles (Multiple Obstacles). We compare our method with random valid K_α using 200 experiments for each scenario. The mean and standard deviation of the losses are summarized in Tab. I. We first conduct 4 subtasks, and each of them consists of 50 experiments. The mean and standard deviation are computed based on these subtasks. The benchmark for Double Integrator and Quadruple Integrator further validates that the proposed framework is useful for generalization to novel environments. Moreover, when we extend the learned α-net to multiple obstacles, our method shows better results.

Scenario	Random	Ours
Double Integrator	26.8832±0.7259	**26.6774±0.5998**
Quadruple Integrator	34.2979±1.1136	**32.2187±0.8840**
Multiple Obstacles	62.2640±2.2653	**40.6258±3.6791**

TABLE I: Benchmark of our proposed framework in three different scenarios: double integrator, quadruple integrator, and double integrator with two obstacles.

In terms of the overall controller design, we assume that a high-level controller is given and fixed in this work, which could be a valid assumption in many applications. Note that the overall performance is determined by both the high-level performance controller and the CBF-QP safety filter.

VII. Conclusion

In this paper, we presented a learning differentiable safety-critical-control framework using control barrier functions for generalization to novel environments, which uses a learning-based method to choose an environment-dependent K_α in exponential control barrier function. Moreover, based on the ECBF formulation, the proposed method ensures the forward invariance of the safe set. We numerically verified the proposed method with 2D double and quadruple integrator systems in novel environments. Our framework can be easily generalized to different shapes of obstacles and nonlinear dynamics. Also, different representation of environment information such as images and point cloud can be used. There are several interesting future directions. For instance, an integrated end-to-end framework can be designed for training the performance controller and ECBF-QP. Another

promising future direction is to test our approach in more general scenarios.

ACKNOWLEDGEMENT

We would like to thank Ayush Agrawal for his helpful discussions.

REFERENCES

[1] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," *Advances in Neural Information Processing Systems*, vol. 32, pp. 9562–9574, 2019.

[2] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 3420–3431.

[3] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014.

[4] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, pp. 3861–3876, 2016.

[5] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable mpc for end-to-end planning and control," *Advances in Neural Information Processing Systems*, vol. 31, pp. 8289–8300, 2018.

[6] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 136–145.

[7] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 2242–2253.

[8] F. S. Barbosa, L. Lindemann, D. V. Dimarogonas, and J. Tumova, "Provably safe control of lagrangian systems in obstacle-scattered environments," in *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 2056–2061.

[9] J. Choi, F. Castañeda, C. Tomlin, and K. Sreenath, "Reinforcement Learning for Safety-Critical Control under Model Uncertainty, using Control Lyapunov Functions and Control Barrier Functions," in *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[10] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A lyapunov-based approach to safe reinforcement learning," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 8103–8112.

[11] I. Cizelj, X. C. D. Ding, M. Lahijanian, A. Pinto, and C. Belta, "Probabilistically safe vehicle control in a hostile environment," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 11 803–11 808, 2011.

[12] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe nonlinear control using robust neural lyapunov-barrier functions," in *5th Annual Conference on Robot Learning*, 2021.

[13] P. L. Donti, M. Roderick, M. Fazlyab, and J. Z. Kolter, "Enforcing robust control guarantees within neural network policies," in *International Conference on Learning Representations*, 2020.

[14] J. Drgona, A. Tuor, and D. Vrabie, "Learning constrained adaptive differentiable predictive control policies with guarantees," *arXiv preprint arXiv:2004.11184*, 2020.

[15] Y. Emam, P. Glotfelter, Z. Kira, and M. Egerstedt, "Safe model-based reinforcement learning using robust control barrier functions," *arXiv preprint arXiv:2110.05415*, 2021.

[16] S.-C. Hsu, X. Xu, and A. D. Ames, "Control barrier function based quadratic programs with application to bipedal robotic walking," in *2015 American Control Conference (ACC)*. IEEE, 2015, pp. 4542–4548.

[17] W. Jin, S. Mou, and G. Pappas, "Safe pontryagin differentiable programming," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[18] E. C. Kerrigan and J. M. Maciejowski, "Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control," in *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No. 00CH37187)*, vol. 5. IEEE, 2000, pp. 4951–4956.

[19] H. K. Khalil, "Nonlinear systems," 2002.

[20] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 657–10 665.

[21] A. Majumdar, A. Farid, and A. Sonar, "Pac-bayes control: learning policies that provably generalize to novel environments," *The International Journal of Robotics Research*, vol. 40, no. 2-3, pp. 574–593, 2021.

[22] Q. Nguyen and K. Sreenath, "Optimal robust control for bipedal robots through control lyapunov function based quadratic programs." in *Robotics: Science and Systems*. Rome, Italy, 2015, pp. 1–9.

[23] ——, "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 322–328.

[24] H. Parwana and D. Panagou, "Recursive feasibility guided optimal parameter adaptation of differential convex optimization policies for safety-critical systems," *arXiv preprint arXiv:2109.10949*, 2021.

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

[26] U. Rosolia and A. D. Ames, "Multi-rate control design leveraging control barrier functions and model predictive control policies," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 1007–1012, 2020.

[27] E. Scukins and P. Ögren, "Using reinforcement learning to create control barrier functions for explicit risk mitigation in adversarial environments," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE Robotics and Automation Society, 2021.

[28] L. Song, N. Wan, A. Gahlawat, C. Tao, N. Hovakimyan, and E. A. Theodorou, "Generalization of safe optimal control actions on networked multi-agent systems," *arXiv preprint arXiv:2109.09909*, 2021.

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

[30] R. Tian, L. Sun, A. Bajcsy, M. Tomizuka, and A. D. Dragan, "Safety assurances for human-robot interaction via confidence-aware game-theoretic human models," *arXiv preprint arXiv:2109.14700*, 2021.

[31] L. Wang, A. D. Ames, and M. Egerstedt, "Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 2659–2664.

[32] W. Xiao and C. Belta, "High order control barrier functions," *IEEE Transactions on Automatic Control*, 2021.

[33] W. Xiao, C. A. Belta, and C. G. Cassandras, "Feasibility-guided learning for constrained optimal control problems," in *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 1896–1901.

[34] J. Zeng, B. Zhang, Z. Li, and K. Sreenath, "Safety-critical control using optimal-decay control barrier function with guaranteed point-wise feasibility," in *2021 American Control Conference (ACC)*, 2021, pp. 3856–3863.

[35] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 3882–3889.

[36] J. Zhang, B. Cheung, C. Finn, S. Levine, and D. Jayaraman, "Cautious adaptation for reinforcement learning in safety-critical settings," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 055–11 065.

[37] W. Zhao, T. He, and C. Liu, "Model-free safe control for zero-violation reinforcement learning," in *5th Annual Conference on Robot Learning*, 2021.