2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Neural Control Barrier Functions for Safe Navigation

Marvin Harms, Mihir Kulkarni, Nikhil Khedekar, Martin Jacquet, Kostas Alexis

***Abstract*— Autonomous robot navigation can be particularly demanding, especially when the surrounding environment is not known and safety of the robot is crucial. This work relates to the synthesis of Control Barrier Functions (CBFs) through data for safe navigation in unknown environments. A novel methodology to jointly learn CBFs and corresponding safe controllers, in simulation, inspired by the State Dependent Riccati Equation (SDRE) is proposed. The CBF is used to obtain admissible commands from any nominal, possibly unsafe controller. An approach to apply the CBF inside a safety filter without the need for a consistent map or position estimate is developed. Subsequently, the resulting reactive safety filter is deployed on a multirotor platform integrating a LiDAR sensor both in simulation and real-world experiments.**

I. Introduction

A host of robotics applications involve the deployment of safety-critical systems in unknown environments. Examples include ground and flying robots deployed for subterranean exploration, search-and-rescue, and forest monitoring [1, 2]. Such deployments remain challenging, especially as onboard localization and mapping is prone to latency, noise and drift, possibly resulting in estimation failure and thus potentially collisions. Responding to this fact, a set of methods have enabled data-driven approaches for collision-free flight without the need of a map. Indicative works relate to privileged learning from an expert policy [3], learned collision prediction with motion primitives [4, 5] or Model Predictive Control [6], and Reinforcement Learning (RL) using depth maps [7]. However, these methods typically do not provide formal guarantees for collision avoidance.

Another avenue for safe navigation, particularly focused on provable assurances, is the use of safety filters [8], which are last resort policies processing controls from any nominal controller in order to render them safe w.r.t. given criteria such as collision avoidance. To that end, predictive safety filters [9] provide a solution for safe navigation, but at a large computational cost. Alternatively, safety filters based on Control Barrier Functions (CBFs) are computationally cheap and have recently received significant attention [10, 11].

Nevertheless, despite the recent advances [12, 13], finding a permissive CBF for constrained high-order systems remains an open challenge. Machine learning has thus also been leveraged to learn CBFs for such systems from data [8, 14]. Such techniques allow to formalize a CBF as safety w.r.t. collision avoidance in unknown environments [15, 16]. However, learning CBFs requires satisfying forward invariance conditions, which is particularly challenging. Previous works either lifted this requirement by considering oversimplistic first-order systems [15], or employed analytic CBFs which in turn struggle to scale to cluttered environments [16]. Some approaches also necessarily employ some prior control policy, along which the CBF is learned [17].

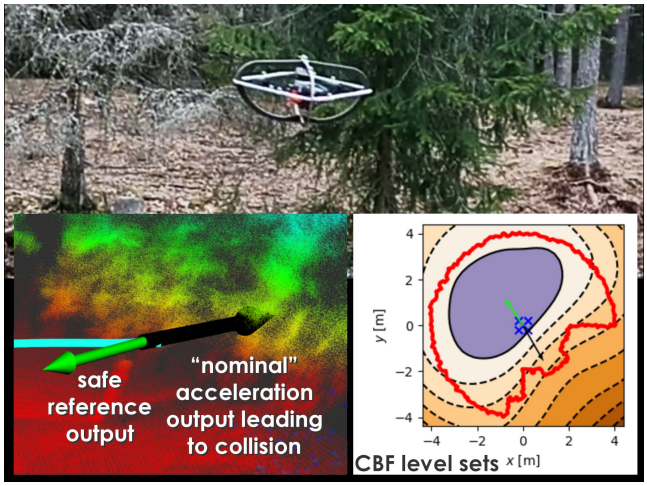

Fig. 1: Instance of the presented experiments for safe navigation through direct learning of neural control barrier functions commanding accelerations. Video: https://www.youtube.com/watch?v=1id0C6jiFEg

Motivated to overcome these limitations, in this work we propose a novel method for learning CBFs for safe navigation of aerial robots integrating range sensing. The contributions are threefold: First, we introduce a novel formulation to jointly learn CBFs and safe controllers for collision avoidance in unknown environments for high-order systems with input constraints, using an adaptation of the State Dependent Riccati Equation (SDRE). The proposed approach is trained by sampling the observation space of range sensors and the vehicle state space entirely in simulation. Second, we propose a policy to reactively apply the CBF in a safety filter without requiring approximation of the often unknown observation dynamics. This policy operates in a local frame, thus relaxing the need for a consistent map. Finally, the proposed safety filter is both statistically evaluated in simulations and experimentally verified on an autonomous quadrotor flying in novel environments (e.g., Fig. 1). The results demonstrate that the learned neural CBFs satisfy safety requirements, in terms of formulation guarantees and practical results, and enable practically safe autonomous navigation exploiting only instantaneous range observations and velocity estimates.

In the remaining paper, Section II presents related work, and III preliminaries. The proposed neural CBFs are detailed in Section IV and their use for navigation in Section V. Evaluations are in Section VI and conclusions in Section VII.

Autonomous Robots Lab, Norwegian University of Science and Technology (NTNU), Trondheim, Norway, marvin.c.harms@ntnu.no

This work was partially supported by the European Commission Horizon Europe project DIGIFOREST (EC 101070405).

II. Related Work

This contribution relates to the body of work in CBFs, their application to safe robot navigation, and CBFs learned from data. CBFs were introduced to ensure safety for critical systems [10, 11] and are becoming increasingly prevalent in robotics [12]. Focusing on navigation, [18] compares artificial potential fields and CBFs. Using DBSCAN on LiDAR maps to cluster obstacles and thus derive state constraints, [19] utilizes CBFs to demonstrate safety for a ground robot. Considering known obstacles locations, [20] develops safe control with multiple Lyapunov-based CBFs. The method in [13] assumes a volumetric map combined with CBFs for safe aerial robot navigation demonstrated in simulation, while [21] enables safe teloperation experimentally.

Focusing on the benefits of data-driven methods, learning-based approaches exploiting CBF certificates have expanded over the recent years [8, 14]. Machine learning can enable the use of CBFs for complex systems for which deriving analytic safety certificates is challenging. A first approach to mention is Safe RL, where a CBF is jointly learned during training to guide the policy to implicitly encode the underlying safety mechanism [22]. In [23], RL is proposed to learn the residuals in the dynamics of a CBF (and a Control Lyapunov Function (CLF)) with a neural network. Aside RL, supervised learning is also employed to derive safety filters based on Quadratic Programming (QP) [17, 24].

Applying learning-based CBFs to safe navigation, the CBF is used to encode safety w.r.t. the environment. The main challenge is to synthesize a CBF based on partial observations of the environment obtained through exteroceptive sensors. An approach is to extract features from the observations (e.g., the distance to the closest obstacle [16, 25]) or geometric information about the obstacles in state space [26]. However, these approaches struggle to scale to high-dimensional inputs (e.g., images, LiDAR scans) and therefore tend to not generalize well to unknown environments. Leveraging deep learning to lift this limitation, [27] uses a Generative Adversarial Network to detect the 3D positions of obstacles in images and infer their time derivatives used to compute a geometric CBF. In [28] a NeRF predicts future images which will result from a given action, allowing to approximate the gradient of the CBF. Yet, this is computationally demanding since the NeRF is queried for each sampled action. Alternatively, the authors in [15] learn a CBF within the observation space with the CBF approximated by a neural network trained by sampling states and observations. However, this requires approximating the temporal gradients of observations which is particularly challenging in non-trivial environments. Finally, [29, 30] learn a Signed Distance Function then used as a CBF.

III. Preliminaries

This section outlines relevant background knowledge.

A. Symbols and Abbreviations

$\mathbf{x} \in \mathcal{X}$, $\mathbf{u} \in \mathcal{U}$	state ($\mathbf{x}$) and input ($\mathbf{u}$) vector
$\mathbf{o} \in \mathcal{O}$	observation vector
$\mathcal{L}_f h, \mathcal{L}_B h\mathbf{u}$	Lie derivatives of h along f, $B\mathbf{u}$
$\frac{d}{dt}V$, $\nabla_{\mathbf{x}} V$	time derivative of V, gradient of V w.r.t. $\mathbf{x}$
$\mathbf{a} \cdot \mathbf{b}$	dot product of $\mathbf{a}$ and $\mathbf{b}$
$\mathbb{I}$	identity matrix
$\|\cdot\|$	euclidean norm
$\text{eig}(\cdot)$	eigenvalues
$\text{proj}_{\mathcal{U}}(\cdot)$	projection onto $\mathcal{U}$
ReLU, ELU	rectified linear unit, exponential linear unit

B. System Definition

We consider the control affine system:

$$\frac{d}{dt}\mathbf{x} = \underbrace{A(\mathbf{x})\mathbf{x}}_{f(\mathbf{x})} + B(\mathbf{x})\mathbf{u} \tag{1}$$

where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ and $\mathbf{u} \in \mathcal{U} \subseteq \mathbb{R}^m$. Further, we divide the state space $\mathcal{X}$ into the compact, open subset $\mathcal{X}_{\text{free}}$ and the closed subset $\mathcal{X}_{\text{obs}}$ such that $\mathcal{X} = \mathcal{X}_{\text{free}} \cup \mathcal{X}_{\text{obs}}$, where $\mathcal{X}_{\text{obs}}$ denotes the set of states we seek to avoid (the constraint set). Further we assume that the origin is safe, i.e. $\mathbf{x} = 0 \in \mathcal{X}_{\text{safe}}$, where $\mathcal{X}_{\text{safe}} \subseteq \mathcal{X}_{\text{free}}$ is a *control invariant* set.

Definition 1 (Control Invariant Set)**.** A set $\mathcal{X}_{\text{safe}}$ is a forward control invariant set for the system (1), if for any $\mathbf{x}(t_0) \in \mathcal{X}_{\text{safe}}$ there exists an input trajectory $\mathbf{u}(t) \in \mathcal{U}$ such that $\mathbf{x}(t) \in \mathcal{X}_{\text{safe}} \quad \forall t \geq t_0$ under the system (1).

The use of control invariant sets is a central concept in safe control as it allows to bound areas in $\mathcal{X}$ where future constraint satisfaction (avoiding $\mathcal{X}_{\text{obs}}$) can be guaranteed.

C. Control Barrier Functions

Following the definition of a safe set, we now define corresponding control barrier functions which can later be used to synthesize safe control actions.

Definition 2 (Control Barrier Function [11])**.** Let $\mathcal{X}_{\text{safe}}$ be the 0-superlevel set of a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ with the property that $\mathcal{X}_{\text{safe}} = \{\mathbf{x} \in \mathcal{X} | h(\mathbf{x}) \geq 0\}$ and $\{\mathbf{x} \in \mathcal{X} | \frac{dh}{d\mathbf{x}}(\mathbf{x}) = 0\} \cap \{\mathbf{x} \in \mathcal{X} | h(\mathbf{x}) = 0\} = \emptyset$. Then, h is a control barrier function if there exists an extended class $\mathcal{K}_\infty$ function $\alpha(\cdot)$ such that for the system (1):

$$h(\mathbf{x}) < 0 \quad \forall \mathbf{x} \in \mathcal{X}_{\text{obs}} \tag{2a}$$

$$\sup_{\mathbf{u} \in \mathcal{U}} [\mathcal{L}_f h(\mathbf{x}) + \mathcal{L}_B h(\mathbf{x})\mathbf{u}] \geq -\alpha(h(\mathbf{x})) \tag{2b}$$

for all $\mathbf{x} \in \mathcal{X}_{\text{safe}}$. Further, an input $\mathbf{u}$ is considered safe with respect to a valid CBF, if it satisfies (2b).

By definition, a CBF creates a safety certificate such that by control invariance, all state trajectories starting within $\mathcal{X}_{\text{safe}}$ can be kept inside $\mathcal{X}_{\text{safe}}$ for all future times by choosing appropriate input trajectories. Asymptotic stability of $\mathcal{X}_{\text{safe}}$ can be achieved by extending requirement (2b) to hold on $\mathcal{X}$. Further, CBFs also provide the informative condition (2b) on a candidate input at any given state, which can be utilized to synthesize safe control actions form unsafe ones. In [10], a simple yet effective way to employ condition (2b) in a QP to create a computationally cheap safety filter is proposed,

yielding the safe input $\mathbf{u}^*$ from an arbitrary (possibly unsafe) reference input $\mathbf{u}_{\text{ref}}$:

$$\begin{aligned} \mathbf{u}^* = \quad & \underset{\mathbf{u}\in\mathcal{U}}{\text{argmin}} \quad ||\mathbf{u}-\mathbf{u}_{\text{ref}}||^2 \\ & \text{s.t.} \quad \mathcal{L}_f h(\mathbf{x}) + \mathcal{L}_B h(\mathbf{x})\mathbf{u} \geq -\alpha(h(\mathbf{x})) \end{aligned} \tag{3}$$

While not being optimal in terms of closed-loop performance, the reactive policy (3) allows to synthesize safe control actions from unsafe ones. Therefore CBFs offer an efficient alternative to predictive safety filters [9], especially for high-order systems with complex constraints, for which the computational complexity can become large.

IV. Neural Navigation CBFs

The above motivates the use of CBFs to navigation systems for autonomous robots. Aiming for a safety filter applied on a multirotor aerial robot with odometry state $\mathbf{x}$, action vector $\mathbf{u}$ subject to constraints $||\mathbf{u}|| \leq u_{\max}$, and a local instantaneous observation of the environment $\mathbf{o}$ (e.g., a scan from an onboard LiDAR), we proceed with a generic formulation of the proposed neural navigation CBFs before specializing to the targeted application in Section V. To that end, dynamics are considered to obey the form in (1) as applicable for the case of translational dynamics of attitude-control multirotors. Accordingly, our goal is to enable collision-free navigation of the system (1) in arbitrary, unknown, static environments. Subsequently, we develop an approach to train and apply CBFs for safe navigation.

A. Constructive Safety from Observation CBFs

In contrast to the traditional formulation of CBFs, where h is a function only of $\mathbf{x}$, we define the CBF as a function of both the exteroceptive observations $\mathbf{o} \in \mathcal{O} \subset \mathbb{R}^o$ and the state $\mathbf{x}$ as $h(\mathbf{o},\mathbf{x})$ [15]. This change is necessary to enable safe navigation in an unknown environment, where the local observations from an onboard range sensor are used to create a CBF describing a safe set in a local, robot-centered frame. This requires some additional care when formulating the invariance condition (2b). Specifically, we assume the following observation model, where our state evolves according to (1) and the observation $\mathbf{o}$ is an unknown function (since the environment is unknown) ζ of the state $\mathbf{x}$ as $\mathbf{o} = \zeta(\mathbf{x})$. Taking the time derivative of $h(\mathbf{o},\mathbf{x})$ results:

$$\frac{d}{dt}h(\mathbf{o},\mathbf{x}) = \underbrace{\nabla_{\mathbf{o}} h(\mathbf{o},\mathbf{x}) \cdot \nabla_{\mathbf{x}}\zeta(\mathbf{x})\frac{d}{dt}\mathbf{x}}_{\text{observation dynamics}} + \underbrace{\nabla_{\mathbf{x}} h(\mathbf{o},\mathbf{x}) \cdot \frac{d}{dt}\mathbf{x}}_{\text{state dynamics}}, \tag{4}$$

where we grouped the terms into the observation dynamics and the state dynamics. Computing the time derivative $\frac{d}{dt}h(\mathbf{o},\mathbf{x})$ requires knowledge of the gradient $\nabla_{\mathbf{x}}\zeta$. Conversely, neither is ζ known, nor does it necessarily have a computable gradient $\nabla_{\mathbf{x}}\zeta$ due to possible discontinuities in the environment. To enable safe navigation with observation CBFs with the safety filter QP (3), we resort to a switching set approach to avoid approximating the observation dynamics. We treat successive observations $\mathbf{o}_{i-1}$, $\mathbf{o}_i$ as discrete-time parameters of the function h, describing separate safe sets defined in the respective local frames they were obtained in. Using the fact that the union of control invariant sets is again a control invariant set, it suffices to enforce invariance of the state $\mathbf{x}$ w.r.t. either valid CBF $h(\mathbf{o}_{i-1},\cdot)$ or $h(\mathbf{o}_i,\cdot)$. Note that $\mathbf{x}$ needs to be expressed relative to the frame the corresponding observation originates from by using the relative transform $\xi(\mathbf{x})$ (Fig. 2). Using a constructive approach, the switching update following a new observation $\mathbf{o}_i$ at time t_i is proposed:

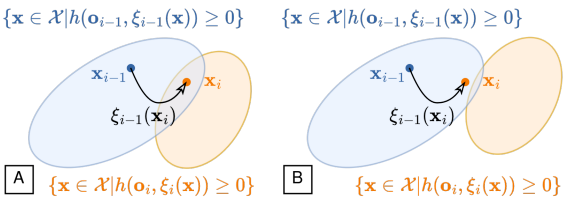

Fig. 2: Fig. A (left): $\mathbf{x}_i$ is in the new safe set an the new certificate (using new latest observation) can be applied. Fig. B (right): $\mathbf{x}_i$ is not in the new safe set and invariance of a previous safe set is enforced.

$$\begin{aligned} &\mathbf{o}_{\text{safe}} \leftarrow \mathbf{o}_i \quad \text{if } h(\mathbf{o}_i,\xi_i(\mathbf{x}(t_i)))) \geq 0 \\ &\xi_{\text{safe}} \leftarrow \xi_i \quad \text{if } h(\mathbf{o}_i,\xi_i(\mathbf{x}(t_i)))) \geq 0 \\ &h(\mathbf{o},\mathbf{x}) = h(\mathbf{o}_{\text{safe}},\xi_{\text{safe}}(\mathbf{x})). \end{aligned} \tag{5}$$

The above update law essentially only updates the certificate parameters ($\mathbf{o}_{\text{safe}}$ and ξ_{safe}), if the state is in the resulting new safe set. Otherwise, the previous parameters are applied, where $\xi_{\text{safe}}(\mathbf{x})$ can be determined by dead-reckoning. The switching set approach is visualized in Fig. 2. It is assumed that dead-reckoning occurs infrequently for short amounts of time to avoid accumulating odometry errors in the relative transform ξ_{safe}. Further, we avoid "locked-in" situations where $\{\mathbf{x}\in\mathcal{X}|h(\mathbf{o},\mathbf{x})\geq 0\} \cap \{\mathbf{x}\in\mathcal{X}|h(\mathbf{o}_i,\xi_i(\mathbf{x}(t_i)))) \geq 0\} = \emptyset$ by parametrization of h introduced in subsection IV-C. We now show that forward invariance of this switching safe set can be achieved.

Assumption 1. For an extended class $\mathcal{K}_\infty$ function $\alpha(\cdot)$, the function $h(\mathbf{o}_i,\cdot)$ is a valid CBF for the system (1) according to Definition 2 for any $\mathbf{o}_i \in \mathcal{O}$.

Theorem 1. Under assumption 1, the switching update (5) applied at discrete time steps $t_i \in \{t_1, t_2, ..., t_e\}, e \in \mathbb{N}$ with $t_i \leq t_{i+1}$ together with any control law satisfying:

$$\nabla_{\mathbf{x}} h(\mathbf{o}_{\text{safe}},\xi_{\text{safe}}(\mathbf{x})) \cdot \frac{d}{dt}\mathbf{x} \geq -\alpha(h(\mathbf{o}_{\text{safe}},\xi_{\text{safe}}(\mathbf{x}))) \tag{6}$$

$\forall t \neq t_i$ with $\mathbf{o}_{\text{safe}},\xi_{\text{safe}}$ from (5) guarantees forward invariance of the switching safe set $\mathcal{X}_{\text{safe}} = \{\mathbf{x} \in \mathcal{X}|h(\mathbf{o}_{\text{safe}},\xi_{\text{safe}}(\mathbf{x})) \geq 0\}$. That is $\mathbf{x}(t_0) \in \mathcal{X}_{\text{safe}}$ implies $\mathbf{x}(t) \in \mathcal{X}_{\text{safe}} \quad \forall t \geq t_0$.

Proof. First, we note that (6) guarantees forward invariance and asymptotic stability of the set $\mathcal{X}_{\text{safe}}$ for the time interval $[t_a, t_b]$ where $\forall i \quad t_i \notin [t_a, t_b]$ ([11], theorem 2). Further, the switching update (5) applied at time step t_i^- (before the switching update), where $\mathbf{x}(t_i^-) \in \mathcal{X}_{\text{safe}}$, always produces $\mathbf{o}_{\text{safe}},\xi_{\text{safe}}$ such that $\mathbf{x}(t_i^+) \in \mathcal{X}_{\text{safe}}$ at t_i^+ (after the switching update). By choosing $t_b = t_i^+$ and $\mathbf{x}(t_a) \in \mathcal{X}_{\text{safe}}$ we thus have $\mathbf{x}(t) \in \mathcal{X}_{\text{safe}} \forall t \in [t_a, t_b]$. By recursively applying this logic with time intervals $[t_0, t_1], (t_1, t_2], ..., (t_{e-1}, t_e], (t_e, \infty)$ with $t_0 \leq t_1$, we have $\mathbf{x}(t) \in \mathcal{X}_{\text{safe}} \forall t \geq t_0$. □

The combined update law (5), (6) allows to consider newer and local CBFs during navigation tasks while forward invariance of the (switching) safe set is always maintained without approximation of $\nabla_{\mathbf{x}}\zeta$.

B. Learning Observation CBFs

To learn CBFs for a general navigation task in unknown environments, the proposed method relies on sampling of the state space $\mathcal{X}$, and additional information on whether the sampled states lie within $\mathcal{X}_{\text{obs}}$ or $\mathcal{X}_{\text{free}}$. This label can be easily computed at low cost and does not require constraints to be evaluated in closed form. Specifically, we are interested in learning the CBF to synthesize safe control actions from arbitrary control policies without *a priori* controllers and expert demonstrations, as is typically done in the literature. As in [15], the safety controller is only learned to ensure the existence of a suitable control action during training and is discarded after training. It thus serves as a "plug 'n' play" safety layer around any nominal navigation policy. Further, we assume that the origin $\mathbf{x} = 0$ is part of the safe set $\mathcal{X}_{\text{safe}}$, i.e., that the origin can be rendered invariant by the parameterization discussed in subsection IV-C.

Given these assumptions, the requirements posed on the learned CBF candidate function $h(\mathbf{o}, \mathbf{x})$ and safe control action $\mathbf{u}(\mathbf{o}, \mathbf{x})$ are:

$$h(\mathbf{o}, \mathbf{x}) \leq 0 \qquad \forall \mathbf{o} \in \mathcal{O}, \quad \forall \mathbf{x} \in \mathcal{X}_{\text{obs}} \tag{7a}$$

$$\mathbf{u}(\mathbf{o}, \mathbf{x}) \in \mathcal{U} \qquad \forall \mathbf{o} \in \mathcal{O}, \quad \forall \mathbf{x} \in \mathcal{X} \tag{7b}$$

$$\frac{d}{dt} h(\mathbf{o}, \mathbf{x}) \geq -\alpha(h(\mathbf{o}, \mathbf{x})) \quad \forall \mathbf{o} \in \mathcal{O}, \quad \forall \mathbf{x} \in \mathcal{X}. \tag{7c}$$

In the following, these requirements are now framed as a single multi-objective optimization problem, where we train the CBF $h(\mathbf{o}, \mathbf{x})$ and the safety controller $\mathbf{u}(\mathbf{o}, \mathbf{x})$ parameterized as neural networks to satisfy these requirements. The proposed network architecture is shown in Fig. 3. In a series of networks, the observations $\mathbf{o}$ and states $\mathbf{x}$ are first passed through a latent network generating a joint latent variable $\mathbf{z}(\mathbf{o}, \mathbf{x})$. This latent variable then gets passed through a controller network and a CBF network to generate $\mathbf{u}(\mathbf{o}, \mathbf{x})$ and $h(\mathbf{o}, \mathbf{x})$, respectively. A static class $\mathcal{K}_\infty$ function α is then applied to $h(\mathbf{o}, \mathbf{x})$ to evaluate condition (2b). During training, the network outputs are passed to a function enforcing requirements (7) and the network weights are updated using stochastic gradient descent. By using the latent representation $\mathbf{z}(\mathbf{o}, \mathbf{x})$, the input to the CBF network (e.g. $\mathbf{z}(\mathbf{o}, \mathbf{x})$) is sensitive to penalties placed violation of (7b) (in contrast to simply using the controller network and CBF network directly on $(\mathbf{o}, \mathbf{x})$), which improves the learning rate. The reason for this is that the input to the controller network and the CBF network is updated by all loss functions.

C. Joint Learning of Controller and CBF

We now propose a formulation to jointly learn safe controllers and the corresponding CBFs using an adaptation of matrix-valued SDRE. For the sake of legibility, we omit any dependence of h on $\mathbf{o}$ in the following derivation, e.g. we replace $h(\mathbf{o}, \mathbf{x})$ with $h(\mathbf{x})$. Our starting point is the Hamilton-Jacobi-Bellman (HJB) equation for a value function $V(\mathbf{x}) \geq 0$ and stage cost $l(\mathbf{x}, \mathbf{u})$:

$$\frac{d}{dt} V(\mathbf{x}) + \sup_{\mathbf{u} \in \mathcal{U}} [l(\mathbf{x}, \mathbf{u})] = 0. \tag{8}$$

The SDRE can now be derived as in [31], assuming a structure of $V(\mathbf{x})$ as $V(\mathbf{x}) = \mathbf{x}^T P(\mathbf{x})\mathbf{x}$ together with a stage cost $l(\mathbf{x}, \mathbf{u}) = \mathbf{x}^T Q(\mathbf{x})\mathbf{x} + \mathbf{u}^T R(\mathbf{x})\mathbf{u}$, for symmetric matrices $P(\mathbf{x})$, $Q(\mathbf{x})$, $R(\mathbf{x})$ as shown in [31]. Inserting these into the HJB equation yields:

$$\nabla_{\mathbf{x}} V(\mathbf{x})^T \cdot \frac{d}{dt}\mathbf{x} + \sup_{\mathbf{u} \in \mathcal{U}} [\mathbf{x}^T Q(\mathbf{x})\mathbf{x} + \mathbf{u}^T R(\mathbf{x})\mathbf{u}] = 0. \tag{9}$$

By inserting the gradient term:

$$\nabla_{\mathbf{x}} V(\mathbf{x}) = \underbrace{[2P(\mathbf{x}) + \mathbf{x}^T(\nabla_{\mathbf{x}} P(\mathbf{x}))]}_{2S(\mathbf{x})}\mathbf{x} \tag{10}$$

and solving for $\mathbf{u}$ by setting the derivative of (9) w.r.t. $\mathbf{u}$ to zero, we arrive at the optimal state feedback law:

$$\mathbf{u}^* = -R^{-1}B^T S\mathbf{x} \tag{11a}$$

$$\mathbf{x}^T[A^T S + S^T A - S^T B R^{-1} B^T S + Q]\mathbf{x} = 0. \tag{11b}$$

This equation is inspired by the SDRE [31], which follows from inserting approximation $\nabla_{\mathbf{x}} V(\mathbf{x}) = 2P(\mathbf{x})\mathbf{x}$ in (9). The SDRE is then given by replacing the non-symmetric matrix $S(\mathbf{x})$ in (11) by the symmetric matrix $P(\mathbf{x})$, allowing standard Riccati solvers can be used to compute $P(\mathbf{x})$. This approximation leads to the lack of global stability properties for SDRE control laws as the additional term $\mathbf{x}^T(\nabla_{\mathbf{x}} P(\mathbf{x}))\mathbf{x}$ affects the behavior of $\frac{d}{dt}V(\mathbf{x})$ far from the origin $\mathbf{x} = 0$ [31]. At this point, we emphasize that the global satisfaction of (11b) implies global stability of the equilibrium $\mathbf{x} = 0$ when applying (11a) (since it is a reformulation of the HJB).

We now aim to formulate a scheme that allows us to jointly generate CBFs and corresponding safe controllers from value functions given in the form of (11). Therefore, we must replace the stage cost terms $\mathbf{x}^T Q(\mathbf{x})\mathbf{x} + \mathbf{u}^T R(\mathbf{x})\mathbf{u}$ in the optimal control formulation by their CBF counterparts, e.g. the extended class $\mathcal{K}_\infty$ function α. We now define a candidate CBF using the value function $V(\mathbf{x})$ as:

$$h(\mathbf{x}) = 1 - V(\mathbf{x}) = 1 - \mathbf{x}^T P(\mathbf{x})\mathbf{x}. \tag{12}$$

For any $P(\mathbf{x})$ which satisfies (11) globally on $\mathcal{X}$, $h(\mathbf{x})$ indeed has invariant superlevel sets (since $V(\mathbf{x})$ has invariant sublevel sets) for the controlled system with feedback law as in (11a). However, any solution to (11b) does imply optimality for some $Q(\mathbf{x}), R(\mathbf{x})$, yet the choice of these cost matrices *a priori* to achieve a large safe set remains unclear. Instead, the CBF must satisfy (2b) by definition, which translates to:

$$\frac{d}{dt} h(\mathbf{x}) = -\mathbf{x}^T[A^T S + S^T A - 2S^T B R^{-1} B^T S]\mathbf{x} \geq -\alpha(h(\mathbf{x})). \tag{13}$$

To bring the above equation into a matrix-valued form, we expand the right hand term to a quadratic form as:

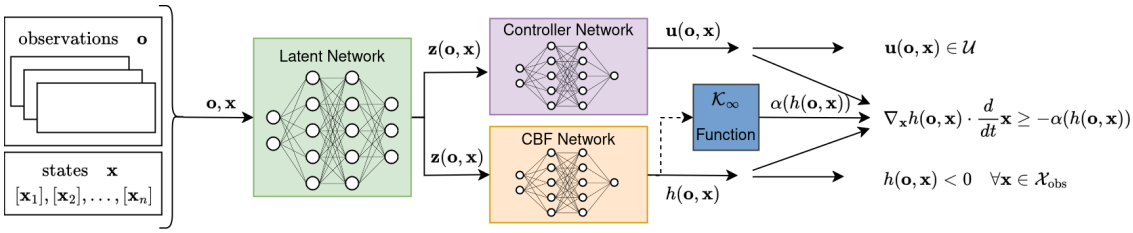

Fig. 3: Proposed network architecture for jointly training a safety controller and CBF. The observations $\mathbf{o}$ and states $\mathbf{x}$ are first passed through a latent network to generate a latent representation $\mathbf{z}(\mathbf{o}, \mathbf{x})$, which is then passed through a controller network and a CBF network to generate the control input $\mathbf{u}(\mathbf{o}, \mathbf{x})$ and CBF value $h(\mathbf{o}, \mathbf{x})$, respectively. The CBF value $h(\mathbf{o}, \mathbf{x})$ is then passed through a (static) class $\mathcal{K}_\infty$ function α. The method relies only on instantaneous observations $\mathbf{o}$ and not a map.

$$\alpha(h(\mathbf{x})) = \alpha(1-\mathbf{x}^T P(\mathbf{x})\mathbf{x}) = \mathbf{x}^T[\frac{\alpha(h(\mathbf{x}))}{h(\mathbf{x})}(\frac{1}{||\mathbf{x}||^2}\mathbb{I}-P)]\mathbf{x} \quad (14)$$

and reinsert into (13) to obtain the necessary condition:

$$\mathbf{x}^T[A^T S+S^T A-2S^T BR^{-1}B^T S-\frac{\alpha(h(\mathbf{x}))}{h(\mathbf{x})}(\frac{1}{||\mathbf{x}||^2}\mathbb{I}-P)]\mathbf{x} \leq 0. \quad (15)$$

This equation can be interpreted as a lower bound of (11), where the terms of the stage cost $S^T BR^{-1}B^T S + Q$ have been replaced with the expansion of α in (14). Here, the quadratic form of α acts as a lower bound on the stage cost. The lower bound is explicitly given by:

$$\mathbf{x}^T[\frac{\alpha(h(\mathbf{x}))}{h(\mathbf{x})}(\frac{1}{||\mathbf{x}||^2}\mathbb{I} - P)]\mathbf{x} \leq \underbrace{\mathbf{x}^T Q\mathbf{x} + \mathbf{u}^{*T} R\mathbf{u}^*}_{l(\mathbf{x},\mathbf{u}^*)}. \quad (16)$$

Note that, unlike the SDRE framework, the bound does allow for $Q(\mathbf{x}) \prec 0$ since the CBF framework only ensures control invariance of all sublevel sets where $h(\mathbf{x}) \leq 0$.

So far, all equations have been expressed in a scalar form, where we only require (15) to hold in scalar form due to the explicit dependence of the matrix P on $\mathbf{x}$. Strictly speaking, only the scalar-valued Eq, (15) is required to hold globally. While being a sufficient condition, the matrix-valued equation:

$$A^T S+S^T A-2S^T BR^{-1}B^T S-\frac{\alpha(h(\mathbf{x}))}{h(\mathbf{x})}(\frac{1}{||\mathbf{x}||^2}\mathbb{I}-P) \leq 0 \quad (17)$$

is a stronger requirement than that the invariance condition (2b) poses. However, we observe that these stronger conditions show favorable properties in terms of training stability and generalizability when jointly learning a control policy and a corresponding CBF. The reason for this is that the scalar condition (2b) itself is not informative enough (as a loss function) to jointly learn safe control policies and CBFs for higher order systems by pure sampling. For higher-order systems, (2b) only provides information along $\nabla_{\mathbf{x}} h(\mathbf{x})$, which leads to local minima in training. Therefore, previous works use first-order systems [15] or resort to using a stable reference controller to guide the training process away from local minima [17]. We overcome this issue by imposing the stricter condition (17). This is generally more restrictive than the scalar form (15) due to the gradient coupling terms in S but provides richer information for learning the matrix P from data. It is important to emphasize that (17) is always satisfied for some α in the trivial case where P and R are independent of $\mathbf{x}$, assuming, (11) is satisfied for some Q and R. We will now make use of (17) to learn $P(\mathbf{x})$ and $R(\mathbf{x})$ from sampled data.

D. Training the Networks

To create a single multi-objective cost function to train the networks depicted in Figure 3, we follow an approach similar to [17], taking the weighted average of the cost functions corresponding to the requirements (7). However, we deviate from [17] by not using any prior safe states and stable controller but insert a condition enforcing satisfaction of (17) to jointly learn CBF and safe controller. The total cost of a minibatch of samples $(\mathbf{o}, \mathbf{x})^j$ of size b is given by:

$$J = \lambda_1 \sum_{(\mathbf{o},\mathbf{x})^j \in (\mathcal{O},\mathcal{X})_{\text{obs}}} \text{ReLU}(h(\mathbf{o}^j, \mathbf{x}^j) + \epsilon_1) \quad + \quad (18a)$$

$$\lambda_2 \sum_{(\mathbf{o},\mathbf{x})^j \in (\mathcal{O},\mathcal{X})_{\text{safe}}} (1 - h(\mathbf{o}^j, \mathbf{x}^j)) \quad + \quad (18b)$$

$$\lambda_3 \sum_{(\mathbf{o},\mathbf{x})^j \in (\mathcal{O},\mathcal{X})} \|\mathbf{u}(\mathbf{o}^j, \mathbf{x}^j) - \text{proj}_{\mathcal{U}}(\mathbf{u}(\mathbf{o}^j, \mathbf{x}^j))\| \quad + \quad (18c)$$

$$\lambda_4 \sum_{(\mathbf{o},\mathbf{x})^j \in (\mathcal{O},\mathcal{X})} \sum_{i=1}^{n} \text{ReLU}((\text{eig}(K(\mathbf{o}^j, \mathbf{x}^j))^i + \epsilon_2)), \quad (18d)$$

where $\lambda_{1-4} \geq 0$ are weighting parameters and $\epsilon_{1,2} \geq 0$ are slack parameters to enforce stricter satisfaction and generalization of the requirements in (7). The first term (18a) penalizes the intersection of the safe set $(\mathcal{O}, \mathcal{X})_{\text{safe}}$ with the obstacle set $(\mathcal{O}, \mathcal{X})_{\text{obs}}$, while (18b) infers expansion of the safe set, and (18c) penalizes violations of requirement 2. In (18d), K corresponds to the symmetric left-hand side of (17) and penalizes violations of (17), enforcing the requirement of control invariance. Further, we add regularization to the term $\frac{d}{dt}P(\mathbf{x})$. The cost J in (18) is then applied to a batch of training samples $(\mathbf{o}, \mathbf{x})^j$ where $\mathbf{o}^j$ is drawn from a distribution of training observations and $\mathbf{x}^j$ is randomly sampled from $\mathcal{X}$.

To improve the satisfaction of requirement (7a), an additional loss:

$$J_{\text{obs}} = \lambda_5 \sum_{(\mathbf{o},\mathbf{x})^j \in \delta(\mathcal{O},\mathcal{X})_{\text{obs}}} \text{ReLU}(h(\mathbf{o}^j, \mathbf{x}^j) + \epsilon_1) \quad (19)$$

is computed over a set of "boundary samples" $(\mathbf{o}, \mathbf{x})^j \in \delta(\mathcal{O}, \mathcal{X})_{\text{obs}}$ is added to (18), where $\delta(\mathcal{O}, \mathcal{X})_{\text{obs}}$ denotes the boundary of $(\mathcal{O}, \mathcal{X})_{\text{obs}}$ (e.g., states on the obstacle boundary).

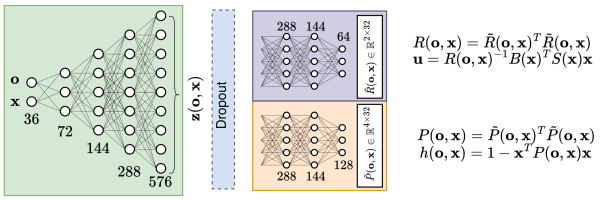

Fig. 4: Network structure and dimensions for quadrotor navigation on xy. The output of the CBF and controller network consist of the sum of 32 "feature matrices" each to allow superposition of features in the final output.

V. Implementation for Quadrotor Navigation

We apply the proposed methodology for the case of the planar dynamics of a quadrotor aerial robot with state variable $\mathbf{x}$ receiving observations $\mathbf{o}$ of its environment from an onboard LiDAR. The observation $\mathbf{o}$ is considered to be a vector of N range measurements in the horizontal plane. Neglecting the attitude dynamics, the state of the system is the linear position and velocity $\mathbf{x} = [x, y, v_x, v_y]^T$ and the input is the linear acceleration $\mathbf{u} = [a_x, a_y]^T$, yielding the linear system dynamics:

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{A(\mathbf{x})} \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}}_{B(\mathbf{x})} \begin{bmatrix} a_x \\ a_y \end{bmatrix}. \tag{20}$$

Further, we assume an input constraint $||\mathbf{u}|| \leq 2\,\mathrm{m/s^2}$. The observation $\mathbf{o}$ is considered to be a vector of $N = 32$ equally-spaced range measurements in the horizontal plane.

A. Dataset

To train the neural CBF to predict safe sets for discontinuous environments with varying obstacle density, a dataset of 10^4 simulated training observations are collected from a randomized environment with varying obstacle placement and density in the Aerial Gym Simulator [32]. To sample the state space $\mathcal{X}$, positions around a robot-aligned frame are sampled uniformly in bearing and distance, while velocities in both axes are sampled uniformly from the interval $[-v_{\max}, v_{\max}]$. Information whether a sample $(\mathbf{o},\mathbf{x})^j$ belongs to $\mathcal{X}_{\text{obs}}$ is computed by projecting the position of $\mathbf{x}^j$ and checking collisions with $\mathbf{o}^j$. Furthermore, it is assumed $\mathbf{x} \in \mathcal{X}_{\text{obs}}$ if the position falls outside a 4 m radius. To create samples on the obstacle boundaries for evaluating (19), raycasting with collision checking in $\mathbf{o}^j$ is used. Note that during training, for each observation $\mathbf{o}^j$ a new state $\mathbf{x}^j$ is sampled.

B. Training

For all the results presented in this work (simulation and experiments), a single set of weights for the network shown in Fig. 4 is used. We use ELU activations for all hidden layers and apply a dropout with $p = 0.1$ to the latent variable. The constraints $P(\mathbf{o},\mathbf{x}) \succ 0$ and $R(\mathbf{o},\mathbf{x}) \succ 0$ are always satisfied as $P(\mathbf{o},\mathbf{x})$ and $R(\mathbf{o},\mathbf{x})$ are computed as the inner product of feature matrices $\tilde{P}(\mathbf{o},\mathbf{x})$ and $\tilde{R}(\mathbf{o},\mathbf{x})$ with itself, respectively. The gradients $\nabla_{\mathbf{x}} h(\mathbf{o},\mathbf{x})$ and $\nabla_{\mathbf{x}} P(\mathbf{o},\mathbf{x})$ needed for (6) and (10) are easily computed via the autograd function in PyTorch[1] during training and at runtime. During a training cycle, a batch $\{(\mathbf{o},\mathbf{x})^j\}_{j=1}^b$ of size b is passed through all networks to compute the loss function (18). The weights of all networks are then updated synchronously using the Adam optimizer. We train for 100 epochs with batch size 32×128 (32 observations, 128 states each) without additional boundary samples and then proceed to train for 250 epochs with batch size 32×256 with additional boundary samples, where the loss (19) is added to (18). We set $\alpha(h)$ to

$$\alpha(h) = \begin{cases} 2 \cdot h, & \text{if } h \geq 0 \\ \frac{1}{0.5+|h|} & \text{otherwise} \end{cases}, \tag{21}$$

where α is selected based on desired behavior of the filter and must admit a solution to (17) given the input constraints. The evaluation statistics shown in Table I show the resulting high degree of satisfaction of requirements (7) on a holdout dataset of 10^3 observations with 256 random states each.

[1] https://pytorch.org/

TABLE I: Evaluation Statistics

Constraint from (7)	violation rate [%]
$h(\mathbf{o},\mathbf{x}) \leq 0 \quad \forall(\mathbf{o},\mathbf{x}) \in (\mathcal{O},\mathcal{X})_{\text{obs}}$	0.084
$\mathbf{u}(\mathbf{o},\mathbf{x}) \in \mathcal{U} \quad \forall(\mathbf{o},\mathbf{x}) \in (\mathcal{O},\mathcal{X})$	0.321
$\frac{d}{dt}h(\mathbf{o},\mathbf{x}) \geq -\alpha(h(\mathbf{o},\mathbf{x}))\forall(\mathbf{o},\mathbf{x}) \in (\mathcal{O},\mathcal{X})$	1.133

C. Recursive Filtering

To apply the trained CBFs in a safety filter, for every new observation, we first apply the observation update (5) and then employ condition (6) in a QP-based safety filter (3). In the case that $\mathbf{o}_{\text{safe}}$ and ξ_{safe} are not updated in (5), an estimate of ξ_{safe} is obtained by integrating the velocity since the last update. This "dead-reckoning" approach is justifiable over short time horizons and eliminates the need for a full pose estimate inside the safety filter. To avoid dead-reckoning for too long, we limit its horizon to η and penalize it in the optimization problem (22). Accounting for the robot dimensions, we evaluate $h(\mathbf{o},\mathbf{x})$ for four states $\mathbf{x}_j$ on a bounding box around the robot and solve the (soft constrained) optimization problem with the slack variable δ and the slack parameter $\lambda_s \gg 0$:

$$\begin{aligned} \mathbf{u}^* = \quad & \underset{\mathbf{u}\in\mathcal{U}}{\operatorname{argmin}} \quad ||\mathbf{u} - \mathbf{u}_{\text{ref}}||^2 + \lambda_s\delta - \lambda_g \mathbf{g}\cdot\mathbf{u} \\ \text{s.t.} \quad & \nabla_{\mathbf{x}_j} h(\mathbf{o},\mathbf{x}_j)\cdot[f(\mathbf{x}_j) + B(\mathbf{x}_j)\mathbf{u}] + \\ & \alpha(h(\mathbf{o},\mathbf{x}_j)) \geq -\delta \quad \forall j \in \{1,2,3,4\} \\ & \delta \geq 0 \end{aligned} \tag{22}$$

to enforce constraint satisfaction for the points $\mathbf{x}_j$. The term λ_g is a tuning parameter, and $\mathbf{g} = \sum_j h(\mathbf{o}_i,\mathbf{x}_j)$ during dead reckoning, and $\mathbf{g} = 0$ otherwise.

VI. Evaluation Studies

1) Aerial Robot: The conducted experiments relied on a custom-made quadrotor with dimensions $0.52 \times 0.52 \times 0.31$ m and a mass of 2.58 kg. The system integrates PX4-based autopilot avionics for low-level control, alongside an NVIDIA Orin NX compute board, an Ouster OS0-64 LiDAR and a VectorNav VN-100 IMU. We use LiDAR localization as in [33] to estimate the odometry of the robot. The input LiDAR data (20 Hz) of the horizontal plane is limited to a

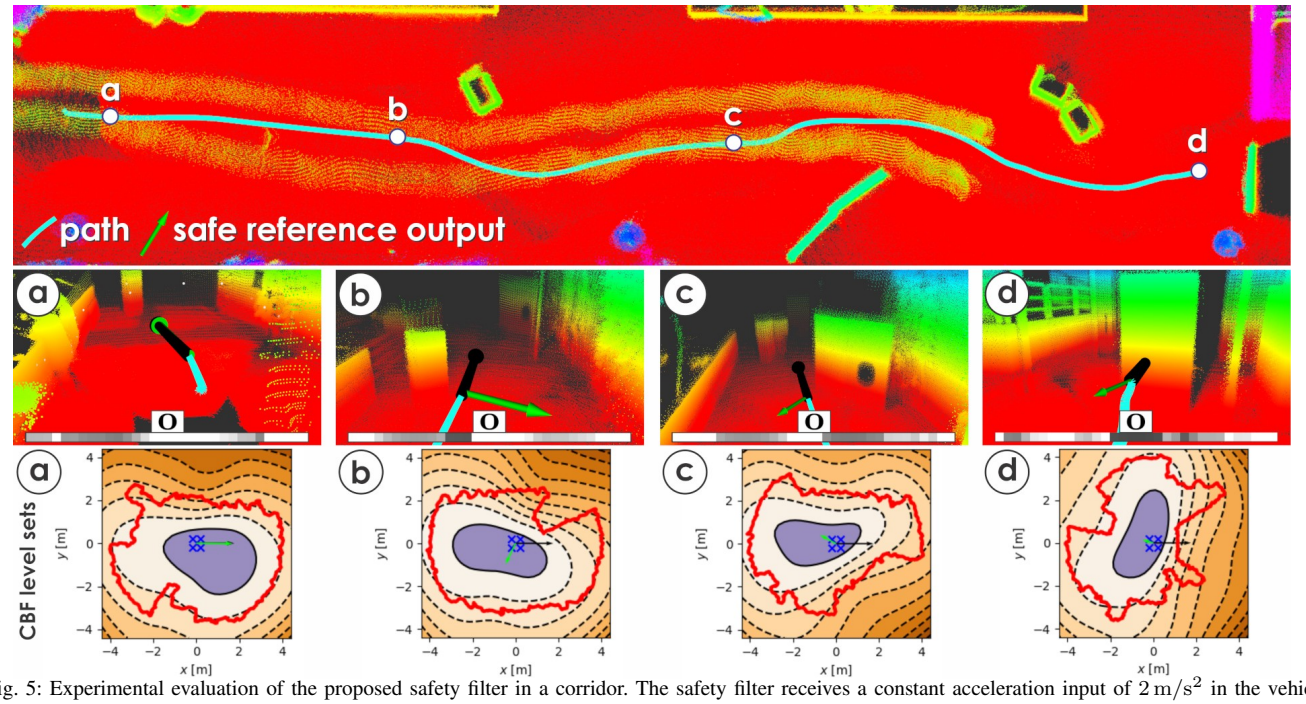

Fig. 5: Experimental evaluation of the proposed safety filter in a corridor. The safety filter receives a constant acceleration input of $2\,\mathrm{m/s^2}$ in the vehicle frame (black arrow) and produces a safe reference output (green arrow). The safety filter passes through safe reference inputs (a), deflects the multirotor when passing obstacles (b&c) and brings the robot to a full stop in front of the final panel (d). The observation (range image) at each instance is shown. CBF level sets are drawn for all instances (a-d) at the current velocity in the vehicle frame where purple denotes $h \geq 0$ and red denotes the obstacle boundary. The position of points $\mathbf{x}_j$ is shown as blue crosses.

TABLE II: Success rate in collision avoidance

Noise level	time delay	$p = 3$	$p = 5$	$p = 10$
$\sigma = 0.0$ m	$\tau = 0.0$ s	97.0	97.2	99.7
$\sigma = 0.02$ m	$\tau = 0.0$ s	97.8	99.2	99.7
$\sigma = 0.02$ m	$\tau = 0.1$ s	95.0	97.0	98.3

range of 4 m and binned angle-wise to create an observation vector $\mathbf{o}$ as in training ($N = 32$). In the following, η is set to 3. The inference time of the neural network including gradient computation is 4.4 ms. Critically, it is underlined again that the proposed approach does not utilize a reconstructed map but only the instantaneous LiDAR scan. The robot's nominal policy for the purposes of testing are commanded horizontal accelerations in the vehicle frame while yaw is regulated against the inertial frame. The proposed safety filter alters the commanded acceleration to ensure safety.

A. Simulation Study

We evaluate the proposed safety filter in a randomized simulation study in the Aerial Gym Simulator [32]. An enclosed environment of 20×10 m is constructed, bounded by 4 walls and a number of p vertical pillars of radii 0.75 m to 1.0 m are placed randomly. We simulate the quadrotor described previously with the simplified dynamics model (20) for 10^3 rollouts. During each rollout, the robot is spawned at one end of the environment and receives a constant, unsafe reference input $\mathbf{u}_{\text{ref}} = [2\,\mathrm{m/s^2}, 0\,\mathrm{m/s^2}]^T$, driving it towards the pillars or the walls. The epoch terminates after 10 s or on collision. Without the intervention of the safety filter, the robot is guaranteed to collide during every rollout. We add artificial Gaussian noise with std σ to the LiDAR image before binning and a time delay τ to the system (20) to account for the attitude dynamics. The ablated success rates

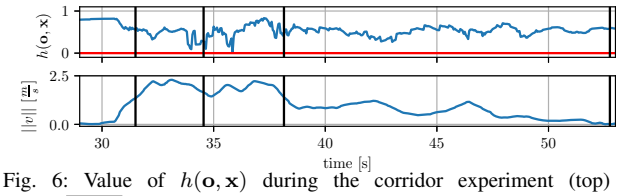

Fig. 6: Value of $h(\mathbf{o}, \mathbf{x})$ during the corridor experiment (top) and $||v|| = \sqrt{v_x^2 + v_y^2}$ (bottom). The times of instances (a-d) are marked as black lines. $h(\mathbf{o}, \mathbf{x})$ is not equal to 1 at rest since it is evaluated at $\mathbf{x}_j$.

(e.g., no collisions) over obstacle densities are shown in Table II. We note that noise on the LiDAR image improves the success rate, as we take the minimum of each bin.

B. Experimental Studies

For the experimental study, two hardware experiments were carried out in indoor and outdoor environments. In both cases, the safety filter operates at a rate of 20 Hz. In the first experiment, the robot's nominal policy is a constant commanded body-frame acceleration of $\mathbf{u}_{\text{ref}} = [2\,\mathrm{m/s^2}, 0\,\mathrm{m/s^2}]^T$. In an obstacle-filled hallway, the proposed safety filter modifies $\mathbf{u}_{\text{ref}}$ to deflect the robot away from obstacles and brings the robot to a full stop in front of the final obstacle, avoiding collisions despite the unsafe reference input along the way. A summary of the experiment is shown in Fig. 5. Considering the value of $h(\mathbf{o}, \mathbf{x})$ in Fig. 6 , we see that the proposed safety filter manages to keep the robot within $\mathcal{X}_{\text{safe}}$ over a wide range of velocities during the entire experiment.

The second experiment is conducted in a forest environment. Here, $\mathbf{u}_{\text{ref}}$ is given by the manual input of a human operator, intentionally trying to guide the robot into obstacles like tree trunks and foliage. A short sequence of the

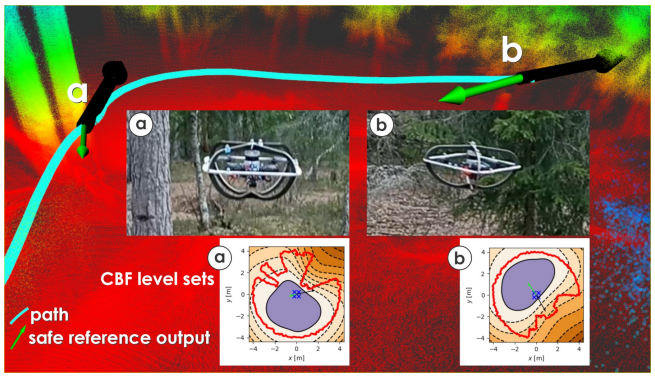

Fig. 7: Experimental evaluation in a forest environment with an adversarial human operator as a reference controller. At instance (a), the human operator tries to crash the multirotor into a tree trunk, while at (b), the obstacle is a spruce. The proposed safety filter counteracts the humans actions, avoiding collision. CBF level sets are shown in purple for the current velocity.

experiment is briefly shown in Fig. 7. The proposed safety filter successfully avoids collision by decelerating the robot and deflecting it away from obstacles.

VII. Conclusions

This work presented a new approach to using learned CBFs for map-less safe navigation of aerial robots in unknown environments. The CBF is learned without any expert controller, using an adaptation of the SDRE. Deep neural networks are trained to represent the CBF using only range observations, and the state of the robot. The learned CBF is utilized in a reactive safety filter that enforces collision avoidance. Training is performed relying purely on simulated data. Randomized simulation studies and real-world experiments demonstrate the applicability of the proposed method in ensuring the safety of the platform.

References

[1] M. Tranzatto *et al.*, "Cerberus in the darpa subterranean challenge," *Science Robotics*, vol. 7, no. 66, p. eabp9742, 2022.

[2] K. P. Valavanis *et al.*, *Handbook of unmanned aerial vehicles*. Springer, 2015, vol. 1.

[3] A. Loquercio *et al.*, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021.

[4] P. Florence *et al.*, "Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps," in *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*. Springer, 2020, pp. 304–319.

[5] H. Nguyen *et al.*, "Uncertainty-aware visually-attentive navigation using deep neural networks," *The International Journal of Robotics Research*, 2023.

[6] M. Jacquet *et al.*, "N-mpc for deep neural network-based collision avoidance exploiting depth images," *arXiv preprint arXiv:2402.13038*, 2024.

[7] M. Kulkarni *et al.*, "Reinforcement learning for collision-free flight exploiting deep collision encoding," *arXiv preprint arXiv:2402.03947*, 2024.

[8] K. P. Wabersich *et al.*, "Data-driven safety filters: Hamilton-jacobi reachability, control barrier functions, and predictive methods for uncertain systems," *IEEE Control Systems Magazine*, vol. 43, no. 5, pp. 137–177, 2023.

[9] ——, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems," *Automatica*, vol. 129, p. 109597, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109821001175

[10] A. D. Ames *et al.*, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[11] ——, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*. IEEE, 2019, pp. 3420–3431.

[12] F. Ferraguti *et al.*, "Safety and efficiency in robotics: the control barrier functions approach," *IEEE Robotics & Automation Magazine*, vol. 29, no. 3, pp. 139–151, 2022.

[13] H. U. Unlu *et al.*, "Control barrier functions and LiDAR-inertial odometry for safe drone navigation in GNSS-denied environments," in *Motion Planning for Dynamic Agents*. IntechOpen, 2023.

[14] C. Dawson *et al.*, "Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control," *IEEE Transactions on Robotics*, 2023.

[15] ——, "Learning safe, generalizable perception-based hybrid control with certificates," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1904–1911, 2022.

[16] S. Keyumarsi *et al.*, "LiDAR-based online control barrier function synthesis for safe navigation in unknown environments," *IEEE Robotics and Automation Letters*, 2023.

[17] C. Dawson *et al.*, "Safe nonlinear control using robust neural lyapunov-barrier functions," in *Conference on Robot Learning*. PMLR, 2022, pp. 1724–1735.

[18] A. Singletary *et al.*, "Comparative analysis of control barrier functions and artificial potential fields for obstacle avoidance," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8129–8136.

[19] Z. Jian *et al.*, "Dynamic control barrier function-based model predictive control to safety-critical obstacle-avoidance of mobile robot," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3679–3685.

[20] I. Jang *et al.*, "Safe control for navigation in cluttered space using multiple lyapunov-based control barrier functions," *IEEE Robotics and Automation Letters*, 2024.

[21] S. Zhou *et al.*, "Control-barrier-aided teleoperation with visual-inertial slam for safe mav navigation in complex environments," *arXiv preprint arXiv:2403.04331*, 2024.

[22] R. Cheng *et al.*, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 3387–3395.

[23] J. Choi *et al.*, "Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions," in *Robotics: Science and Systems (RSS)*, 2020.

[24] A. Taylor *et al.*, "Learning for safety-critical control with control barrier functions," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 708–717.

[25] L. Song *et al.*, "Safe reinforcement learning for LiDAR-based navigation via control barrier function," in *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2022, pp. 264–269.

[26] W. Xiao *et al.*, "Barriernet: Differentiable control barrier functions for learning of safe robot control," *IEEE Transactions on Robotics*, 2023.

[27] H. Abdi *et al.*, "Safe control using vision-based control barrier function (V-CBF)," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 782–788.

[28] M. Tong *et al.*, "Enforcing safety for vision-based controllers via control barrier functions and neural radiance fields," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 10 511–10 517.

[29] M. Srinivasan *et al.*, "Synthesis of control barrier functions using a supervised machine learning approach," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 7139–7145.

[30] K. Long *et al.*, "Learning barrier functions with memory for robust safe navigation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4931–4938, 2021.

[31] T. Çimen, "State-dependent riccati equation (sdre) control: A survey," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 3761–3775, 2008, 17th IFAC World Congress.

[32] M. Kulkarni *et al.*, "Aerial gym–isaac gym simulator for aerial robots," *arXiv preprint arXiv:2305.16510*, 2023.

[33] S. Khattak *et al.*, "Complementary multi–modal sensor fusion for resilient robot pose estimation in subterranean environments," in *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2020, pp. 1024–1029.