In-Distribution Barrier Functions: Self-Supervised Policy Filters that Avoid Out-of-Distribution States

Fernando Castañeda
University of California, Berkeley
FCASTANEDA@BERKELEY.EDU

Haruki Nishimura
Toyota Research Institute, Los Altos, CA
HARUKI.NISHIMURA@TRI.GLOBAL

Rowan McAllister
Toyota Research Institute, Los Altos, CA
ROWAN.MCALLISTER@TRI.GLOBAL

Koushil Sreenath
University of California, Berkeley
KOUSHILS@BERKELEY.EDU

Adrien Gaidon
Toyota Research Institute, Los Altos, CA
ADRIEN.GAIDON@TRI.GLOBAL

Abstract

Learning-based control approaches have shown great promise in performing complex tasks directly from high-dimensional perception data for real robotic systems. Nonetheless, the learned controllers can behave unexpectedly if the trajectories of the system divert from the training data distribution, which can compromise safety. In this work, we propose a control filter that wraps any reference policy and effectively encourages the system to stay in-distribution with respect to offline-collected safe demonstrations. Our methodology is inspired by Control Barrier Functions (CBFs), which are model-based tools from the nonlinear control literature that can be used to construct minimally invasive safe policy filters. While existing methods based on CBFs require a known low-dimensional state representation, our proposed approach is directly applicable to systems that rely solely on high-dimensional visual observations by learning in a latent state-space. We demonstrate that our method is effective for two different visuomotor control tasks in simulation environments, including both top-down and egocentric view settings.

Keywords: Distributional Shift, Control Barrier Functions, State Representation Learning

1. Introduction

The modern advances in the representation learning literature have been an enabling factor for the recent surge of a wide variety of methods for robotic control directly from images or high-dimensional sensory observations (Watter et al., 2015; Ebert et al., 2018; Hafner et al., 2019; Lenz et al., 2015; Zhang et al., 2019; Van Hoof et al., 2016). These approaches for visuomotor planning and control have the potential to solve challenging tasks in which the state of the system might not be directly observable, or even not possible to model analytically. While promising, the high-dimensionality of the problem make these methods susceptible to several open challenges. For example, the exploration requirements of reinforcement learning (RL) algorithms are significantly exacerbated for these tasks, due to the high-dimensionality of the observations. This means that trying to learn safe control policies using RL often requires us to accept that abundant failures will occur during training. On the other hand, supervised learning approaches for control, such as behavioral cloning, would in principle seem less prone to exhibit unsafe behaviors. However, it is
In-Distribution Barrier Functions

well known that simply because of their data-dependent nature, these methods are still susceptible to a key challenge named distributional shift: if the trajectories of the system divert from the training data distribution, the controller might take unexpected actions.

On the other hand, the control theory literature extensively covers the problem of long-horizon constraint satisfaction. In particular, Control Barrier Functions (CBFs, Ames et al. (2014)) are a popular model-based tool used to restrict the trajectories of the system from entering undesirable regions of the state-space. One of the properties of CBFs that explain their recent popularity is that they decouple the problem of constraint satisfaction from any performance objective. Specifically, if a CBF is available, then Ames et al. (2014) showed that we can construct a minimally invasive safety filter that transforms into safety-preserving control actions any unsafe commands that an arbitrary reference policy could output.

The main question we want to address in this work naturally emerges from the previous discussion: can we take inspiration from CBFs to avoid out-of-distribution (OOD) states when using data-driven controllers for visuomotor tasks? Even though CBFs are model-based tools that, as such, require knowledge of the state-space and dynamics of the system, the recent advances on learning latent state-space representations and associated dynamics models clearly set a path for linking data-driven visuomotor policy learning with the use of model-based control-theoretic tools such as CBFs.

Contributions: We present an end-to-end self-supervised approach for learning a task-agnostic policy filter which prevents the system from entering OOD states. We do not assume knowledge of the state-space or system dynamics. In addition, our framework only requires an offline-collected dataset of safe demonstrations (where the concept of safety is only linked to the demonstrator’s subjectivity, as it is their responsibility to provide the dataset). We therefore do not require any unsafe demonstrations to learn a safe policy filter, in contrast to most other works tackling constrained policy learning. Furthermore, to the best of our knowledge, this is the first work that uses CBFs for constructing policy filters in learned latent state-spaces. This endows our approach with the flexibility of being applicable to systems with high-dimensional sensory observations, in contrast to most prior CBF-based methods. We present simulation experiments on two different visuomotor control tasks, which suggest that our framework, taking only raw RGB images as input, can learn to significantly reduce the distributional shift from safe demonstrations and, consequently, critically improve the safety of both systems.

2. Related Work

There exist some constructive procedures for synthesizing CBFs based on sum-of-squares programming (Jarvis-Wloszek et al., 2003; Majumdar et al., 2013; Dai and Permenter, 2022; Wang et al., 2022) or Hamilton-Jacobi reachability (Choi et al., 2021). However, these methods require knowledge of the dynamics of the system and typically suffer from scalability issues for high-dimensional systems. More in line with this work, some recent results show that CBFs can be learned from data (Jin et al., 2020; Dawson et al., 2022; Qin et al., 2022; Robey et al., 2020; Lindemann et al., 2021; Abate et al., 2021; Jagtap et al., 2020). None of these works, however, consider systems with high-dimensional observations. Furthermore, the works Jin et al. (2020); Dawson et al. (2022); Qin et al. (2022) assume a priori knowledge of a control-invariant safe set, and focus on building a CBF for that particular set. The line of research of Robey et al. (2020); Lindemann et al. (2021) has the most similar problem setup to our work, as they also consider learning from safe demonstrations.
Although, notably, the authors provide formal verification arguments for the learned CBFs, their methods are not applicable to high-dimensional observations, assume a nominal dynamics model is given, and use an algorithmic approach for the detection of the boundary of the dataset that does not scale to large datasets. Other recent approaches build signed distance functions from sensory measurements that are obtained from a LiDAR or stereo cameras (Long et al., 2021; Srinivasan et al., 2020; Cosner et al., 2022). However, these functions are not encouraged to satisfy any set invariance property.

Extensions of the CBF-based control filters to systems with dynamics or measurement-model uncertainty have also been recently proposed (Nguyen and Sreenath, 2021; Castañeda et al., 2021; Taylor et al., 2021; Dhiman et al., 2021; Dean et al., 2021). These works assume that a CBF is provided, and formulate uncertainty-robust optimization problems for the controller design. They can be considered complementary to our deterministic but end-to-end approach. Future work should explore quantifying uncertainty estimates within our framework to robustify the learned policy filters.

Several existing approaches for OOD prevention learn density models of training data that can be then used to restrict the agent from taking low likelihood actions or moving towards unvisited states (McAllister et al., 2019; Richter and Roy, 2017; Wu et al., 2019; Kumar et al., 2019). Although some of these methods have been shown to be effective at offline RL settings that are specially susceptible to distributional shift, the learned density models have no notion of control invariance and, therefore, do not consider the problem of how to prevent distributional shift over a long time horizon. A notable exception is the work of Kang et al. (2022) to constrain long-term distributional shift, in which a min-max Bellman backup operator is constructed so that Lyapunov-like functions arise as value functions of an offline RL problem. This work however does not consider the extension to visuomotor control tasks in learned latent spaces. Furthermore, for our approach we choose not to rely on a min-max backup operator to learn the certificate function and, instead, use the very suitable theory of CBFs to devise a self-supervised learning framework.

Finally, the work of Wilcox et al. (2022) presents a framework to learn safe sets in a latent state-space for iterative control tasks. Compared to this work, our framework has the advantage that it is task-agnostic and does not require any interactions with the environment during training.

3. Background on Control Barrier Functions

We start by introducing some necessary background on Control Barrier Functions, which are tools from the nonlinear control literature that serve to enforce safety constraints for systems with known dynamics. As will be clear later, CBFs are particularly well-suited for continuous-time nonlinear control-affine systems of the form

\[ \dot{x} = f(x) + g(x)u, \]

where \( x \in \mathcal{X} \subset \mathbb{R}^n \) is the state and \( u \in \mathcal{U} \subset \mathbb{R}^m \) the control input. We assume that \( f : \mathcal{X} \to \mathbb{R}^n \) and \( g : \mathcal{X} \to \mathbb{R}^{n \times m} \) are locally Lipschitz continuous.

In the CBF literature, safety is considered as a set invariance problem. In particular, we say that a control policy \( \pi : \mathcal{X} \to \mathcal{U} \) assures the safety of system (1) with respect to a set \( \mathcal{X}_{\text{safe}} \subset \mathcal{X} \) if the set \( \mathcal{X}_{\text{safe}} \) is forward invariant under the control law \( \pi \), i.e., for any \( x_0 \in \mathcal{X}_{\text{safe}} \), the solution \( x(t) \) of system (1) under the control law \( \pi \) remains within \( \mathcal{X}_{\text{safe}} \) for all \( t \geq 0 \).

**Definition 1 (Control Barrier Function, Ames et al. (2017))** We say that a continuously differentiable function \( B : \mathcal{X} \to \mathbb{R} \) is a Control Barrier Function (CBF) for system (1) with associated safe-set \( \mathcal{X}_{\text{safe}} \subset \mathcal{X} \) if the following three conditions are satisfied:
\[ B(x) \geq 0 \quad \forall x \in X_{\text{safe}}, \]  
\[ B(x) < 0 \quad \forall x \in X \setminus X_{\text{safe}}, \]

\[ \exists u \in U \text{ s.t. } \dot{B}(x, u) + \gamma(B(x)) \geq 0 \quad \forall x \in X, \]

where \( \gamma : \mathbb{R} \to \mathbb{R} \) is an extended class \( \mathcal{K}_{\infty} \) function.

The existence of a CBF \( B \) guarantees that for system (1) any Lipschitz continuous control policy \( \pi \) satisfying

\[ \pi(x) \in \{ u \in U : \nabla B(x)[f(x) + g(x)u] + \gamma(B(x)) \geq 0 \} \]

(5)

will render the set \( X_{\text{safe}} \) forward invariant (Ames et al., 2017, Corollary 2).

For a given task-specific reference controller \( \pi_{\text{ref}} : X \to U \) that might be safety-agnostic, the condition of (5) can be used to formulate an optimization problem that, when solved at every time-step, yields a minimally-invasive policy safety filter (Ames et al., 2014):

\[ \pi_{\text{CBF}}(x) = \arg\min_{u \in U} \| u - \pi_{\text{ref}}(x) \|^2 \quad \text{(CBF-QP)} \]

s.t. \( \nabla B(x)[f(x) + g(x)u] + \gamma(B(x)) \geq 0. \)

Assuming that the actuation constraints that define \( U \) are linear in \( u \), this problem is a quadratic program (QP). This is a consequence of the dynamics of the system (1) being control-affine, and it practically means that the problem can be solved to a high precision very quickly (around \( 10^3 \)Hz). This is critical since the CBF-QP needs to be solved at the real-time control frequency.

4. Problem Statement

The CBF-QP constitutes a very appealing approach for practitioners: it provides a task-agnostic minimally invasive filter that can wrap safety around any given policy \( \pi_{\text{ref}} \), and therefore rewrite any unsafe control input that \( \pi_{\text{ref}} \) could output at any time. However, designing a valid CBF is nontrivial. In fact, it is still an active research topic even when assuming perfect knowledge of the dynamics of the system (Dai and Permenter, 2022; Choi et al., 2021; Wang et al., 2022). The two main difficulties in the design of a CBF are the following: first, a control-invariant set \( X_{\text{safe}} \) must be obtained (which in general is different from the geometric constraint set that could be obtained, for instance, from a signed-distance field) and, second, a function that satisfies condition (4) must be found for that set. Furthermore, even after obtaining a CBF, solving the CBF-QP requires perfect state and dynamics knowledge.

With our framework, we take initial steps towards building a safe policy filter from high-dimensional observations. Specifically, we take inspiration from CBFs to design an end-to-end learning framework to constrain deep learning models to remain in-distribution of the training data. We take as input a dataset of high-dimensional observations of different safe demonstrations, and build a neural CBF-like function that encourages the system to always stay in-distribution with respect to the observations from the safe demonstrations. This, in turn, significantly improves the safety of the system during deployment.

More concretely, for a given dataset of \( N \) safe trajectories \( \mathbb{D} = \left\{ (I_i^t, u_i^t)_{t=0}^{T_i} \right\}_{i=1}^{i=N} \) we tackle the problem of designing a policy filter that can be applied to any reference controller \( \pi_{\text{ref}} \) to detect and override actions from \( \pi_{\text{ref}} \) that lead to OOD states. We denote \( I_i^t \) and \( u_i^t \) the high-dimensional
In this section, we introduce a self-supervised approach for synthesizing neural CBF-like functions whose aim is to constrain the system to remain in-distribution with respect to an offline dataset of safe demonstrations. We call these functions in-Distribution Barrier Functions (iDBFs). We will for now assume that we have a parametric continuous-time control-affine model of the dynamics of the system in a state-space $\mathcal{X} \subset \mathbb{R}^n$

$$\dot{x} = f_\theta(x) + g_\theta(x)u,$$

and present the iDBF learning procedure for this system. Furthermore, for this section we assume that the dataset $\mathcal{D}$ of safe trajectories contains true state measurements, i.e., $\mathcal{D} = \{(x_i^t, u_i^t)_{t=0}^{T_i}\}_{i=1}^{N}$, where $x_i^t$ and $u_i^t$ are the state and control input, respectively, measured at time $t$ for the $i$th trajectory. In Section 6, we will provide details on how to learn a dynamics model of this form in a latent state-space when we have a dataset containing high-dimensional sensory observations.

We parameterize an iDBF $B_\phi : \mathcal{X} \rightarrow \mathbb{R}$ as a neural network with parameters $\phi$, and construct an empirical loss function that encourages it to satisfy the three CBF conditions (2), (3) and (4) with respect to a set $\mathcal{X}_{\text{safe}}$ that is also implicitly learned through self-supervision. To design the loss function, we take inspiration from previous literature on learning CBFs (Dawson et al., 2022; Qin et al., 2022; Chang et al., 2019). Nevertheless, instead of assuming that the safe-set $\mathcal{X}_{\text{safe}}$ is given and that we can sample from it and from its unsafe complement $\mathcal{X}_{\text{unsafe}} = \mathcal{X} \setminus \mathcal{X}_{\text{safe}}$, we build our loss function in a self-supervised manner just from the dataset of safe demonstrations. We accomplish this by leveraging ideas from contrastive learning (Gutmann and Hyvärinen, 2010; Oord et al., 2018; Chopra et al., 2005; Weinberger and Saul, 2009; Schroff et al., 2015). In particular, as we explain in detail later, we build a contrastive distribution from which to sample candidate unsafe states, given that we do not have any unsafe demonstrations in our dataset. The loss function we propose for learning an iDBF takes the following form:

$$L_{\text{iDBF}} = \frac{u_{\text{safe}}}{N_{\text{safe}}} \sum_{x_{\text{safe}}} \left[ \epsilon_{\text{safe}} - B_\phi(x_{\text{safe}}) \right]^+ + \frac{u_{\text{unsafe}}}{N_{\text{unsafe}}} \sum_{x_{\text{unsafe}}} \left[ \epsilon_{\text{unsafe}} + B_\phi(x_{\text{unsafe}}) \right]^+ + \frac{w_{\text{ascent}}}{N_{\text{safe}}} \sum_{(x_{\text{safe}}, u_{\text{safe}})} \left[ \epsilon_{\text{ascent}} - \left( \nabla B_\phi(x_{\text{safe}})[f_\theta(x_{\text{safe}}) + g_\theta(x_{\text{safe}})u_{\text{safe}}] + \gamma(B_\phi(x_{\text{safe}})) \right) \right]^+,$$

where $[\cdot]^+ := \max(0, \cdot)$; $(x_{\text{safe}}, u_{\text{safe}})$ are samples from the empirical distribution of the dataset $\mathcal{D}$; $x_{\text{unsafe}}$ are samples from a contrastive distribution that we will define soon; $w_{\text{safe}}, w_{\text{unsafe}}$ and $w_{\text{ascent}}$ are the weights of the different loss terms; and $\epsilon_{\text{safe}}, \epsilon_{\text{unsafe}}$ and $\epsilon_{\text{ascent}}$ are positive constants that serve to enforce strictly the inequalities and generalize outside of the training data.

The goal of the first two terms in the loss function is to learn an iDBF that has a positive value in states that belong to the data distribution of safe demonstrations, and negative everywhere else (meaning we are encouraging the satisfaction of conditions (2) and (3) of the definition of CBF).
Note that this classification objective is very related to the notion of energy-based models (EBMs)—neural network density models that assign a low energy value to points close to the training data distribution and a high value to points that are far from it (Hinton, 2002). In fact, we took inspiration from the Noise Contrastive Estimation (NCE, Gutmann and Hyvärinen (2010)) training procedure of EBMs, in particular the InfoNCE loss (Oord et al., 2018), to design (7). Intuitively, these methods use a noise contrastive distribution to generate candidate examples where to increase the value of the energy of the EBM, while decreasing the energy at the training data points. We precisely want the opposite result for our problem: a high value of $B_{\phi}$ on the training data distribution, and a low value everywhere else. However, we have one additional requirement, which is that the iDBF should have a value of zero at the boundary, as set by conditions (2) and (3). This is the reason why we design the two first terms of the loss function using the $[\cdot]^+$ operator.

In the third term of the loss (7), note that we do not encourage the satisfaction of condition (4) over the entire state-space, but only over the dataset of safe demonstrations. However, in the definition of CBF, if condition (4) is only satisfied $\forall x \in X_{\text{safe}}$ instead of $\forall x \in X$, the CBF still guarantees the control-invariance of $X_{\text{safe}}$. We are therefore using our empirical data distribution of safe demonstrations as a sampling distribution covering the set $X_{\text{safe}}$, which we are also implicitly learning as the zero-superlevel set of $B_{\phi}$. Furthermore, compared to prior approaches that encourage the satisfaction of condition (4) for a single policy (Dawson et al., 2022; Qin et al., 2022), we instead use all pairs $(x_{\text{safe}}, u_{\text{safe}})$ present in the dataset $D$ to compute this term of the loss. This way, we force the set of admissible control inputs (5) to be as large as our dataset allows, reducing the conservatism of the learned iDBF.

In order to generate the contrastive distribution from which to sample $x_{\text{unsafe}}$, as we ultimately want to learn the iDBF in a latent state-space in which it might not be intuitive how to construct a noise distribution, we take the following steps. 1) Based on the dataset of safe demonstrations $\mathcal{D}$, we train a neural behavioral cloning (BC) model that outputs a multi-modal Gaussian distribution over actions conditioned on the state, with density $\pi_{\text{BC}}(u|x)$. 2) Then, during the training process of the iDBF, for each $x_{\text{safe}}$ state sampled from $\mathcal{D}$ we randomly take $N_{\text{candidate}}$ control inputs $u_{\text{candidate}}$ and evaluate their density value based on the BC model $\pi_{\text{BC}}(u_{\text{candidate}}|x_{\text{safe}})$. 3) If the value of the density falls below a threshold, then that control input is forward-propagated for one timestep using the dynamics model (6) to generate a sample $x_{\text{unsafe}}$. This way, we generate a contrastive data distribution by propagating actions that are unlikely present in the dataset of safe demonstrations. Furthermore, by only propagating these actions for one timestep, the contrastive distribution is close to the training data, which is desirable for the learning process (Gutmann and Hirayama, 2011).

6. Learning iDBFs from High-Dimensional Observations

After introducing the training procedure for an iDBF when the state representation and dynamics model (6) are given, we now relax these assumptions and present an approach to learn a latent state-space representation and a continuous-time dynamics model of the form (6), suitable to be integrated in the same end-to-end learning framework. We therefore now consider precisely the problem setting described in Section 4, in which we only assume having access to a dataset containing observation-action pairs of safe demonstrations $\mathcal{D} = \{(I_i, u_i)_{t=0}^{T_i} \}_{i=1}^{N}$.

We use an autoencoder architecture to obtain the latent state-space representation, and employ the training procedure of Neural Ordinary Differential Equations (Neural ODEs, Chen et al. (2018)) to learn a dynamics model of the form (6) in the latent state-space. Note that by enforcing the
continuous-time control-affine structure of the dynamics model, we ensure that the iDBF-QP policy filter (equivalent to the CBF-QP, see Figure 1) obtained with the learned iDBF and dynamics model will also be a quadratic program.

The inference procedure of our end-to-end learning framework is depicted in Figure 1. We use a recursive encoder network $E_{\psi}$ that takes the current measurement $I_k$, as well as the previous latent state $x_{k-1}$ and action $u_{k-1}$ to generate the new latent state $x_k$ at each time-step $k$. The decoder network $D_{\xi}$ generates a reconstructed observation $\hat{I}_k$ for each latent state $x_k$. The proposed loss function for the latent state-space representation and dynamics model penalizes both the prediction error of the dynamics model and the observation reconstruction error:

$$
\mathcal{L}_\text{dyn} = \frac{1}{N_{\text{dyn}}(T_{\text{pred}} + 1)} \sum_{j=1}^{N_{\text{dyn}}} \sum_{k=0}^{T_{\text{pred}}} w_{\text{state}} \left\| \hat{x}_{t_j+k|t_j} - x_{t_j+k} \right\|^2 + w_{\text{rec}2} \left\| \hat{I}_{t_j+k|t_j} - I_{t_j+k} \right\|^2 + w_{\text{rec}2} \left\| \hat{I}_{t_j+k} - I_{t_j+k} \right\|^2.
$$

(8)

Here, $x_{t_j+k} = E_{\psi}(I_{t_j+k}, x_{t_j+k-1}, u_{t_j+k-1})$ is the latent state at timestep $t_j + k$. $\hat{x}_{t_j+k|t_j}$ denotes the latent state prediction obtained by forward-propagating the dynamics model (6) to timestep $t_j + k$ starting from the state $x_{t_j}$ and using zero-order hold on the sequence of control inputs $(u_{t_j}, u_{t_j+1}, \ldots, u_{t_j+k-1})$. Additionally, $\hat{I}_{t_j+k|t_j} := D_{\xi}(\hat{x}_{t_j+k|t_j})$ is the reconstructed observation from the dynamics prediction for timestep $t_j + k$. Finally, $\hat{I}_{t_j+k} := D_{\xi}(x_{t_j+k})$ is the encoded-decoded observation at timestep $t_j + k$.

Note that we use a multiple-shooting error for the loss (8), as the prediction horizon $T_{\text{pred}}$ does not need to coincide with the length of the trajectories in the dataset $\mathbb{D}$. In particular, the loss (8) is computed by sampling a batch of trajectories from $\mathbb{D}$ and then splitting them into $N_{\text{dyn}}$ portions of length $T_{\text{pred}}$. The initial timestep of each portion $j = 1, \ldots, N_{\text{dyn}}$ is denoted as $t_j$. The first two terms in the loss function are then penalizing the state and reconstruction error of the multistep predictions of the dynamics model from each initial state $x_{t_j}$. The last term in the loss function penalizes the reconstruction error of the autoencoder directly, without using the dynamics model. The recent results of Beintema et al. (2021) show that multiple shooting loss functions lead to more accurate predictions compared to single-step prediction losses, and to better conditioned learning problems compared to single-shooting propagation losses.

An iDBF can be learned together with the autoencoder and dynamics model by optimizing jointly the losses (7) and (8). For the iDBF loss, each $x_{\text{safe}}$ is obtained by encoding the observations
In this section, we present the empirical evaluation of our framework on two different simulation environments: a toy example of a robot navigation task using top-down images of the scene, and an autonomous driving scenario with egocentric image observations. For both cases, given a safety-agnostic reference controller \( \pi_{\text{ref}} \), we use our iDBF-QP at each timestep with the latest image measurement to find the closest control input to \( \pi_{\text{ref}} \) among those that prevent the system from entering OOD states (see Figure 1). For each environment, we train the iDBF, autoencoder and dynamics model using a dataset containing 64 \( \times \) 64 RGB images of offline-collected trajectories.

**Robot Navigation with Top-Down View Images:** In this example, a circular robot with radius of 1 meter navigates inside of a 10 \( \times \) 10 meter room that has a square-shaped 4 \( \times \) 4 meter static obstacle in the middle, as shown in Figure 2 (left). The underlying dynamics of the robot are those of a 2D single integrator, with two control inputs corresponding to the \( x \) and \( y \) velocity commands, although we do not assume having access to that knowledge. Instead, we only have a dataset of image-action pairs corresponding to 5000 trajectories of 100 points each (corresponding to 2 seconds since the time-step is 0.02s). These trajectories satisfy two requirements: 1) the robot should never collide against the obstacle, and 2) the center of the robot should never leave the room limits. The trajectories are collected applying random actions at each time-step, and we check both conditions before adding a trajectory to the dataset. We use our framework to train an autoencoder with

---

**Table 1:** Evaluation of the collision rate and cumulative filter intervention (a measure of how intrusive the filter is with respect to the reference controller) for the top-down view robotic navigation example (over 20 simulations of 5-seconds each with random initial and goal states) and for the egocentric view autonomous driving example (over 20 simulations of 50-seconds each with random initial heading angles). For the BC and ensemble filters, we provide results for 3 different threshold values: \((\delta_{\text{low}}, \delta_{\text{mid}}, \delta_{\text{high}}) = (0.32, 0.35, 0.38)\) for the navigation example, and \((0.2, 0.5, 0.8)\) for driving; and \((\delta_{\text{low}}, \delta_{\text{mid}}, \delta_{\text{high}}) = (0.0005, 0.001, 0.002)\) for both examples.

<table>
<thead>
<tr>
<th></th>
<th>( \pi_{\text{ref}} )</th>
<th>Ours</th>
<th>BC Filter</th>
<th>Ensemble Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \delta_{\text{low}} )</td>
<td>( \delta_{\text{mid}} )</td>
<td>( \delta_{\text{high}} )</td>
<td>( \delta_{\text{low}} )</td>
</tr>
<tr>
<td><strong>Top-Down Navigation</strong></td>
<td>Collision Rate (%)</td>
<td>0.28 \pm 0.27</td>
<td>0.28 \pm 0.27</td>
<td>0.28 \pm 0.27</td>
</tr>
<tr>
<td></td>
<td>Cumulative Intervention</td>
<td>0.0 \pm 0.0</td>
<td>0.0 \pm 0.0</td>
<td>0.0 \pm 0.0</td>
</tr>
<tr>
<td><strong>Egocentric Driving</strong></td>
<td>Collision Rate (%)</td>
<td>21.94 \pm 1.85</td>
<td>14.44 \pm 2.69</td>
<td>8.78 \pm 1.86</td>
</tr>
<tr>
<td></td>
<td>Cumulative Intervention</td>
<td>278.1 \pm 32.6</td>
<td>713.8 \pm 1.4</td>
<td>726.7 \pm 2.6</td>
</tr>
</tbody>
</table>

---
Latent state-space of dimension 3, a dynamics model, and an iDBF. The reference policy $\pi_{\text{ref}}$ simply applies a velocity in the direction of a goal-point, with magnitude proportional to the distance. In Figure 2, we show the results of applying our iDBF-QP when the goal state (marked with an $\times$) is outside of the room limits and at the other side of the obstacle. Even though the reference controller is trying to take the shortest path, which would go through the obstacle, the iDBF-QP prevents the robot from first, colliding with the obstacle, and second, from having its center exit the room limits.

Autonomous Driving with Egocentric View Images: We use the environment provided by Kahn et al. (2018), which is based on theBullet physics simulator and the Panda3d graphics engine (Goslin and Mine, 2004) to obtain egocentric RGB image measurements. The car navigates in a corridor which has four 90-degree turns to form a square-shaped center-line. One of such turns is shown in the snapshots of Figure 3. The car has two control inputs: the desired forward velocity and the steering angle. Given the high-order dynamics of the simulator, we collect data manually to make sure no trajectories included in the dataset are deemed to collide with any of the walls. We split the collected data into 450 trajectories of 100 points each (5 seconds since the timestep is 0.05s). This makes for a much sparser and less diverse (since it is collected by a human) dataset compared to the previous example. During deployment, we use a reference controller $\pi_{\text{ref}}$ that simply drives the car forward at a constant speed of 3.5m/s. Our iDBF-QP framework of Figure 1, taking the latest egocentric RGB measurement as input, is very effective at preventing the car from colliding against the walls, as shown in Table 1. Figure 3 contains snapshots of our iDBF-QP forcing the car to take a turn as it approaches a corner, even though the reference command is to drive forward.

Using these simulation environments we also aim to compare our proposed approach with other techniques for avoiding distributional shift. Other works that consider this problem use data density models to constrain the learned policies (Richter and Roy, 2017; McAllister et al., 2019; Wu et al., 2019), or use uncertainty estimation schemes, such as ensemble models, to avoid taking actions that lead to highly uncertain states (Chua et al., 2018). We build our baselines upon a conditional BC density model of the training data and an ensemble of latent state-space dynamics models:

**BC Density Filter Baseline:** As explained in Section 5, we train a BC multi-modal Gaussian model that is used to generate the contrastive training distribution for the iDBF. For any state, the BC model outputs a probability distribution over actions, with density function $\pi_{\text{BC}}(u|x)$. We train this BC model using privileged true-state information of the system, and use its density values to build a filter that serves as an apples-to-apples baseline comparison to our approach. Specifically, the baseline also takes the reference controller $\pi_{\text{ref}}$ and, at every timestep, it finds the closest control
action to $\pi_{\text{ref}}(x)$ that satisfies $\pi_{\text{BC}}(u|x) \geq p$, out of 200 randomly sampled actions. If no control action satisfying that condition is found, the reference control input is applied without filtering. Given the clear dependence on the threshold value $p$, we implement this baseline for several values of $p$ and show the results in Table 1 for three representative cases $p_{\text{low}}$, $p_{\text{mid}}$, and $p_{\text{high}}$.

**Ensemble Variance Filter Baseline:** We also train an ensemble of independent latent state-space dynamic models ($f_\theta$ and $g_\theta$), keeping the rest of the framework introduced in Section 6 unchanged. During deployment, at every timestep we look for the closest control action to $\pi_{\text{ref}}(x)$ that keeps the variance $\sigma^2_{\text{ens}}(x,u)$ of the predicted dynamics $f_\theta(x) + g_\theta(x)u$ under a threshold $\delta$. As in the previous baseline, we also look over 200 randomly sampled actions at each timestep, and different threshold levels $\delta_{\text{low}}$, $\delta_{\text{mid}}$, and $\delta_{\text{high}}$. Again, if no control action satisfying the threshold condition is found, the reference control input is applied without filtering.

In Table 1, we provide a summary of the comparison results for both environments. We use the collision rate as a proxy for distributional shift, since the training data only includes collision-free trajectories. The collision rate for the robot navigation example is computed as the fraction of time that the robot spends either in collision with the obstacle or having its center-point outside of the room limits. For the driving scenario, the collision rate is the fraction of time that the robot is in collision with any of the walls. For both examples, our method drastically reduces the collision rate compared to using the reference (unfiltered) controller. Furthermore, we achieve the lowest collision rates when compared to the baselines. From the baselines, only the BC density filter (with a very restrictive threshold $p_{\text{high}}$) manages to achieve small collision rates, at the cost of a very high cumulative filter intervention rate. The filter intervention rate is computed for both examples as $\sum_t \|u_t - \pi_{\text{ref}}(x_t)\|^2$, where each control input dimension is normalized between $-1$ and $1$.

**8. Conclusion**

In this work, we take first-steps towards merging control-theoretic CBFs with practical robotic tasks that involve high-dimensional perception modules. We consider a realistic problem setting in which no unsafe demonstrations are available, and take a self-supervised learning approach to learn a function that effectively restricts the system from diverging towards OOD states. By learning this function in a latent state-space, our framework should be flexible-enough to be applicable to a wide variety of visuomotor tasks, and should be compatible with the use of large-scale pretrained representation learning models. Another important direction for future work would be to use probabilistic encoding and dynamics models to be able to robustify our proposed filters with respect to prediction uncertainties. Additionally, exploring the use of loss functions that are not based on reconstruction, by exploiting the value function nature of the iDBF, could be another promising direction.
Acknowledgments

The authors would like to thank Dr. Jean Mercat, Dr. Hongkai Dai and Dr. Katherine Liu for their insightful comments and suggestions.

References


IN-DISTRIBUTION BARRIER FUNCTIONS


