2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

ORN-CBF: Learning Observation-conditioned Residual Neural Control Barrier Functions via Hypernetworks

Bojan Derajić[1,2], Sebastian Bernhard[1] and Wolfgang Hönig[2]

***Abstract*— Control barrier functions (CBFs) have been demonstrated as an effective method for safety-critical control of autonomous systems. Although CBFs are simple to deploy, their design remains challenging, motivating the development of learning-based approaches. Yet, issues such as suboptimal safe sets, applicability in partially observable environments, and lack of rigorous safety guarantees persist. In this work, we propose observation-conditioned neural CBFs based on Hamilton-Jacobi (HJ) reachability analysis, which approximately recover the maximal safe sets. We exploit certain mathematical properties of the HJ value function, ensuring that the predicted safe set never intersects with the observed failure set. Moreover, we leverage a hypernetwork-based architecture that is particularly suitable for the design of observation-conditioned safety filters. The proposed method is examined both in simulation and hardware experiments for a ground robot and a quadcopter. The results show improved success rates and generalization to out-of-domain environments compared to the baselines.**

I. INTRODUCTION

One of the most prominent and well-established approaches to safety-critical control is the so-called *safety filtering*. This approach functions by monitoring the nominal control (obtained from, e.g., a performance-oriented controller or a human operator) and modifying it when necessary to maintain safe operation of the system. The nominal control is usually modified by a switching mechanism or through an optimization procedure [1]. In this paper, we focus on the second approach, which specifically utilizes control barrier functions (CBFs).

CBFs, inspired by control Lyapunov functions, are scalar functions defined over the system's state space that characterize control invariant sets based on Nagumo's Theorem on set invariance [2], [3]. In practice, the key advantage of using CBFs is emphasized for control-affine systems, where the safety filtering is reduced to solving a quadratic program (QP) and can be performed with high frequency [4]. Although CBFs provide an elegant way to enforce safety, the main difficulty lies in designing a proper CBF. This problem is especially complex for nonlinear systems with state and input constraints, and most existing approaches make assumptions or simplifications to obtain a solution [1]. Also, an additional layer of complexity is present for systems such as mobile robots that operate in unknown environments based on local observations. For such systems, the CBF must be generated in real time using the available observations, meaning that the existing methods for offline CBF design are not applicable [5].

In this paper, we propose a learning-based approach for designing observation-conditioned CBFs that overcome the limitations of existing methods by leveraging a few key features. First, we use an efficient hypernetwork-based architecture in which the hypernetwork parametrizes the main network given available observations (2D occupancy grid in our experiments), while the main network approximates the target function over the system's state space. This architecture improves efficiency by inferring the hypernetwork (complex model) only once when a new observation becomes available, while the main network (simple model) is queried frequently for its value and gradient. Second, we use Hamilton-Jacobi (HJ) reachability analysis as the source of supervision during training, resulting in CBFs that approximately recover the maximal control invariant set [6], [7], [8]. Finally, instead of directly approximating the HJ value function, the main network approximates only the residual component with respect to the signed distance function (SDF). Since the residual function is always nonnegative [9], by applying a nonnegative activation function at the output of the main network, the safe set associated with the predicted CBF never intersects with the observed failure set. The summary of the contributions is:

- A novel observation-conditioned neural CBF for safe navigation of mobile robots in unknown environments with arbitrary distribution and shapes of obstacles. The proposed method approximately recovers the optimal safe set and it guarantees, by design, that the predicted safe set does not intersect with the observed failure set.
- Extensive evaluation of the proposed ORN-CBF (Fig. 1) method for a ground robot and a quadcopter, both in 3D simulations and hardware experiments, demonstrating its performance and application flexibility.

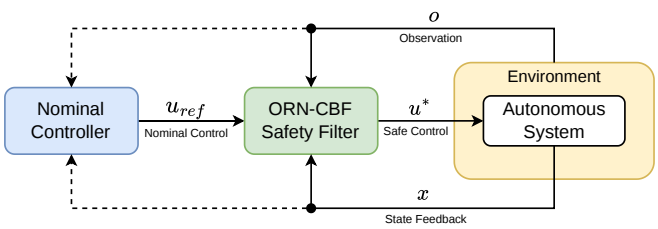

Fig. 1. Safety filtering for autonomous systems operating based on environment observations with the proposed ORN-CBF method.

[1]AUMOVIO, Germany
[2]Technical University of Berlin, Germany
Contact e-mail: bojan.derajic@aumovio.com
This work is funded by the German Federal Ministry for Economic Affairs and Climate Action within the project *nxtAIM*.

II. RELATED WORK

There is a significant amount of work on CBFs published in recent years as they have been proven to be a powerful, yet elegant way to enforce safety [3]. Numerous researchers contributed through their work on different variants of CBFs, including exponential, discrete-time, robust, time-varying, and adaptive CBFs [10]. Nevertheless, designing a proper CBF for a general nonlinear system with state and control constraints remains the major challenge in the field [1].

The particularly relevant stream of research leverages machine learning (ML) to design CBFs in a more flexible and scalable way. For example, supervised ML is applied in [11] to construct CBFs from LiDAR rays, learning CBFs from expert trajectories is proposed in [12], while the neural CBFs for systems with limited control are introduced in [13]. Also, an uncertainty-aware CBF method based on Gaussian processes is developed in [14]. However, these approaches often fail to recover the maximal safe set, which can be obtained via HJ reachability analysis [8]. The connection between CBFs and HJ reachability is established in [6], where the novel control-barrier value function (CBVF) is introduced. This approach is further extended in [7] to iteratively refine candidate CBFs, hand-crafted or learned, to enlarge the initial safe set. Similarly, authors in [15] use HJ reachability to locally modify potentially unsafe regions associated with a neural CBF, improving the overall scalability. Still, these methods have been demonstrated only for systems operating in known environments, which is the key difference to our method.

Regarding the design of neural CBFs in unknown environments, the method in [16] learns a neural SDF approximation for the given observation and uses it as a CBF. The main drawback is that the training is performed online, limiting its application to systems with slower dynamics. Differently, the authors in [17] use a discrete-time CBF, which requires predicting the sensor observation one time step into the future. This approach is feasible for simple sensors such as 2D laser scan, but might be impractical for more complex modalities. On the other hand, authors in [5] train a neural model to approximate the value function associated with the state-dependent Riccati equation based on a 2D laser scan and use it as CBF.

The common downside of the observation-based CBFs mentioned above is that they fail to recover the optimal safe set and lack any safety guarantees. Similar to [18] and [9] that employ a neural approximation of the HJ value function for model predictive control (MPC) design, we use such an approximation as a CBF. We also exploit certain mathematical properties of the HJ value function, allowing us to guarantee that the resulting safe set never includes the failure set. Moreover, we leverage a hypernetwork-based architecture, making our method particularly efficient for safety filtering based on observations.

III. PRELIMINARIES

A. System Dynamics

In this paper, we consider continuous-time systems with control-affine dynamics:

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

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state vector and $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input. We assume that f_c satisfies the standard requirements on the solution existence and uniqueness, and that full state feedback is available.

B. Control Barrier Functions

A continuously differentiable function $h : \mathcal{X} \to \mathbb{R}$ is a *control barrier function (CBF)* if there exists an (extended) class κ_∞ function α such that for the system (1)

$$\sup_{u \in \mathcal{U}} \nabla h(x)^\top f_c(x, u) \geq -\alpha\left(h(x)\right), \forall x \in \mathcal{X}. \tag{2}$$

If we define set $\mathcal{S}$ as

$$\mathcal{S} = \{x \in \mathcal{X} : h(x) \geq 0\}, \tag{3}$$

then $\mathcal{S}$ is forward invariant and we call it the *safe set* [3].

Taking into account that the system is control-affine, based on the Lie derivatives $L_f h(x)$ and $L_g h(x)$ with respect to $f(x)$ and $g(x)$, respectively, the inequality (2) can be rewritten as

$$\sup_{u \in \mathcal{U}} \left[L_f h(x) + L_g h(x) u\right] \geq -\alpha\left(h(x)\right), \forall x \in \mathcal{X}, \tag{4}$$

which is linear in u for a given x. Moreover, if the set of admissible controls $\mathcal{U}$ can be represented by a set of linear constraints, the CBF condition (4) can be used to design a safety filter in the form of a quadratic program (CBF-QP) [1], [4]. The CBF-QP safety filter provides safe control u^* by solving the following optimization problem for the nominal control u_{ref} and state x:

$$\begin{aligned} u^* = \arg\min_{u \in \mathcal{U}} \; & \frac{1}{2}\|u_{ref} - u\|^2 \\ s.t. \quad & L_f h(x) + L_g h(x) u + \alpha\left(h(x)\right) \geq 0. \end{aligned} \tag{5}$$

Thanks to modern QP solvers, CBF-QP can run at relatively high frequencies even with scarce computational resources, and we will design such a safety filter in this paper.

C. Hamilton-Jacobi Reachability Analysis

HJ reachability analysis is a framework for verification and analysis of dynamical systems grounded in the optimal control theory [19]. In this paper, we consider undisturbed systems described by (1). If the system starts at state $x(t)$, then $\xi^u_{x,t}(\tau)$ is the system's state at time τ after applying $u(\cdot)$ over time horizon $[t, \tau]$. The set of states that should be ultimately avoided is called a failure set $\mathcal{F}$. Also, we define a backward reachable tube (BRT) $\mathcal{B}(t)$, which is a set of initial states from which the system will reach $\mathcal{F}$ within the time horizon $[t, T]$ for any control $u(\cdot)$, i.e.

$$\mathcal{B}(t) = \{x : \forall u(\cdot),\, \exists \tau \in [t, T],\, \xi^u_{x,t}(\tau) \in \mathcal{F}\}. \tag{6}$$

To compute the BRT for a given failure set, we define $\mathcal{F}$ as the zero-sublevel set of a failure function $F(x)$, i.e. $\mathcal{F} = \{x : F(x) \leq 0\}$. The computation of the BRT can be formulated as an optimization problem where the objective is the minimal distance to $\mathcal{F}$ over the time horizon, i.e.

$$J(x, t, u(\cdot)) = \min_{\tau \in [t,T]} F(\xi^u_{x,t}(\tau)). \tag{7}$$

Since $\mathcal{F}$ represents an unsafe region, the goal is to find the optimal control that will maximize this distance and therefore we introduce the following value function:

$$V(x, t) = \sup_{u(\cdot) \in \mathcal{U}} \{J(x, t, u(\cdot))\}. \tag{8}$$

This value function can be computed using the dynamic programming principle to solve the following Hamilton-Jacobi-Bellman Variational Inequality (HJB-VI) [19]:

$$\min \left\{ \frac{\partial}{\partial t} V(x, t) + H(x, t),\ F(x) - V(x, t) \right\} = 0, \quad V(x, T) = F(x). \tag{9}$$

In the formulation above, Hamiltonian $H(x, t)$ is defined as

$$H(x, t) = \max_{u \in \mathcal{U}} \nabla V(x, t)^\top f_c(x, u). \tag{10}$$

Once the value function (8) is obtained, the corresponding BRT can be described as its zero-sublevel set, i.e.

$$\mathcal{B}(t) = \{x : V(x, t) \leq 0\}, \tag{11}$$

and the maximal safe set is its complement: $\mathcal{S}(t) = \mathcal{B}(t)^c$.

In this paper, the failure function is the space occupied by obstacles and defined as an SDF $d(x)$, i.e., $F(x) \coloneqq d(x)$. Also, by assuming a terminal time $T = 0$ and a static environment, we propagate the value function backward in time until it converges to its steady-state value $V(x) = \lim_{t \to -\infty} V(x, t)$ and then use $V(x)$ as the target CBF during training[1].

IV. METHODOLOGY

A. Observation-conditioned CBFs

When dealing with autonomous systems operating based on the exteroceptive observations of the environment, the CBF is generally a function of the system's state x and observation o, i.e. $h \coloneqq h(x, o)$. In that case, the CBF constraint (2) can be written as:

$$\nabla_x h(x, o)^\top \dot{x} + \nabla_o h(x, o)^\top \dot{o} \geq -\alpha\left(h(x, o)\right). \tag{12}$$

In this formulation, the main difficulty arises due to the unknown observation dynamics $\dot{o}$. This dynamics can be partially described by employing the sensor's model and predicting how the observation will change, e.g., due to the robot's motion [17]. However, this approach does not account for the obstacles that are not yet observed because the information about the distribution of obstacles in the environment is not available a priori.

To mitigate unknown observation dynamics, we leverage the following two properties that appear often in practice. First, the observations are typically updated with a lower frequency compared to the state feedback, which means that a particular observation is constant over a considerable period of time. Second, when an observation is updated, the new information is usually introduced at the edges of the perception field[2]. The only requirement is that the perception field is large enough so that the obstacles that appear on the observation boundary do not make the new state unsafe instantly. Based on that, we can interpret the CBF as being conditioned on the environment observation, i.e. $h \coloneqq h(x|o)$, and the CBF constraint is defined as:

$$\dot{h}(x|o) = \nabla_x h(x|o)^\top \dot{x} \geq -\alpha\left(h(x|o)\right). \tag{13}$$

The key practical implication of this interpretation is that the constraint formulation remains unchanged compared to the CBFs defined only for x. Additionally, by interpreting a CBF as a function conditioned on observation, we can directly use the existing numerical tools for HJ reachability analysis to compute the HJ value function for the given observation. Otherwise, it would be necessary to model and simulate observation dynamics, which would be impractical.

In essence, a new CBF is generated when a new observation becomes available, which is the most critical moment. Whether the system stays safe when the new observation occurs depends on multiple factors such as the observation update rate, the maximal speed of the robot, the size of the perception field, the maximal size of an obstacle and the corresponding BRT shape. As mentioned earlier, it is required that the robot stay within the safe set, i.e., outside of the BRT, when the new observation occurs and the CBF is updated. We illustrate this in Fig. 2 where the robot moves with the maximal speed v_{max} and the observation is updated every δt_o. In the worst case, the robot will move maximally $v_{max}\delta t_o$ between the two observations and the obstacle will penetrate the perception field (blue region) for the same length in the new observation at $t = t' + \delta t_o$. It is then important that the robot is outside of the obstacle's BRT $\mathcal{B}$ at that moment in order to continue safe motion.

B. Observation-conditioned residual neural CBF

Among the most important properties of a CBF is the size of the safe set $\mathcal{S}$ associated with that CBF. This is our main motivation for using the HJ value function as a CBF since this function recovers the optimal safe set for the given system [6], [7], [8]. However, computing the HJ value function in real time is generally intractable and therefore we use a learning-based approach, which allows us to approximate the value function based on available observations and then employ that approximation as an observation-conditioned CBF, i.e., $\hat{h}(x|o) \coloneqq \hat{V}(x|o)$.

[1]Although HJ value function is a viscosity solution to the HJB-VI and theoretically might not be differentiable everywhere, the gradients can always be computed via numeric or automatic differentiation in practice.

[2]New information may also appear closer to the robot when occluded objects suddenly become visible, but this problem is beyond the scope of this paper.

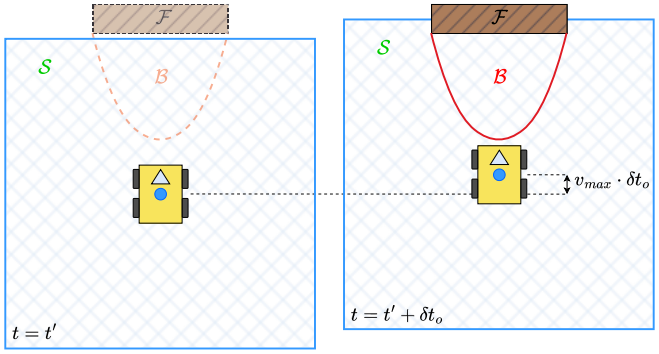

Fig. 2. Illustration of the critical moment when the new observation occurs. The robot must be outside of the BRT $\mathcal{B}$ corresponding to the suddenly observed failure set $\mathcal{F}$. This can be achieved by considering all aspects such as the robot's maximal speed, observation size and update rate, maximal expected size of an obstacle, and the shape of the corresponding BRT.

First, we recall that the HJ value function obtained by solving HJB-VI (9) is less than or equal to the corresponding failure function [9]. In our context, since the failure function is equal to the SDF, we have that $h(x|o) \leq d(x|o), \forall x$, which means that the CBF can be obtained by subtracting a nonnegative residual function from the SDF:

$$h(x|o) = d(x|o) - r(x|o) \;\; s.t. \;\; r(x|o) \geq 0, \forall x. \tag{14}$$

Also, we note that a continuous SDF approximation $\hat{d}(x|o)$ can be derived by a simple interpolation of a discretized SDF $\bar{d}$, which in turn can be efficiently computed from an observation such as an occupancy grid. As a result, the formulation above allows us to learn only the residual component instead of the complete CBF.

We approximate the residual using a neural network with parameters Θ and denote it with $\hat{r}_\Theta(x|o)$. To ensure a nonnegative value for all inputs, we apply a nonnegative activation at the model's output. A straightforward option would be the ReLU function defined as $\max(0, z)$. However, the gradient of the function is always equal to 0 for negative inputs, which can negatively impact the training process. Also, the activation should be continuously differentiable, so we opt for the softplus function:

$$\eta(z) = \log\left(1 + \exp(z)\right). \tag{15}$$

In addition, given that the approximation of the CBF gradient $\nabla\hat{h}(x|o)$, and therefore $\nabla\hat{r}_\Theta(x|o)$, is required for the CBF constraint (13), we use the multilayer perceptron (MLP) model with sinusoidal activation functions. Those models have been proven to work well for applications where the gradient of the approximated function has to be accurately approximated as well [20], [21].

However, instead of directly learning parameters Θ, we take the hypernetwork-based architecture which involves two neural networks - the hypernetwork and the main network. The hypernetwork is a large and expressive trainable model that parametrizes the main network for the given hypernetwork input. On the other hand, the main network is a simple model that approximates the target function for that particular input of the hypernetwork. In our context, the input to the hypernetwork is the discretized SDF $\bar{d}$ obtained from the observation o, while the main network is the previously described MLP model approximating the residual function for the given observation. This architecture is particularly suitable for our observation-conditioned interpretation of CBF safety filtering because the hypernetwork essentially conditions the main network to approximate the residual for the given observation. Also, this architecture has been demonstrated to be more efficient than a single-network model because the hypernetwork is inferred only once when the new observation becomes available, which happens at a lower rate, while the main network is queried for its value and gradient with a much higher frequency [18]. Moreover, the two models can be designed almost completely separately; the only requirement is that the number of outputs of the hypernetwork is equal to the number of parameters of the main network.

In summary, for the given observation o (e.g. occupancy grid map), we first compute a discretized SDF $\bar{d}$ and provide it to the hypernetwork and to the interpolation function. The hypernetwork outputs parameters Θ that parameterize the main network representing residual $\hat{r}_\Theta(x|o)$, the interpolation function represents $\hat{d}(x|o)$, while the approximated CBF is obtained as

$$\hat{h}(x|o) = \hat{d}(x|o) - \hat{r}_\Theta(x|o). \tag{16}$$

By implementing the main network and interpolation function in a framework that supports automatic differentiation (e.g. PyTorch or CasADi), the gradients $\nabla\hat{d}(x|o)$ and $\nabla\hat{r}_\Theta(x|o)$ can be efficiently obtained for the given state x. The CBF gradient $\nabla\hat{h}(x|o)$ is then used to compute the Lie derivatives $L_f\hat{h}(x|o)$ and $L_g\hat{h}(x|o)$, which are finally provided to the CBF-QP together with $\hat{h}(x|o)$. We call this method observation-conditioned residual neural CBF (ORN-CBF) and the complete architecture is visualized in Fig. 3.

To train the hypernetwork for the proposed ORN-CBF method, we resort to the supervised approach, which requires both inputs and the corresponding target outputs. In our setting, we have two sets of inputs: a set of discretized SDFs (inputs to the hypernetwork) and a set of state space grid points (inputs to the main network). The target outputs are HJ value functions numerically computed over the same state space grid, representing the target CBFs paired with the corresponding observations. As a loss function, we use the radially weighted MSE (RWMSE) loss function proposed in [18], which improves approximation accuracy near the zero-level sets of the value function.

Taking into account that the robot travels a relatively short distance in space between two observations, instead of learning the HJ value function over the complete observable space like in [18], we can learn the HJ value function only for the surrounding positions reachable from the current position before the observation is updated. For example, the environment observation in our experiments is a square-shaped occupancy grid centered at the robot's position at the time when the observation occurs. Assuming that the

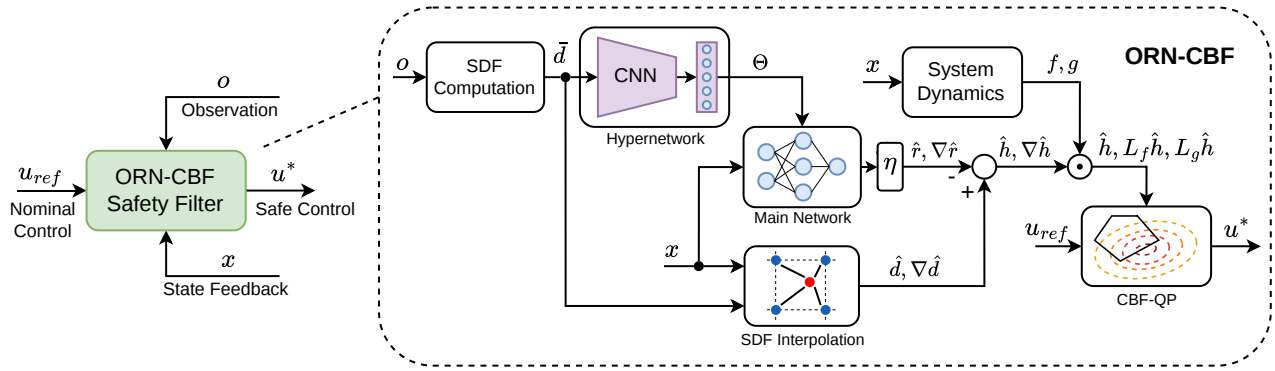

Fig. 3. A detailed architecture of the ORN-CBF safety filter. Based on the observation o, a discretized SDF $\bar{d}$ is computed, which is then fed into the hypernetwork and the SDF interpolation function. The hypernetwork parametrizes the main network, which approximates the nonnegative residual $\hat{r}$ and its gradient $\nabla\hat{r}$. The approximate CBF value $\hat{h}$ and gradient $\nabla\hat{h}$ are obtained by subtracting $\hat{r}$ and $\nabla\hat{r}$ from interpolated SDF value $\hat{d}$ and gradient $\nabla\hat{d}$, respectively. In the end, $L_f\hat{h}$ and $L_g\hat{h}$ are computed based on $\nabla\hat{h}$ and the system dynamics and together with $\hat{h}$ provided to the CBF-QP safety filter.

robot's maximal speed is v_{max} in all directions and that the observations are updated every δt_o, the robot can reach only positions within the circle of radius $v_{max}\delta t_o$ before the observation is recentered at the robot's new position. Therefore, it is enough to approximate the HJ value function only for the region that includes those reachable positions[3]. The key practical benefit of this insight is a more efficient learning process since the neural CBF is trained to approximate the HJ value function only for this patch, and not the complete observed space. This results in lower memory requirements, faster training and less complex models.

V. EXPERIMENTAL RESULTS

In this section, we first introduce the considered robot models, followed by details on the hypernetwork training process and an analysis of the results obtained from simulation and hardware experiments[4].

A. Robot Models

The first type of robot is a ground robot modeled as a Dubins car, which is a first-order unicycle model with constant linear speed and limited angular speed. In practice, this model describes systems for which linear speed is not controllable or its change over time is negligible. Some examples include vehicles with faulty braking systems, forklifts transporting unstable objects, fixed-wing aircraft, marine vessels, etc. The state vector $x = [p_x,\, p_y,\, \theta]^\top$ includes positional coordinates (p_x, p_y) and orientation angle θ, while the control input is only the angular speed, i.e. $u = \omega$. The dynamics equation is

$$\dot{x} = [v\cos(\theta),\, v\sin(\theta),\, \omega]^\top, \tag{17}$$

where $v = 1.0$ m/s and $\omega \in [-0.5, 0.5]$ rad/s.

The second type of robot is a quadcopter modeled as a 2D double integrator with limited speed and acceleration. Such models are used in practice for quadcopters operating at constant heights with relatively low acceleration capabilities. Some examples are quadcopters transporting heavy objects at construction sites, in agriculture, or for delivery. The state of the robot is $x = [p_x,\, p_y,\, v_x,\, v_y]^\top$, where (p_x, p_y) are positional coordinates and v_x, v_y are velocities. The control vector $u = [a_x,\, a_y]^\top$ consists of linear accelerations, while the dynamics equation is

$$\dot{x} = [v_x,\, v_y,\, a_x,\, a_y]^\top, \tag{18}$$

where $v_x, v_y \in [-2, 2]$ m/s and $a_x, a_y \in [-1, 1]$ m/s^2.

B. Model Training

As described in Section IV, we train the model in a supervised fashion, and the first part is to obtain a suitable dataset consisting of inputs and target outputs. For the hypernetwork inputs, we first collect a set of observations in the form of occupancy grid maps and compute the corresponding discretized SDFs using the method described in [22]. For the ground robot, we collect 2,500 costmaps of size 6×6 m with resolution 0.06 m from a 3D model of a warehouse environment. Similarly, we collect 2,000 costmaps of size 6×6 m with resolution 0.075 m from a forest-like 3D environment for the quadcopter model. We additionally augment both datasets by flipping horizontally and rotating for 90°, 180° and 270° each original and flipped costmap, resulting in 20,000 costmaps for the ground robot and 16,000 costmaps for the quadcopter.

Next, we define state space grids over which the HJ value functions are computed and which are used as the input to the main network. We define state space grids of shape $100 \times 100 \times 30$ for the Dubins car and $80 \times 80 \times 20 \times 20$ for the 2D double integrator model. To compute the HJ value functions, we use the *hj_reachability*[5] Python package. The computation takes ∼1.17 h in the first case, and ∼16.3 h in the second case, on a single RTX3090 GPU. Then, we store the value functions and grid points only for the subset of the initial grid according to the explanation in the last paragraph of Section IV. Concretely, for both models, we store only the central patch in the positional subspace of

[3]This holds only for the positional coordinates of the state vector, while the rest of the state space should be covered completely.

[4]Code will be released after the paper is published.

[5]https://github.com/StanfordASL/hj_reachability

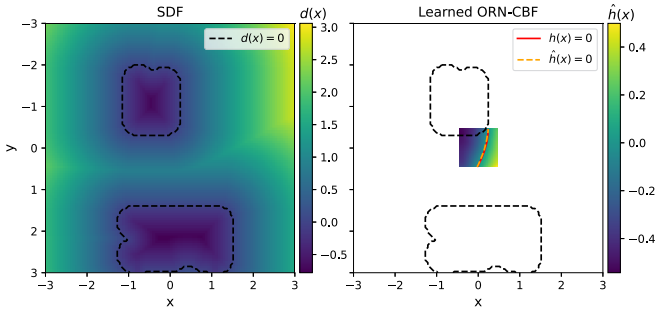

Fig. 4. A slice of a learned ORN-CBF for the Dubins car for $\theta = \frac{\pi}{2}$ rad.

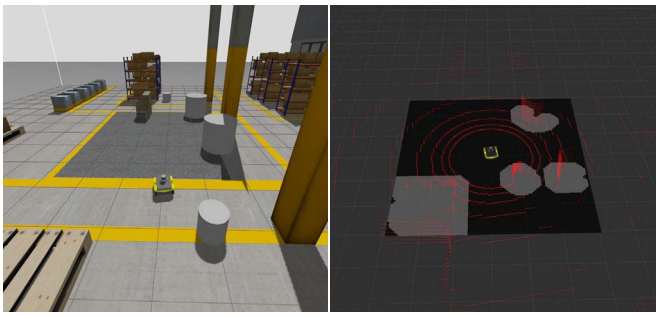

Fig. 5. Left: Scene from the 3D simulation experiment with the ground robot in the warehouse environment. Right: The corresponding observation.

size 16×16, meaning that we store $16 \times 16 \times 30$ grid points and the corresponding function values for the Dubins car and $16 \times 16 \times 20 \times 20$ for the 2D double integrator.

After aggregating the datasets, we proceed with the model training. For both types of robots, we use deep CNN models as the hypernetwork and MLP models as the main network. We use the same architecture of the main network in both cases: 32-32-32-16-16-16-8-8-8 units with sinusoidal activations in hidden layers and a single output with softplus activation, while the number of inputs is equal to the corresponding state dimension. During the training, at each epoch, a discretized SDF is propagated through the hypernetwork and the main network is parametrized with the predicted parameters. Then, the cropped state space grid is propagated through the main network to obtain the HJ value function predictions. Finally, the loss function is evaluated for the predicted and true value functions, and the parameters of the hypernetwork are updated via the gradient descent method. In this paper, we use the RWMSE loss function proposed in [18] and train models for 100 epochs with a batch size of 16. We use the Adam optimizer with an initial learning rate of 10^{-4}, which is decreased by a factor of 10 at epochs 85 and 95. The training process takes $\sim$0.5 h for the Dubins car and $\sim$2.9 h for the 2D double integrator on a single RTX3090 GPU, which is roughly 20 times faster compared to the method in [18] under the same setting. In Fig. 4 we visualize a sample input to the hypernetwork and the corresponding learned ORN-CBF for the Dubins car model.

C. Simulation Experiments

The proposed ORN-CBF method is first evaluated in simulations implemented using the Gazebo 3D simulator. The ground robot, modeled as a Dubins car, is tested in a warehouse environment visualized in Fig. 5. As a nominal controller, we use a simple MPC local planner, which uses the SDF as a collision avoidance constraint (SDF-MPC). The ORN-CBF method is examined against SDF-MPC alone and more advanced MPC-based methods that have been proven to work well in practice for safe navigation in unknown environments: DCBF-MPC [23] and NTC-MPC [18]. The related works on neural CBFs [17] and [5] assume different observation modalities, and therefore are not directly applicable in our setting. Besides that, as an ablation study, we test a simpler variant of the proposed

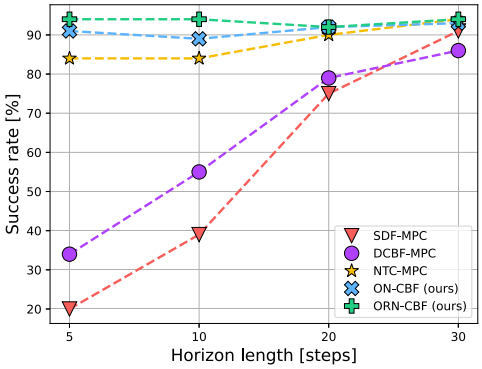

Fig. 6. Obtained success rates for different lengths of the MPC prediction horizon for the Dubins car model. The results show that the proposed ORN-CBF method, and its simple version ON-CBF, outperform the MPC-based baselines for different horizon lengths.

method, which directly approximates the HJ value function by the main network instead of the residual function. We call this method observation-conditioned neural CBF (ON-CBF). Even though this version has a less complex architecture, it does not provide any guarantees on the approximation error. As it is standard in practice, we use a linear function for $\alpha(\cdot)$, i.e. $\alpha(h) = kh, k \in [0, \infty)$.

All the MPC planners run with the control frequency of 20 Hz and use a sampling time of 0.1 s over the prediction horizon. The CBF-QP safety filters run at 200 Hz with $k = 2.6$, and the costmap is updated at 20 Hz. The MPC planners and CBF-QP are implemented in CasADi [24] with the IPOPT NLP solver and qpOASES QP solver, respectively. We run 100 experiments where the task is to navigate the robot from the initial to the goal position based only on the local observation while avoiding collisions with randomly distributed obstacles. Since both performance and safety of the MPC planners largely depend on the prediction horizon, we repeat experiments for different horizon lengths. The obtained success rates are shown in Fig. 6, demonstrating the advantage of using the proposed ORN-CBF safety filter. Moreover, even the simpler ON-CBF variant outperforms the baseline methods for short prediction horizon.

The quadcopter robot, modeled as a 2D double integrator, is evaluated in a forest-like environment shown in Fig. 7. As the nominal controller, we design a linear-quadratic regulator

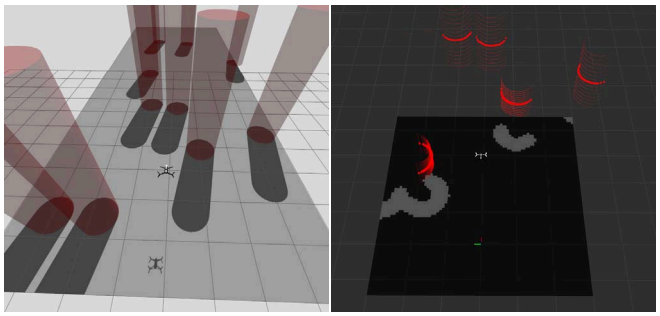

Fig. 7. Left: Scene from the 3D simulation experiment with the quadcopter in the forest-like environment. Right: The corresponding observation.

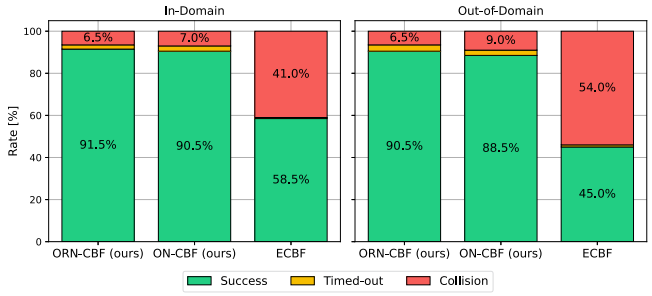

Fig. 8. Results obtained from the simulation experiments with a quadcopter. The results indicate strong generalization properties of the proposed ORN-CBF and ON-CBF methods compared to a hand-tuned ECBF.

(LQR) with $Q = \mathrm{diag}(10, 10, 1, 1)$ and $R = \mathrm{diag}(1, 1)$.

Instead of evaluating different nominal controllers, we examine the generalization properties of the learned CBFs. ORN-CBF and ON-CBF are trained using data from a forest-like environment with cylindrical obstacles of 0.5 m radius (in-domain environment). Both methods are then evaluated for the environment where obstacles' radii range from 0.2 m to 1.0 m (out-of-domain environment). The task is to navigate from the initial to the goal position, and in both cases, there are 200 scenarios with 10 obstacles randomly distributed in the area through which the robot navigates. The CBF-QP runs at 200 Hz and the costmap is updated at 5 Hz.

For comparison, we design an exponential CBF (ECBF) [25] based on the SDF $d(x)$, which results in a second-order ECBF. The gain matrix $K_{ECBF} = [1.0, 2.0]$ is tuned for the first environment and evaluated for both environments. The obtained results for $k = 0.7$ are presented in Fig. 8, indicating solid robustness of the proposed CBF methods compared to a classical ECBF. Failing to achieve 100% rate can be attributed to multiple factors such as learning error, signal delays, finite costmap size and resolution, etc. In Fig. 9, we visualize the motion of the quadcopter from a representative scenario in simulation experiments.

D. Hardware Experiments

We perform hardware experiments with a ground robot modeled as a Dubins car, shown in Fig. 10. To evaluate sim-to-real robustness, we use the same ORN-CBF and ON-CBF models trained on synthetic data. We compare against the same MPC-based baselines using a prediction horizon of

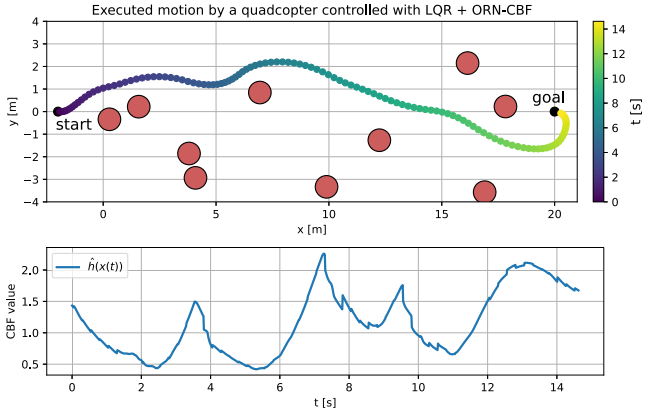

Fig. 9. Up: Visualized motion of the quadcopter controlled by the LQR and the ORN-CBF safety filter. Down: The corresponding change of the ORN-CBF value over time.

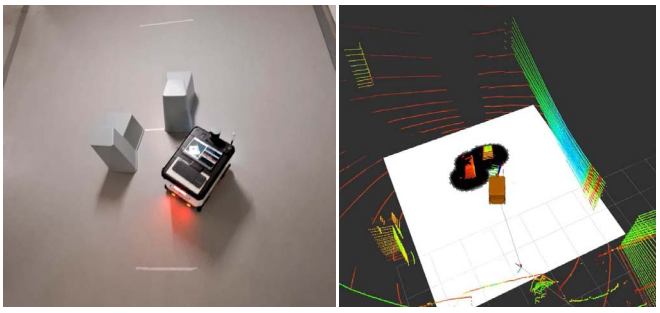

Fig. 10. Left: Ground robot used in the hardware experiments. Right: Visualization of the local observation.

TABLE I
RESULTS FROM THE HARDWARE EXPERIMENTS WITH THE DUBINS CAR

	SDF-MPC	DCBF-MPC	NTC-MPC	ORN-CBF	ON-CBF
Success rate	20%	40%	70%	**100%**	**100%**

10 steps, a sampling time of 0.1 s, and a control frequency of 20 Hz. The CBF safety filters run at 200 Hz, and all computations are performed onboard. For each method, we run 10 experiments with random obstacle distributions. The results in Table I show that the proposed safety filters perform significantly better than the baselines. Fig. 11 illustrates the CBF value over time for ORN-CBF and ON-CBF in one representative scenario. The CBF values occasionally dip slightly below zero due to model mismatch and measurement noise, yet collisions are avoided thanks to a small buffer zone that accounts for these uncertainties.

We also conduct hardware experiments with a Crazyflie quadcopter equipped with 4 laser-based distance sensors. The local costmap is created by a constant rotation of the quadcopter around the z-axis during motion. We deploy the same LQR nominal controller and learned ORN-CBF safety filter used in simulations, while the costmap is updated at the rate of 10 Hz and computation is performed offboard. The experiments illustrate successful obstacle avoidance when using the proposed ORN-CBF method, despite the challenging

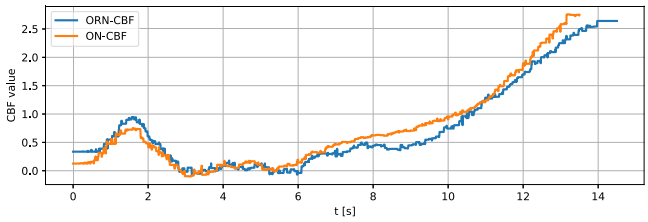

Fig. 11. CBF values for the ORN-CBF and ON-CBF methods during one of the hardware experiments with the ground robot.

hardware setup (see the supplemental video).

VI. DISCUSSION ON LIMITATIONS

We use numerical tools for HJ reachability during data generation, but these tools are typically not applicable to systems with more than six state dimensions. To improve scalability, one can use the self-supervised approach for model training as in [21], or compute approximate HJ value functions using the MPC framework [26].

On the other hand, the assumption about a static environment might limit the scope of use for our method. A solution would be to employ HJ reachability analysis for time-varying failure sets as in [9]. This would require a different data generation process and a hypernetwork that can process temporal data instead of only the current observation.

VII. CONCLUSIONS

We propose a novel learning-based method, called ORN-CBF, for designing observation-conditioned CBFs in unknown environments. The resulting CBF represents an HJ value function and approximately recovers the maximal safe set for the current observation. By learning only the residual component of the HJ value function, we guarantee that the predicted safe set never includes the observed failure set. Moreover, the proposed hypernetwork-based architecture enables a computationally efficient safety filter.

We extensively evaluate ORN-CBF in simulations on a ground robot and a quadcopter. The results show improved success rates and robust out-of-domain performance compared to the baselines. Hardware experiments further demonstrate that the safety filters can be readily deployed on different real robots. Future work will extend the method to dynamic environments and higher-dimensional systems.

REFERENCES

[1] K.-C. Hsu, H. Hu, and J. F. Fisac, "The Safety Filter: A Unified View of Safety-Critical Control in Autonomous Systems," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 7, no. Volume 7, pp. 47–72, 2024.

[2] M. Nagumo, "Über die Lage der Integralkurven gewöhnlicher Differentialgleichungen," *Proc. Phys.-Math. Soc. Japan, 3rd Ser.*, vol. 24, pp. 551–559, 1942.

[3] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control Barrier Functions: Theory and Applications," in *Eur. Control Conf. (ECC)*, 2019, pp. 3420–3431.

[4] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *IEEE Conf. Decis. Control (CDC)*, 2014, pp. 6271–6278.

[5] M. Harms, M. Kulkarni, N. Khedekar, M. Jacquet, and K. Alexis, "Neural Control Barrier Functions for Safe Navigation," in *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2024, pp. 10 415–10 422.

[6] J. J. Choi, D. Lee, K. Sreenath, C. J. Tomlin, and S. L. Herbert, "Robust Control Barrier-Value Functions for Safety-Critical Control," in *IEEE Conf. Decis. Control (CDC)*, 2021, pp. 6814–6821.

[7] S. Tonkens and S. Herbert, "Refining Control Barrier Functions through Hamilton-Jacobi Reachability," in *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2022, pp. 13 355–13 362.

[8] O. So, Z. Serlin, M. Mann, J. Gonzales, K. Rutledge, N. Roy, and C. Fan, "How to Train Your Neural Control Barrier Function: Learning Safety Filters for Complex Input-Constrained Systems," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2024, pp. 11 532–11 539.

[9] B. Derajić, M.-K. Bouzidi, S. Bernhard, and W. Hönig, "Residual Neural Terminal Constraint for MPC-based Collision Avoidance in Dynamic Environments," in *Conf. Robot. Learn. (CoRL)*, 2025.

[10] K. Garg, J. Usevitch, J. Breeden, M. Black, D. Agrawal, H. Parwana, and D. Panagou, "Advances in the Theory of Control Barrier Functions: Addressing practical challenges in safe control synthesis for autonomous and robotic systems," *Annu. Rev. Control*, vol. 57, p. 100945, 2024.

[11] M. Srinivasan, A. Dabholkar, S. Coogan, and P. A. Vela, "Synthesis of Control Barrier Functions Using a Supervised Machine Learning Approach," in *IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*. Las Vegas, NV, USA: IEEE, 2020, pp. 7139–7145.

[12] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni, "Learning Control Barrier Functions from Expert Demonstrations," in *IEEE Conf. Decis. Control (CDC)*, 2020, pp. 3717–3724.

[13] S. Liu, C. Liu, and J. Dolan, "Safe Control Under Input Limits with Neural Control Barrier Functions," in *Conf. Robot. Learn. (CoRL)*, 2022.

[14] J. Li, Q. Liu, W. Jin, J. Qin, and S. Hirche, "Robust Safe Learning and Control in an Unknown Environment: An Uncertainty-Separated Control Barrier Function Approach," *IEEE Robot. Autom. Lett.*, vol. 8, no. 10, pp. 6539–6546, 2023.

[15] S. Tonkens, A. Toofanian, Z. Qin, S. Gao, and S. Herbert, "Patching Approximately Safe Value Functions Leveraging Local Hamilton-Jacobi Reachability Analysis," in *IEEE Conf. Decis. Control (CDC)*, 2024, pp. 3577–3584.

[16] K. Long, C. Qian, J. Cortes, and N. Atanasov, "Learning Barrier Functions With Memory for Robust Safe Navigation," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4931–4938, 2021.

[17] C. Dawson, B. Lowenkamp, D. Goff, and C. Fan, "Learning Safe, Generalizable Perception-Based Hybrid Control With Certificates," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 1904–1911, 2022.

[18] B. Derajić, M.-K. Bouzidi, S. Bernhard, and W. Hönig, "Learning Maximal Safe Sets Using Hypernetworks for MPC-Based Local Trajectory Planning in Unknown Environments," *IEEE Robot. Autom. Lett.*, vol. 10, no. 9, pp. 8842–8849, 2025.

[19] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi reachability: A brief overview and recent advances," in *IEEE Conf. Decis. Control (CDC)*, 2017, pp. 2242–2253.

[20] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in *Neural Inf. Process. Syst. (NeurIPS)*, 2020, pp. 7462–7473.

[21] S. Bansal and C. J. Tomlin, "DeepReach: A Deep Learning Approach to High-Dimensional Reachability," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2021, pp. 1817–1824.

[22] C. Maurer, R. Qi, and V. Raghavan, "A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 265–270, 2003.

[23] J. Zeng, B. Zhang, and K. Sreenath, "Safety-Critical Model Predictive Control with Discrete-Time Control Barrier Function," in *Amer. Control Conf. (ACC)*, 2021, pp. 3882–3889.

[24] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[25] Q. Nguyen and K. Sreenath, "Exponential Control Barrier Functions for enforcing high relative-degree safety-critical constraints," in *Amer. Control Conf. (ACC)*, 2016, pp. 322–328.

[26] Z. Feng, L. Qiu, and S. Bansal, "Bridging Model Predictive Control and Deep Learning for Scalable Reachability Analysis," in *Robot. Sci. Syst. (RSS)*, 2025.