EigenSafe: A Spectral Framework for Learning-Based Stochastic Safety Filtering

Inkyu Jang
Seoul National University
janginkyu.larr@gmail.com

Jonghae Park
Seoul National University
bdfire1234@snu.ac.kr

Chams E. Mballo
UC Berkeley
cmballo@berkeley.edu

Sihyun Cho
Seoul National University
sihyun01@snu.ac.kr

Claire J. Tomlin
UC Berkeley
tomlin@eecs.berkeley.edu

H. Jin Kim
Seoul National University
hjinkim@snu.ac.kr

Abstract: We present EigenSafe, an operator-theoretic framework for learning-enabled safety-critical control for stochastic systems. In many robotic systems where dynamics are best modeled as stochastic systems due to factors such as sensing noise and environmental disturbances, it is challenging for conventional methods such as Hamilton-Jacobi reachability and control barrier functions to provide a holistic measure of safety. We derive a linear operator governing the dynamic programming principle for safety probability, and find that its dominant eigenpair provides information about safety for both individual states and the overall closed-loop system. The proposed learning framework, called EigenSafe, jointly learns this dominant eigenpair and a safe backup policy in an offline manner. The learned eigenfunction is then used to construct a safety filter that detects potentially unsafe situations and falls back to the backup policy. The framework is validated in three simulated stochastic safety-critical control tasks.

Keywords: Robot Safety, Safety-Critical Control, Stochastic Systems

1 Introduction

A key challenge in the deployment of autonomous robotic systems is ensuring safe operation. This is typically formalized as requiring that, either deterministically or with high probability, the system's state remains within a safe subset of the state space. Current methods for ensuring safe operation of robots, such as Hamilton–Jacobi (HJ) reachability analysis [1] and control barrier functions (CBFs) [2], are particularly effective for deterministic systems for which a model is available. Such models are necessary for reliable characterization of the range of behaviors of the system. However, many complex real-world robotic systems are stochastic in nature, especially with the rise of learning-enabled components.

This stochasticity in robotic systems arises from several factors: environment-induced randomness, sensing limitations, and modeling uncertainty arising from learning-based components. The environment can inject stochastic disturbances into the system through both random external forces, such as wind or contact forces, which are inherently stochastic processes, and actuator or sensor noise caused by effects such as the photoelectric effect, thermal fluctuations, or electromagnetic interference. Robotic systems rely on sensors to measure both the external environment and their internal state for functions such as mapping, navigation, and control. Due to their inherent limitations, such as latency in measurement and limited sensing bandwidth, a large portion of the robot's state is unobservable or, at best, available through noisy measurements, which adds additional uncertainty into the robot dynamics. Another source of stochasticity is from learning-enabled components which are often employed in modern robotic systems. Such components introduce model uncertainty due to learned model errors, which can stem from limited model capacity; a representation mismatch between the learned and real models; or insufficient training data, which is a particularly common issue in robotics where hardware experiments are costly and data collection is slow.

Workshop on Safe and Robust Robot Learning for Operation in the Real World at CoRL 2025.

Various optimization-based approaches have been developed, including extensions of HJ reachability and CBFs, to handle real-world stochastic systems; see, for example, the surveys [3] on HJ reachability in reinforcement learning and [4] on data-driven and learning-based safety filters. These methods and their extensions to stochastic systems are particularly effective in providing a probabilistic measure of safety for any given state (or state-action pairs) in the state space, thus enabling probabilistic comparisons of safety across different initial states. Nevertheless, they fail to provide a measure of the safety of the overall closed-loop system's long-term behavior, which is crucial to capture in stochastic systems. It is not only important to determine how safe a state is relative to other states in the state space, but also to quantify the decay rate of the safety probability of the closed-loop system under a given policy from a given initial state.

In this work, we present **EigenSafe**, an operator-theoretic framework that provides a more holistic characterization of safety for stochastic systems than existing methods. Our presentation in this paper focuses on the key theoretical ideas and their implications rather than full technical proofs, to emphasize the conceptual underpinnings and practical applicability of the proposed method. In **EigenSafe**, we derive the dynamic programming principle for safety probability and show that it is governed by a linear operator, whose dominant (largest in absolute value) eigenvalue and associated eigenfunction encode critical safety information. The eigenvalue represents the overall safety of the closed-loop system, while the eigenfunction quantifies relative safety of each state–action pair. Together, they provide a safety measure explicitly tied to the safety probability. The eigenfunction value especially serves a role analogous to a safety Q function. We propose a loss functional that allows joint learning of this dominant eigenpair together with a safe backup policy, in a model-free and offline manner. The learned eigenfunction is used to formulate a safety filter that detects when the action is potentially unsafe and falls back to the backup policy. The proposed safety filter is tested in three simulated dynamical systems.

1.1 Related Work

We review relevant works that address safety-critical control problems for stochastic systems from the traditional control and data-driven learning perspectives.

Safety-Critical Control. The design of control architectures that guarantee the safe operation of safety-critical systems has received extensive research attention. Among the state-of-the-art approaches, HJ reachability [1, 5] and CBFs [2] are widely studied, with safety typically encoded as inequality constraints on the state space. Although originally developed for deterministic systems, both methods have since been extended to stochastic settings, taking into account various notions of safety measures, e.g., finite-time safety probability [6, 7, 8], conditional value-at-risk [9, 10], etc [11]. Such measures are useful for assessing the safety of individual states (or state–action pairs), but they are limited in evaluating the overall safety of the closed-loop system over long horizons. Similar to the proposed method, martingale-based approaches, such as stochastic CBFs, provide time-varying probabilistic guarantees, typically showing that the safety probability is lower bounded by an exponentially or linearly decaying function of time [12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. The decay rate here could reflect the closed-loop safety of the system, but the resulting probability bounds are often overly loose. These methods often suffer from poor scalability in terms of computational burden with respect to the system's dimensionality. The use of physics-based learning methods has been shown to be a viable solution to this [22, 23].

Learning Safety Certificates from Data. Beyond model-based methods, a rapidly growing body of work combines safety-critical control methods with learning methods to approximate the safety value function or reachable sets, with already a number of good survey papers on this topic [3, 24, 25, 26]. CBF learning methods typically require knowledge about the dynamics model, either completely accurate [27, 28, 29] or at least approximate [30, 31]. Reinforcement learning (RL) methods can be used to solve the optimal control problem associated with HJ reachability [32, 33, 34]. Some works solve an RL problem subject to reachability constraints, e.g., [35, 36]. One can leverage the strengths of learning-based methods to address more complex or learned safety constraints [37, 38, 39, 40, 41, 42]. More recently, with world models gaining popularity [43, 44,

45, 46], many works perform RL-based reachability analysis with respect to the latent dynamics and learned safety constraints. The mentioned methods, however, are based on reachability analysis for deterministic systems and fail to capture the stochastic nature of learned models, especially with latent spaces. There are a few works that explicitly address stochasticity and safety probability using the similar optimal control formulation [47, 48]. However, the infinite-horizon safety probability they compute is typically zero everywhere unless the system can be made almost-surely safe, making it hard to compare relative safety across different states.

2 Problem Setup

2.1 Safety-Critical Dynamics

The dynamics considered in this paper is modeled as a discrete-time Markov decision process (MDP) without the reward term, a 3-tuple (S, A, P), where S, A are the state and action spaces, and P is the transition probability that describes the dynamics:

$$s_{t+1} \sim P(\cdot|s_t, a_t). \tag{1}$$

Here, $s_t, s_{t+1} \in S$ are the system's state at time t and $t+1$, respectively, and $a_t \in A$ is the action taken at time t.

Consider a safety constraint $s_t \in C$, where $C \subsetneq S$ is the *safe set*: For example, C could represent the set of states satisfying a collision-avoidance constraint. Once the system leaves C for the first time, a permanent safety failure is recorded, and all subsequent system behavior is considered unsafe. We model this using an *unsafe (termination) state* $K \notin C$. If $s_t \notin C$, then the trajectory is marked as terminated by setting $s_{t'} = K$ for all $t' \geq t$, regardless of the action taken thereafter. Since we do not distinguish between different failure modes, we consider the existence of only one unsafe state and write the state space as $S = C \cup \{K\}$. The safety probability of the system can then be expressed as $\mathbb{P}[s_t \neq K]$, which is the probability that the system remains safe at time t. This probability is non-increasing in time (i.e., $\mathbb{P}[s_{t+1} \neq K] \leq \mathbb{P}[s_t \neq K]$) and depends on the chosen (feedback) policy and the initial state. Although a policy that keeps the system almost surely safe ($\mathbb{P}[s_t \neq K] = 1$) may not exist, the problem of interest in this paper is to learn a policy or safety filter that maintains the safety probability above a prescribed threshold.

2.2 Dynamic Programming for Safety Probability

Let π be either a deterministic or stochastic policy such that $a_t \sim \pi(\cdot|s_t)$ for all $t \geq 0$. Under this policy, we define the safety probability $Z_\pi(t, x)$ as the probability that the system is in the safe set at time t when starting from the initial state $x \in S$:

$$Z_\pi(t, x) \coloneqq \mathbb{P}_\pi[s_t \neq K | s_0 = x], \tag{2}$$

where $\mathbb{P}_\pi$ denotes the probability measure under the policy π.

From the law of total probability and the Markov property, one can easily see that the $(t+1)$-step safety probability from the initial condition $s_0 = x$, denoted $Z_\pi(t+1, x)$, can be expressed as the expectation of the t-step safety probability from the next state $s_1 = x^+$, $Z_\pi(t, s_1)$. This yields the following dynamic programming principle: for any policy π, $t \in \{0, 1, \cdots, \}$, and $x \in S$,

$$Z_\pi(t+1, x) = \mathbb{E}_{u \sim \pi(\cdot|x),\ x^+ \sim P(\cdot|x,u)} \left[Z_\pi(t, x^+) \right]. \tag{3}$$

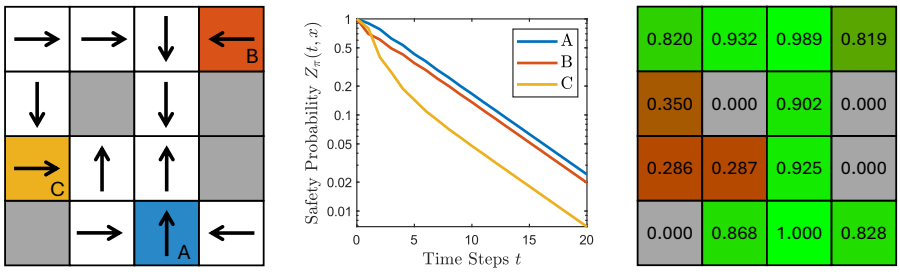
Let $(\mathcal{F}, \|\cdot\|_\infty)$ be a Banach space of scalar functions on S, i.e., $\mathcal{F} : \{\beta : S \to \mathbb{R}\}$, where $\|\cdot\|_\infty$ is the supremum norm defined as $\|\beta\|_\infty = \sup_{x \in S} |\beta(x)|$. We define the operator $T_\pi : D_T \subset \mathcal{F} \to \mathcal{F}$ as

$$T_\pi \beta(x) \coloneqq \mathbb{E}_{u \sim \pi(\cdot|x),\ x^+ \sim P(\cdot|x,u)} [\beta(x^+)], \tag{4}$$

where the domain of T_π is $D_T \coloneqq \left\{ \beta : S \to \mathbb{R} \middle| \beta(K) = 0 \right\}$. We can further express T_π as the integral[1]

$$T_\pi \beta(x) = \int_S \beta(y) p_\pi(y|x) dy = \int_C \beta(y) p_\pi(y|x) dy, \tag{5}$$

[1]The integrals hereafter should be summations in case of discrete state or action spaces.



(a) (b) (c)

Figure 1: **A toy example describing the meaning of the dominant eigenpair of T_π.** (a) This toy example consists of a finite state space represented by the non-gray cells, and a finite action space represented by an arrow on each of the cells. At every time step, the system jumps to the adjacent cell in the arrow direction with probability 0.6, and to another adjacent cell or does not move with probability 0.1. If the system reaches a gray cell or escapes the map, it is considered unsafe and marked as terminated, i.e., reached the unsafe state K. (b) The safety probability given initial conditions A, B, C, corresponding to the colored cells in (a). Note that the vertical axis is log-scale and the slopes of all three curves converge to the same value, which corresponds to the dominant eigenvalue γ_π. (c) The dominant eigenfunction values. It can be seen that cells with higher values correspond to safer points. For finite-state systems, one can directly perform eigendecomposition of the matrix representation of T_π to obtain the eigenpair.

where $p_\pi(\cdot|x)$ is the probability measure on S induced by the policy π, with probability density function $p_\pi(y|x) \coloneqq \int_A \pi(u|x)P(y|x,u)du$, which can also be understood as the density of the next state y given the current state x and policy π. Note that second equality holds since $\beta(x) = 0$ outside the set C. One can also find that the image of T_π is a subset of D_T, making T_π an endomorphism (a mapping from D_T onto itself). This follows from the fact that the system cannot recover from an unsafe state; hence, $\beta \in D_T$ implies $T_\pi\beta(x) = 0$ for all $x \notin C$. Combined with the initial condition $Z_\pi(0,x) = 1_C(x)$, the dynamic programming principle (3) allows one to evaluate $Z_\pi(t,x)$ by applying T_π t times to the indicator function:

$$Z_\pi(t,x) = T_\pi^t 1_C(x) = \underbrace{T_\pi \circ \cdots \circ T_\pi}_{t \text{ times}} 1_C(x). \tag{6}$$

2.3 Spectral Properties of T_π

It is evident from the definition that regardless of π, T_π is a *linear* operator. That is, if β_1 and β_2 are two scalar functions on S and c_1 and c_2 are scalars, then $T_\pi(c_1\beta_1 + c_2\beta_2) = c_1T_\pi\beta_1 + c_2T_\pi\beta_2$. Moreover, regardless of π, T_π is a nonnegative operator, meaning that if $\beta : S \to \mathbb{R}$ is nonnegative everywhere, then so is $T_\pi\beta$. The operator T_π can also be understood as the restriction of the Koopman operator [49] of the closed-loop system to the domain D_T which is a vector subspace of the set of all measurable scalar functions on S.

As T_π is a linear operator, we can now examine its spectral properties. Firstly, we find that T_π is non-expansive with respect to the supremum norm $\|\beta\|_\infty \coloneqq \sup_{x\in C} |\beta(x)|$, because for all $x \in C$,

$$|T_\pi\beta(x)| = \left|\int_C \beta(y)p_\pi(y|x)dy\right| \le \int_C |\beta(y)|p_\pi(y|x)dy \le \|\beta\|_\infty \int_C p_\pi(y|x)dy \le \|\beta\|_\infty . \tag{7}$$

This means that every eigenvalue of T_π has an absolute value of less than or equal to 1. Also, Perron-Frobenius theorem says that the biggest eigenvalue in absolute value of the nonnegative linear operator T_π is positive real, which we call the dominant eigenvalue and denote by γ_π. Moreover, the eigenfunction (i.e., the eigenvector) corresponding to γ_π is also positive. That is, there exists a function $\phi_\pi : S \to \mathbb{R}_{\ge 0}$ such that $\phi_\pi(x) \neq 0$ for at least one $x \in C$, and $T_\pi\phi_\pi(x) = \gamma_\pi\phi_\pi(x)$, for all $x \in S$.

Usually, the eigenvalue γ_π has multiplicity 1, meaning that such ϕ_π is unique up to scalar multiplication. In that case, once t becomes large enough so that the faster-decaying modes vanish, then

$Z_\pi(t,x)$ becomes approximately proportional to $\phi_\pi(x)$ and γ_π^t. For any $\beta \in D(T_\pi)$, there exists a constant $c \in \mathbb{R}$ such that

$$T_\pi^t \beta(x) = c \cdot \phi_\pi(x) \gamma_\pi^t + o(\gamma_\pi^t). \tag{8}$$

This means that the asymptotic decay rate of Z_π is equal to γ_π and uniform across the state space, The eigenfunction value $\phi_\pi(x)$ measures how relatively safe a given state x is compared to other states in C. That is, γ_π is a safety measure for the closed-loop system itself, ϕ_π is the safety measure for the given state. See Fig. 1 for a simple illustrative example.

2.4 The Safety Q Function

In this subsection, we extend the discussion regarding the spectral properties of T_π to compare safety between different actions. For that, we introduce a Q function for the safety probability Z_π, which is defined as follows:

$$Q_\pi(t,x,u) \coloneqq \mathbb{P}_\pi \left[s_t \notin C \middle| s_0 = x, a_0 = u \right], \tag{9}$$

where $\mathbb{P}_\pi$ here measures the probability of the event assuming $a_k \sim \pi(\cdot|s_k)$ for all $k \in \{1, \cdots, t-1\}$. Similar to Z_π, Q_π should have nonnegative values not greater than 1, and

$$\mathbb{E}_{u \sim \pi(\cdot|x)}[Q_\pi(t,x,u)] = Z_\pi(t,x). \tag{10}$$

Another interpretation of Q_π function is the safety probability of the *augmented* autonomous system with state $\hat{s}_t \coloneqq (s_t, a_t)$ defined as

$$s_{t+1} \sim P(\cdot|s_t, a_t), \quad a_{t+1} \sim \pi(\cdot|s_{t+1}) \tag{11}$$

whose safety characteristics is the same to that of the original closed-loop system under policy π. Similar to T_π, we can define the operator

$$A_\pi \beta(x,u) \coloneqq \mathbb{E}_{x^+ \sim P(\cdot|x,u), u^+ \sim \pi(\cdot|x^+)} \left[\beta(x^+, u^+) \right] \tag{12}$$

on the domain $D_A \coloneqq \left\{ \beta : S \times A \to \mathbb{R} \;\middle|\; \beta(K,a) = 0,\ \forall a \in A \right\}$, and derive the similar dynamic programming principle $Q_\pi(t+1,x,u) = A_\pi Q_\pi(t,x,u)$[2] and

$$Q_\pi(t,x,u) = A_\pi^t 1_{C \times A}(x,u) = \underbrace{A_\pi \circ \cdots \circ A_\pi}_{t \text{ times}} 1_{C \times A}(x,u). \tag{13}$$

Since A_π is also a nonnegative linear operator and the safety characteristics of (11) are the same as the original closed-loop system, it will have the dominant eigenpair (γ_π, ψ_π) such that $A_\pi \psi_\pi(x,u) = \gamma_\pi \psi_\pi(x,u)$ for all $(x,u) \in S \times A$, with $\gamma_\pi \in [0,1]$ being the same to the dominant eigenvalue of T_π, ψ_π is a nonzero, nonnegative function on $S \times A$. We can also say $\mathbb{E}_{u \sim \pi(\cdot|x)} \psi_\pi(x,u) = \phi_\pi(x)$, after appropriately scaling ψ and ϕ. The eigenfunction ψ_π quantifies the safety level of a given state-action pair (x,u), and can be used to train the policy π so that the eigenvalue γ_π, i.e., the overall safety of the system, improves.

3 The EigenSafe Framework

The proposed EigenSafe framework consists of two phases. In the offline phase, a safe backup policy π and (γ_π, ψ_π), the eigenpair of A_π are jointly learned. In the online phase, a safety filter based on the eigenfunction ψ_π is employed to prevent any potentially dangerous action given from the user of the higher-level decision maker.

3.1 Offline Phase: Jointly Learning the Eigenfunction and the Backup Policy

Suppose a dataset $\mathcal{D}$ consisting of state-action-next state tuples is given, where for all $(x,u,x^+) \in \mathcal{D}$, $x \in C$ but x^+ might be K. This is a natural assumption because once the system has reached the unsafe state K during data collection, then the trajectory cannot proceed further. EigenSafe in

[2]Here, the operator A_π is applied to the *curried* form of the Q function $Q(t,\cdot,\cdot) : (x,u) \mapsto Q(t,x,u)$.

its simplest form parametrizes the eigenfunction ψ_π as a table of values (for discrete state and action spaces) or a neural network (for continuous spaces), and minimizes the following loss functional:

$$\mathcal{J}_{\text{eig}}[\lambda, \psi] = \frac{W_\lambda}{|\mathcal{D}|} \sum_{(x,u,x^+)\in\mathcal{D}} \big(\psi(x^+, u^+) - \lambda\psi(x, u)\big)^2 + W_n \cdot \left(\max_{(x,u,\cdot)\in\mathcal{D}} \psi(x, u) - 1\right)^2 \tag{14}$$

where u^+ is drawn from the policy $\pi(\cdot|x^+)$, W_λ and W_n are positive weights which can be tuned, and $\psi(x^+, u^+)$ returns 0 if $x^+ = K$. This loss function is closely related to power iteration. Similar to RL where a discount rate less than 1 ensures convergence when the Bellman operator is repeatedly applied, since the power iteration

$$\lambda \leftarrow \|A_\pi\psi\|_\infty, \quad \psi \leftarrow \frac{A_\pi\psi}{\|A_\pi\psi\|_\infty} \tag{15}$$

converges to the dominant eigenpair almost surely if the initial guess has nonzero component in the eigenfunction direction, we can expect that optimizing (14) also converges to the dominant eigenpair with proper normalization $\|\psi\|_\infty = 1$. Upon that, an additional loss term

$$\mathcal{J}_+[\psi] = \frac{W_+}{|\mathcal{D}|} \sum_{(x,u,\cdot)\in\mathcal{D}} \text{ReLU}(-\psi(x, u)) \tag{16}$$

could be added to $\mathcal{J}_{\text{eig}}$ in order to facilitate fast convergence in the initial course of training, where W_+ is a tunable nonnegative weight. Empirically, this additional loss quickly drops down to zero and does not affect the overall learned result, as the dominant eigenfunction is theoretically guaranteed to have nonnegative values everywhere in C.

Simultaneously, a backup policy that maximizes the eigenfunction value is trained. Jointly optimizing $\mathcal{J}_{\text{eig}}$ and a policy maximizing ψ_π values would lead to the safest policy possible for the given safety-critical system, leading to the result that π can serve as a safe backup policy, which is essential in the safety filtering phase. Any Q learning method could be used to find such π. In this paper's implementation, we minimize the following DDPG-style [50] loss functional to find a deterministic backup policy $u_t = \pi(x_t)$:

$$\mathcal{J}_{\text{policy}}[\pi] = -\frac{1}{|\mathcal{D}|} \sum_{(x,\cdot,\cdot)\in\mathcal{D}} \psi(x, \pi(x)). \tag{17}$$

This loss will try to learn a policy that increases the current ψ values, thereby increasing the eigenvalue γ_π. Suppose a new policy π' *improves* π in terms of the ψ_π values, i.e., $\psi_\pi(x, \pi'(x)) \geq \psi_\pi(x, \pi(x))$ for all $x \in C$. Then,

$$A_{\pi'}\psi_\pi(x, u) = \mathbb{E}_{x^+\sim P(\cdot|x,u)}[\psi(x^+, \pi'(x^+))] \geq \mathbb{E}_{x^+\sim P(\cdot|x,u)}[\psi(x^+, \pi(x^+))] = A_\pi\psi_\pi(x, u) \tag{18}$$

for all $x \in S$, so $\|A_{\pi'}\psi\|_\infty \geq \|A_\pi\psi\|_\infty$ meaning that the spectral radius of $A_{\pi'}$, which is equal to $\gamma_{\pi'}$, is not smaller than γ_π. If the policy gradient always improves the policy, then the eigenvalue should converge because it is a non-decreasing sequence upper bounded by the maximum possible value 1. In practice, the limit is the safest achievable one through the policy gradient.

3.2 Online Phase: Safety Filtering

In the online phase, in order to maintain safety of the robot system while respecting the user-given reference action, we employ a safety filter based on the learned backup policy and the eigenpair. The proposed safety filter has the following form:

$$a_t \sim \begin{cases} \pi_{\text{ref}}(\cdot|t, s_t) & \text{if } \phi(s_t) > \epsilon \\ \pi(\cdot|s_t) & \text{if } \phi(s_t) \leq \epsilon, \end{cases} \tag{19}$$

where π_{ref} is the reference action, e.g., manual control or a higher-level decision-making module, and $\epsilon > 0$ is a threshold the user can specify to well balance between safety and performance. One can recover $\phi(s_t)$, the eigenfunction of T_π, by directly querying $\phi(s_t) = \psi(s_t, \pi(s_t))$ if π is a deterministic policy, or in case of stochastic policy it can be learned via optimizing the loss $\mathcal{J}_\phi[\phi] = \frac{1}{|\mathcal{D}|}\sum_{(x,\cdot,\cdot)\in\mathcal{D}}(\phi(x) - \psi(x, u))^2$ with the learned ψ and π, where u here is sampled from the stochastic policy $\pi(\cdot|x)$.

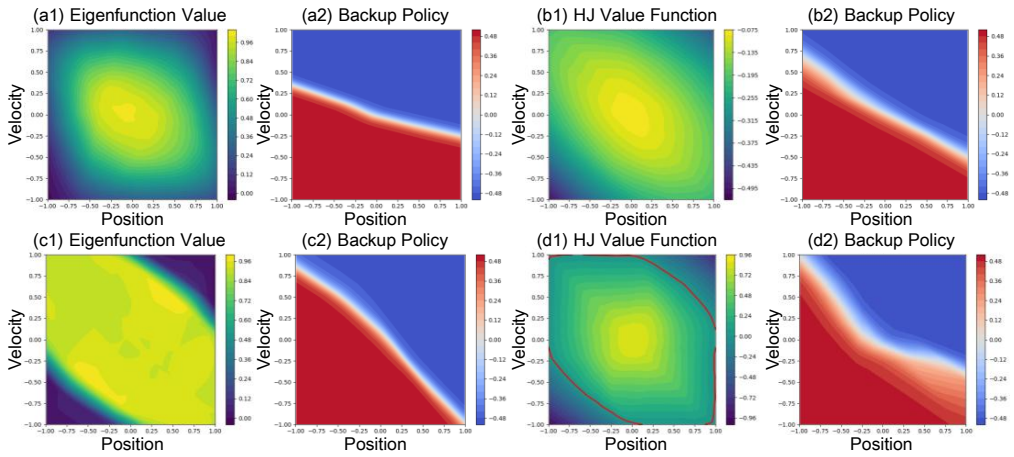


Figure 2: **Experiment results (the learned safety value function and the backup policy) for the double integrator system.** (a) EigenSafe with noise, (b) [32] with noise, (c) EigenSafe without noise, (d) [32] without noise. EigenSafe captures the relative safety between different states in both cases, but as seen in (b1), the reachability-based method [32] fails to converge with stochasticity and yields an empty invariant set. The super-zero level set of (d1), which is the infinite-horizon backward reachable set of C is marked with a red closed curve. The learned eigenvalues are 0.9652 for the stochastic case (a), and 1.0042 for the deterministic case (c).

4 Simulation Experiments

Double Integrator. We first evaluate EigenSafe using a double-integrator system with position and speed limits in two settings: a *stochastic* system in which Gaussian noise is added to the action, and the *deterministic* case without noise. For each setting, we compare against an RL implementation of HJ-reachability-based synthesis [32]. The dataset is obtained by uniform sampling over $C \times A$, and the learned safety value function and the backup policy are shown in Fig. 2. Further experimental details are given in the appendix.

It can be found in Fig. 2 (a) that EigenSafe successfully captures the relative safety across states for the stochastic system. In contrast, the reachability–based method [32] fails to reliably converge to a stable safe set, as seen in Fig. 2 (b). EigenSafe also produces reasonable results for the deterministic system (Fig. 2 (c)). Since safety probabilities do not decay in the deterministic case, the dominant eigenvalue should theoretically be 1 and the eigenfunction should be the indicator function of an invariant set. Empirically, the learned eigenfunction approaches the indicator of the *maximal* control-invariant set within the bounds, consistent with the reachability-based baseline.

Gymnasium Environments. We also experiment in two gymnasium environments: Half Cheetah and Lunar Lander. Their details can be found in the appendix.

For the Half Cheetah environment, we consider two tasks with different constraints: (i) *Cheetah Balance*, where the angle between the torso and the ground must not exceed $\pi/2$ (to prevent flipping), and (ii) *Cheetah Run*, where the horizontal velocity must remain positive (to encourage forward motion). Although the environment itself is deterministic, frequent contacts make its aggregate behavior appear stochastic. The trajectory dataset used in the training was collected with random actions.

The Lunar Lander environment itself is stochastic due to randomized terrain and wind disturbances. We train on a mixed set of trajectories, comprising (i) trajectories generated by the environment's default heuristic controller with additional random perturbations and (ii) trajectories generated by a purely random policy. States are labeled as unsafe whenever the lander moves beyond half of the image frame from the center, or when it makes ground contact with a body part other than its legs.

For both environments, as shown in Fig. 3, the resulting safety filter significantly reduces observed safety violations compared with the unfiltered reference actions. In addition, the learned eigenfunction values showed to be well capturing the system's safety level.

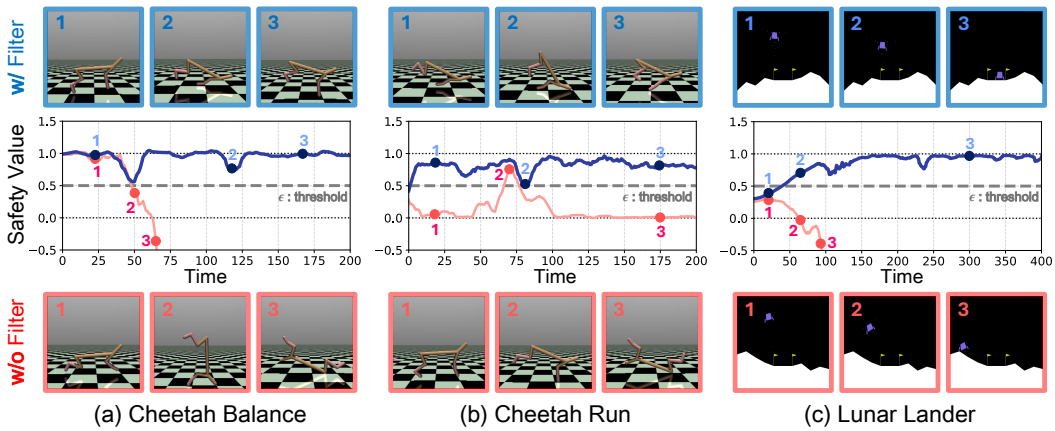


Figure 3: **Evolution of safety values and corresponding images.** All trajectories are generated using random actions as references. The learned safety values decrease when the agent takes potentially unsafe actions, thereby activating the safety filter (i.e., switching to the backup policy) whenever the safety value falls below the threshold. For clearer visualization, safety value curves are smoothed with a window size of 10 time steps.

5 Conclusion

This paper presents **EigenSafe**, a novel theoretical framework for learning safety value functions based on linear operator theory rather than optimal control. We derive a linear Bellman-like operator for dynamic programming for safety probability, whose dominant eigenpair provide a safety measure that is directly tied to the safety probability. In particular, the dominant eigenfunction serves as a safety Q function that quantifies relative safety of a given state-action pair. The proposed framework jointly learns the dominant eigenpair and a safe backup policy. Then in the deployment phase, unsafe inputs from the higher-level controller are filtered using the eigenfunction values and replaced with the learned safe backup policy.

We summarize limitations and future research directions as follows.

Overestimation Bias in Safety Quantification. One major limitation we observed in the experiments is a tendency for the learned eigenpair to overestimate the system's safety. In particular, the learned eigenvalue often reached above 1 which it can never exceed in theory. This overestimation phenomenon is well-known and common in RL [51], but it is particularly problematic in terms of safety. In the future work, we plan to apply techniques such as [51, 52] to mitigate this issue.

Out-of-Distribution (OOD) Failures. For safety-critical systems, since their safety cannot be reliably assessed, states that are outside of the dataset distribution should regarded as unsafe. However, the loss function (14) does not explicitly treat OOD state-action pairs as unsafe. Combined with the abovementioned overestimation bias, this could lead to undesirable drifts into OOD regions where the eigenfunction might assign higher values, particularly when the safe domain C is unbounded. Future work will address this by integrating OOD avoidance into the safety constraints, e.g., [38, 40, 42], or by introducing additional penalties near the boundary of the training dataset.

Future Research Directions. In addition to addressing these limitations, we aim to extend EigenSafe in two key directions. Firstly, we plan to integrate it with learned latent dynamics and safety constraints, as in [41], to handle high-dimensional observations such as images. Since latent dynamics are inherently noisy and provide only partial information about the true state, this setting naturally necessitates the use of a stochastic control framework. Future work will also deploy EigenSafe on physical robot hardware where real-world stochastic effects such as actuator noise and sensing errors exist, which originally motivated the development of this framework.

Acknowledgments

This work was supported by Samsung Research Funding & Incubation Center of Samsung Electronics under Project Number SRFC-IT2402-17. The work by Inkyu Jang was partially supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (RS-2024-00407121).

References

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

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

[3] M. Ganai, S. Gao, and S. L. Herbert. Hamilton-Jacobi reachability in reinforcement learning: A survey. *IEEE Open Journal of Control Systems*, 3:310–324, 2024. doi:10.1109/OJCSYS.2024.3449138.

[4] K. P. Wabersich, A. J. Taylor, J. J. Choi, K. Sreenath, C. J. Tomlin, A. D. Ames, and M. N. Zeilinger. Data-driven safety filters: Hamilton-jacobi reachability, control barrier functions, and predictive methods for uncertain systems. *IEEE Control Systems Magazine*, 43(5):137–177, 2023. doi:10.1109/MCS.2023.3291885.

[5] K. Margellos and J. Lygeros. Hamilton–jacobi formulation for reach–avoid differential games. *IEEE Transactions on automatic control*, 56(8):1849–1861, 2011.

[6] A. Abate, M. Prandini, J. Lygeros, and S. Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11):2724–2734, 2008. ISSN 0005-1098. doi:https://doi.org/10.1016/j.automatica.2008.03.027. URL `https://www.sciencedirect.com/science/article/pii/S0005109808002677`.

[7] N. Schmid and J. Lygeros. Probabilistic reachability and invariance computation of stochastic systems using linear programming. *IFAC-PapersOnLine*, 56(2):11229–11234, 2023.

[8] N. Schmid, M. Fochesato, T. Sutter, and J. Lygeros. Joint chance constrained optimal control via linear programming. *IEEE Control Systems Letters*, 8:736–741, 2024. doi:10.1109/LCSYS.2024.3403470.

[9] A. Hakobyan, G. C. Kim, and I. Yang. Risk-aware motion planning and control using cvar-constrained optimization. *IEEE Robotics and Automation Letters*, 4(4):3924–3931, 2019. doi:10.1109/LRA.2019.2929980.

[10] M. P. Chapman, R. Bonalli, K. M. Smith, I. Yang, M. Pavone, and C. J. Tomlin. Risk-sensitive safety analysis using conditional value-at-risk. *IEEE Transactions on Automatic Control*, 67 (12):6521–6536, 2022. doi:10.1109/TAC.2021.3131149.

[11] M. L. Bujorianu, R. Wisniewski, and E. Boulougouris. Stochastic safety for markov chains. *IEEE Control Systems Letters*, 5(2):427–432, 2021. doi:10.1109/LCSYS.2020.3002475.

[12] M. Prandini, J. Hu, C. Cassandras, and J. Lygeros. Stochastic reachability: Theory and numerical approximation. *Stochastic hybrid systems, Automation and Control Engineering Series*, 24:107–138, 2006.

[13] A. P. Vinod and M. M. Oishi. Stochastic reachability of a target tube: Theory and computation. *Automatica*, 125:109458, 2021.

[14] J. Steinhardt and R. Tedrake. Finite-time regional verification of stochastic non-linear systems. *The International Journal of Robotics Research*, 31(7):901–923, 2012.

[15] A. Clark. Control barrier functions for complete and incomplete information stochastic systems. In *2019 American Control Conference (ACC)*, pages 2928–2935. IEEE, 2019.

[16] P. Jagtap, S. Soudjani, and M. Zamani. Formal synthesis of stochastic systems via control barrier certificates. *IEEE Transactions on Automatic Control*, 66(7):3097–3110, 2021. doi:10.1109/TAC.2020.3013916.

[17] C. Santoyo, M. Dutreix, and S. Coogan. A barrier function approach to finite-time stochastic system verification and control. *Automatica*, 125:109439, 2021.

[18] S. Yaghoubi, K. Majd, G. Fainekos, T. Yamaguchi, D. Prokhorov, and B. Hoxha. Risk-bounded control using stochastic barrier functions. *IEEE Control Systems Letters*, 5(5):1831–1836, 2020.

[19] F. B. Mathiesen, S. C. Calvert, and L. Laurenti. Safety certification for stochastic systems via neural barrier functions. *IEEE Control Systems Letters*, 7:973–978, 2023. doi:10.1109/LCSYS.2022.3229865.

[20] R. Cosner, P. Culbertson, A. Taylor, and A. Ames. Robust Safety under Stochastic Uncertainty with Discrete-Time Control Barrier Functions. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi:10.15607/RSS.2023.XIX.084.

[21] R. K. Cosner, P. Culbertson, and A. D. Ames. Bounding stochastic safety: Leveraging freedman's inequality with discrete-time control barrier functions. *IEEE Control Systems Letters*, 8:1937–1942, 2024. doi:10.1109/LCSYS.2024.3409105.

[22] S. Bansal and C. J. Tomlin. Deepreach: A deep learning approach to high-dimensional reachability. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1817–1824. IEEE, 2021.

[23] H. Zhang, M. Tayal, J. Cox, P. Jagtap, S. Kolathaya, and A. Clark. Stochastic neural control barrier functions. *arXiv preprint arXiv:2506.21697*, 2025.

[24] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1):411–444, 2022.

[25] C. Dawson, S. Gao, and C. Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 39(3):1749–1767, 2023. doi:10.1109/TRO.2022.3232542.

[26] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, and A. Knoll. A review of safe reinforcement learning: Methods, theories, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):11216–11235, 2024. doi:10.1109/TPAMI.2024.3457538.

[27] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni. Learning control barrier functions from expert demonstrations. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3717–3724, 2020. doi:10.1109/CDC42340.2020.9303785.

[28] S. Liu, C. Liu, and J. Dolan. Safe control under input limits with neural control barrier functions. In *Proceedings of The 6th Conference on Robot Learning*, volume 205, pages 1970–1980. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/liu23e.html.

[29] 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 *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11532–11539. IEEE, 2024.

[30] J. Choi, F. Castañeda, C. J. Tomlin, and K. Sreenath. Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions. In *Robotics: Science and Systems (RSS)*, 2020.

[31] Z. Qin, D. Sun, and C. Fan. Sablas: Learning safe control for black-box dynamical systems. *IEEE Robotics and Automation Letters*, 7(2):1928–1935, 2022.

[32] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8550–8556, 2019. doi:10.1109/ICRA.2019.8794107.

[33] J. Li, D. Lee, J. Lee, K. S. Dong, S. Sojoudi, and C. Tomlin. Certifiable reachability learning using a new lipschitz continuous value function. *IEEE Robotics and Automation Letters*, 10 (4):3582–3589, 2025. doi:10.1109/LRA.2025.3535183.

[34] K.-C. Hsu, D. P. Nguyen, and J. F. Fisac. Isaacs: Iterative soft adversarial actor-critic for safety. In *The 5th Annual Learning for Dynamics and Control Conference*, volume 211, pages 90–103. PMLR, 15–16 Jun 2023. URL https://proceedings.mlr.press/v211/hsu23a.html.

[35] O. So and C. Fan. Solving Stabilize-Avoid via Epigraph Form Optimal Control using Deep Reinforcement Learning. In *Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi:10.15607/RSS.2023.XIX.085.

[36] D. Yu, H. Ma, S. Li, and J. Chen. Reachability constrained reinforcement learning. In *The 39th International Conference on Machine Learning*, volume 162, pages 25636–25655. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/yu22d.html.

[37] G. Chou, D. Berenson, and N. Ozay. Learning constraints from demonstrations. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 228–245. Springer, 2018.

[38] K. Kang, P. Gradu, J. J. Choi, M. Janner, C. Tomlin, and S. Levine. Lyapunov density models: Constraining distribution shift in learning-based control. In *The 39th International Conference on Machine Learning*, volume 162, pages 10708–10733. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/kang22a.html.

[39] K. Kim, G. Swamy, Z. Liu, D. Zhao, S. Choudhury, and S. Z. Wu. Learning shared safety constraints from multi-task demonstrations. *Advances in Neural Information Processing Systems*, 36:5808–5826, 2023.

[40] F. Castañeda, H. Nishimura, R. T. McAllister, K. Sreenath, and A. Gaidon. In-distribution barrier functions: Self-supervised policy filters that avoid out-of-distribution states. In *The 5th Annual Learning for Dynamics and Control Conference*, volume 211, pages 286–299. PMLR, 15–16 Jun 2023. URL https://proceedings.mlr.press/v211/castaneda23a.html.

[41] K. Nakamura, L. Peters, and A. Bajcsy. Generalizing safety beyond collision-avoidance via latent-space reachability analysis. In *Robotics: Science and Systems (RSS)*, 2025.

[42] J. Seo, K. Nakamura, and A. Bajcsy. Uncertainty-aware latent safety filters for avoiding out-of-distribution failures. *arXiv preprint arXiv:2505.00779*, 2025.

[43] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[44] V. Micheli, E. Alonso, and F. Fleuret. Transformers are sample-efficient world models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vhFu1Acb0xb.

[45] G. Zhou, H. Pan, Y. LeCun, and L. Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

[46] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse control tasks through world models. *Nature*, pages 1–7, 2025.

[47] M. Ganai, Z. Gong, C. Yu, S. Herbert, and S. Gao. Iterative reachability estimation for safe reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 69764–69797, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/dca63f2650fe9e88956c1b68440b8ee9-Paper-Conference.pdf.

[48] S. Huh and I. Yang. Safe reinforcement learning for probabilistic reachability and safety specifications: A lyapunov-based approach. *arXiv preprint arXiv:2002.10126*, 2020.

[49] S. E. Otto and C. W. Rowley. Koopman operators for estimation and control of dynamical systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):59–87, 2021.

[50] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[51] H. Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

[52] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative Q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

[53] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A Experiment Details

A.1 Double Integrator

The double integrator system used in the paper consists of a two-dimensional state $s_t = (p_t, v_t)$, where p_t and v_t represents the position and velocity at time t, respectively. This system is controlled using bounded acceleration actions $a_t \in A = [-0.5, 0.5]$. We use the following time-discretized model:

$$s_{t+1} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} s_t + \begin{bmatrix} dt^2/2 \\ dt \end{bmatrix} a_t + \begin{bmatrix} \sigma dt/2 \\ \sigma\sqrt{dt} \end{bmatrix} w_t, \tag{20}$$

where $dt = 0.05$, w_t are independent and identically distributed (i.i.d.) samples from the standard normal distribution $\mathcal{N}(0, 1^2)$. We consider the box constraint for safety: $s_t \in C = [-1, 1] \times [-1, 1]$. The result shown in Fig. 2 was carried out using two σ values: $\sigma = 1$ for Fig. 2 (a), (b), and $\sigma = 0$ for Fig. 2 (c), (d), representing stochastic and deterministic dynamic models, respectively.

For both methods, we used a uniformly sampled state-action-next state batch every epoch. For fair comparison with [32], we used the same hyperparameters for both. For [32], a discount rate $\gamma = 0.99$ was used. Since [32] is theoretically based on deterministic HJ reachability only, it did not converge with γ greater than 0.99. [32] also requires a continuous scalar function $l : S \to \mathbb{R}$ whose super zero level set coincides with C, which we set

$$l(x) = 1 - \|x\|_\infty . \tag{21}$$

The hyperparameter values other than abovementioned are summarized in Table 1.

Table 1: Hyperparameter values for training EigenSafe and [32] for the double integrator example. The superscripts $^{\text{E}}$ and $^{\text{H}}$ denote EigenSafe and [32], respectively.

Parameter	Value
Batch size $^{\text{EH}}$	2048
Learning rate (eigenfunction) $^{\text{EH}}$	3×10^{-4}
Learning rate (policy) $^{\text{EH}}$	2×10^{-3}
Optimizer $^{\text{EH}}$	Adam [53]
Nonlinearity $^{\text{EH}}$	ReLU
Weight W_λ $^{\text{E}}$	3.0
Weight W_n $^{\text{E}}$	1.0
Weight W_+ $^{\text{E}}$	1.0
# of hidden layers for safety value functions $^{\text{EH}}$	1
# of hidden units per layer for safety value functions $^{\text{EH}}$	128
# of hidden layers for the policy $^{\text{EH}}$	1
# of hidden units per layer for the policy $^{\text{EH}}$	16
# of gradient steps for training $^{\text{EH}}$	4000
Initial λ value $^{\text{E}}$	1.0
HJ value function discount rate γ $^{\text{H}}$	0.99

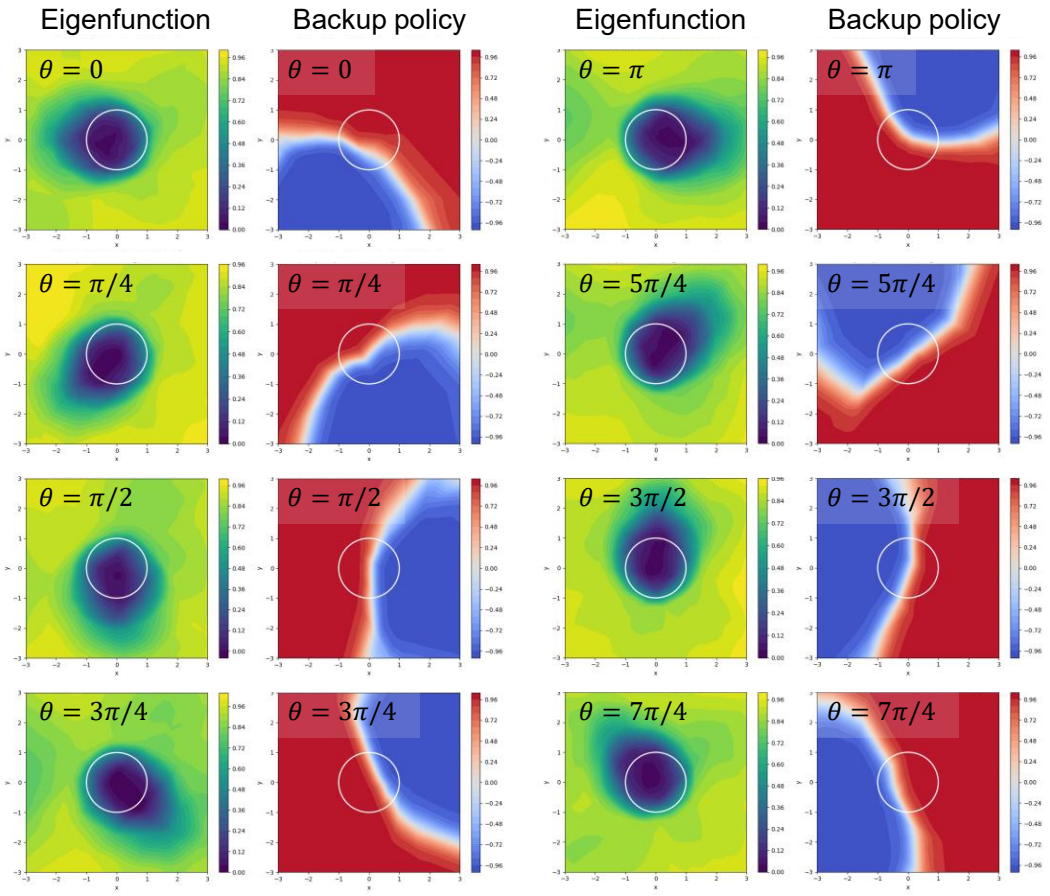


Figure 4: **Experimental results on the Dubins car.** Eight slices corresponding to different heading angles θ are shown. The heading angles are measured in the counterclockwise direction from the positive horizontal axes. In the backup policy plots, colors closed to red correspond to turning left, while blue colors mean right turns. It can be seen that the eigenfunction values well captures the expected potentially unsafe regions right before the circular obstacle, depending on the heading angle.

A.2 Dubins Car

We also test EigenSafe in the stochastic constant-speed Dubins car model with three-dimensional state space $s_t = (x_t, y_t, \theta_t)$ and one-dimensional action space $a_t \in A = [-1, 1]$. Its dynamics model is written as follows:

$$s_{t+1} = \begin{bmatrix} \text{wrap}(x_t + \cos(\theta_t)dt + z_1\sqrt{dt}, [-3, 3)) \\ \text{wrap}(y_t + \sin(\theta_t)dt + z_2\sqrt{dt}, [-3, 3)) \\ \text{wrap}(\theta_t + a_t dt + z_3\sqrt{dt}, [-\pi, \pi)) \end{bmatrix}, \tag{22}$$

where the wrap function projects any real value into the specified interval by adding or subtracting integer multiples of the interval length until it lies within the interval. For example, $\text{wrap}(3.1, [-3, 3)) = -2.9$ and $\text{wrap}(-4, [-\pi, \pi)) = 2\pi - 4$. This wrapping was introduced to obtain a more meaningful eigenpair: Since the speed of the car is fixed in the dynamics model, without this modification, the system would escape the domain at a very high probability in finite time, leading to $\gamma_\pi \approx 0$. The time interval is $dt = 0.05$ and the noises $z_{(\cdot)}$ are i.i.d. samples from the normal distribution $\mathcal{N}(0, 0.5^2)$. Similarly to the double integrator example, we uniformly sample state-action-next state data from $C \times A$, where the safe set C is defined as

$$C = \{(x, y, \theta) : x^2 + y^2 \geq 1\}. \tag{23}$$

The learned safety value function for the Dubins car example is shown in Fig. 4. The learned eigenvalue is 1.0019. It can be seen that the learned eigenfunction well captures the expected potentially unsafe states right next the obstacle. The hyperparameter values used in this example are summarized in Table 2.

Table 2: Hyperparameter values for training EigenSafe for the Dubins car example

Parameter	**Value**
Batch size	10000
Learning rate (eigenfunction)	1×10^{-3}
Learning rate (policy)	2×10^{-3}
Optimizer	Adam [53]
Nonlinearity	ReLU
Weight W_λ	3.0
Weight W_n	1.0
Weight W_+	1.0
# of hidden layers for safety value functions	1
# of hidden units per layer for safety value functions	128
# of hidden layers for the policy	1
# of hidden units per layer for the policy	16
# of gradient steps for training	5000
Initial λ value	1.0

A.3 Gymnasium environments

Half Cheetah. The default parameters for the dynamics were used.

Lunar Lander. We used the *continuous action space* version for the Lunar Lander example, which is generally considered as more challenging than the default discrete action space version which only considers bang-bang control. Other parameters are set to default, except for the safety constraints, which is delineated in Section 4.

More details can be found in the official Gymnasium documentation at `https://gymnasium.farama.org/`. The hyperparameters used in the experiments in Gymnasium environments are summarized in Table 3.

Table 3: Hyperparameters for training EigenSafe for the Gymnasium environments

Parameter	**Value**
Offline dataset size	1×10^6 (Half Cheetah), 5×10^6 (Lunar Lander)
Episode length	200 (Half Cheetah), 400 (Lunar Lander)
Minibatch size	512
Learning rate	1×10^{-4}
Optimizer	Adam [53]
Nonlinearity	ReLU
Weight W_λ	1.0
Weight W_n	1.0
Weight W_+	1.0
# of hidden layers	2
# of hidden units per layer	1024
# of gradient steps for training	5×10^5
Initial λ value	1.0

The learning curves for the two losses (eigenfunction and policy losses), along with the eigenvalues are shown in Fig. 5. It can be seen that all the losses and the eigenvalues converge well in practice. As we stated in Section 5, the eigenvalues often appear to be overestimating the safety of the system (having values greater than 1), which is a well-known and common issue in RL [51], happening because the policy optimization tends to pick actions whose eigenfunctions are overestimated by noise during training. This should be addressed in the future works. We also point out that the fluctuation of these values along the course of training is not significant compared to other RL algorithms, demonstrating computational robustness of EigenSafe, mainly because the training was done using a fixed offline dataset.

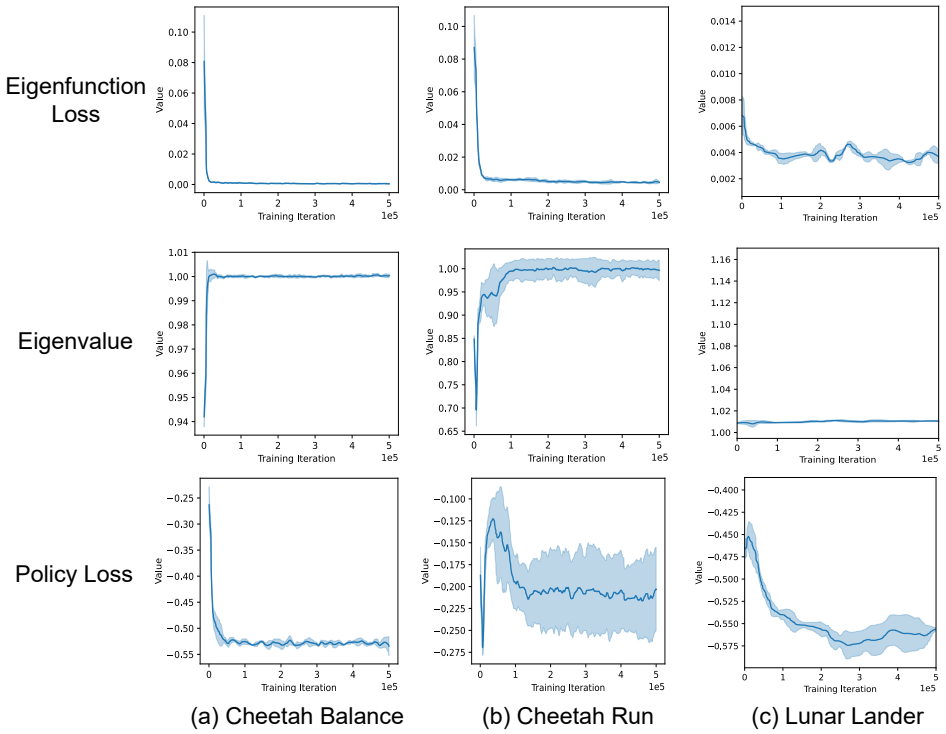


Figure 5: **Learning curves of EigenSafe in Gymnasium experiments (4 seeds).** The convergence of the eigenfunction and policy losses, and the eigenvalue during training demonstrates the stability of the joint optimization.