Temporal Difference Learning of Probabilistic Reach-Avoid Value Functions

Elias Fontanari[1], Pietro Noah Crestaz[1,2], Matteo Saveriano[1] and Andrea Del Prete[1]

***Abstract*— Reach-avoid sets, defining the states from which a task can be completed while avoiding unsafe regions, are fundamental tools for deploying robots in environments subject to disturbances and uncertainty, as well as for runtime task-feasibility monitoring, allowing prediction of whether a state is safe and will lead to successful task completion. Classical Hamilton-Jacobi Reachability (HJR) and its recent neural extensions (e.g., Lipschitz Continuous Reachability Learning, DeepReach) treat disturbances adversarially, computing the reach-avoid set under worst-case disturbances from a bounded set. This guarantees safety but can yield overly conservative sets when disturbances are stochastic rather than adversarial, and additionally requires the disturbance set to be known or specified in advance.**

We propose Probabilistic reach-avoid Value Function (PVF), a self-supervised method, applicable to black-box systems, for computing reach-avoid sets as the states with a certain reach-avoid probability under stochastic, potentially unbounded disturbances. We evaluate our approach against HJR-based baselines on systems of increasing complexity, up to a Hopper locomotion task, and show that our probabilistic sets are less conservative and more accurate than adversarial HJR sets when disturbances are non-adversarial.

I. Introduction

Autonomous robotic systems are increasingly deployed to perform complex tasks in environments subjected to different sources of noise and uncertainty. Determining, from a given state, whether a task of interest can be completed while satisfying critical safety constraints can often be framed in terms of the *Reach-Avoid set*: the set of states from which the system can reach a target region while avoiding an unsafe one. Avoid and Reach-Avoid sets are essential in runtime safety monitoring, as done in model predictive shielding [1]. Reach-avoid sets are also a useful tool for computing Control Invariant Sets (CIS) [2]: if the target set used to compute a reach-avoid set is itself control invariant, the resulting reach-avoid set is also control invariant, and typically larger. CISs are essential tools in safe control methods such as Model Predictive Control (MPC) [3], since constraining the terminal state of the associated optimal control problem to lie in a CIS ensures recursive feasibility and constraint satisfaction. Therefore, the accurate computation of reach-avoid sets is extremely useful for deploying robots in safety-critical settings.

A. Related works

Existing approaches to computing reach-avoid sets are predominantly grounded in Hamilton-Jacobi Reachability (HJR) [4], which treats disturbances as an adversary selecting the worst-case action from a bounded set. This comes with two main drawbacks: i) these methods cannot be applied to disturbances with unbounded support, such as a Gaussian distribution, and ii) when the disturbance is random they tend to be extremely conservative. Classic HJR methods rely on grid-based dynamic programming over the state space, which scales exponentially with the system's dimensionality, rendering them intractable for high-dimensional systems.

For this reason, the computation of reach-avoid sets has been addressed mainly by two lines of work, alleviating the curse of dimensionality: physics-informed neural networks such as DeepReach [5], and reinforcement learning [6], [7]. DeepReach learns the reach-avoid set through a curriculum learning procedure that enforces the conditions of the associated Hamilton-Jacobi-Isaacs partial differential equation (PDE). RL-based methods, instead, learn a critic that encodes the safe set, replacing the PDE-based Hamilton-Jacobi formulation with a Bellman recursion that can be solved via RL algorithms. However, in the adversarial setting the disturbance policy must be learned jointly with the control policy, turning training into a min-max optimization problem that is notoriously unstable and slow to converge. In addition, the theoretical guarantees typically rely on regularity assumptions on the dynamics, most commonly Lipschitz continuity, which may not hold for hybrid or contact-rich systems. DeepReach has been tested on large nonlinear systems such as quadrotors, but, unlike RL methods, it cannot be applied to non-differentiable dynamical systems, such as legged robots or black-box systems, since the dynamics is needed in the computation of the Hamiltonian.

A separate line of work, stochastic reachability analysis [8], [9], addresses reach-avoid problems directly as probabilities of success under a stochastic disturbance model, formulated via dynamic programming over the system's transition kernel in a measure-theoretic setting. While closer in spirit to our probabilistic formulation, these approaches rely on the numerical solution of the associated Bellman recursion over a gridded or otherwise explicitly represented state space, which precludes their use with the high-dimensional, black-box dynamics typical of robotic simulators.

In another line of work [10] the authors used actor-critic RL to learn a safe policy, where the critic expresses a pseudo-probability of future failure. However, several aspects of this formulation limit its use as a set-computation method. First, the critic addresses only the avoid component: it estimates the probability of failure with no notion of task completion and cannot distinguish states from which the task is

[1]Industrial Engineering Department, University of Trento, Trento, Italy.
[2]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France.

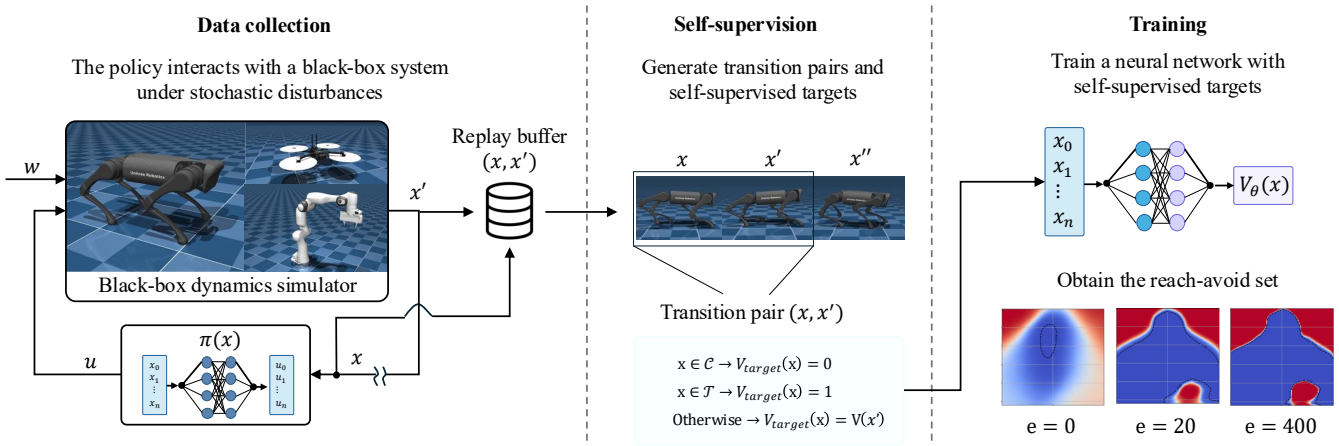


Fig. 1. Overview of the PVF pipeline. *Data collection:* Trajectories are collected by running a nominal policy $\pi(x)$ on black-box systems (e.g., legged robots, manipulators, quadrotors) subject to stochastic disturbances w, and stored as transition pairs (x, x') in a replay buffer. *Self-supervision:* Piecewise target labels $V_{\text{target}}(x)$ are assigned based on constraint violation (0), task completion (1), or bootstrapped value evaluation ($V_\theta(x')$). *Training:* A neural network parameterizing $V_\theta(x)$ is optimized via temporal-difference loss to output the continuous reach-avoid success probability.

achievable from states that are merely never-failing. Second, the critic is defined with respect to the policy induced by the critic itself, a self-referential construction that the authors report to be excessively pessimistic early in training and that requires implementation-level corrections, namely computing Bellman targets under the unconstrained policy and maintaining a separate near-on-policy replay buffer.

Another method capable of computing reach-avoid sets, specifically used to compute CISs, is Viability Boundary Optimal Control (VBOC) [11], which avoids resorting to HJR by directly identifying states on the boundary of the reach-avoid set and training a neural network to represent it. The target set corresponds to the system's equilibria, while the avoid set represents collisions with the environment or joint limit violations. By directly learning the boundary of the reach-avoid set, VBOC achieves higher data efficiency compared to HJR, but its main drawbacks are that it does not account for disturbances and can only handle fully-actuated systems.

B. Contribution

We introduce **PVF** (**P**robabilistic Reach-Avoid **V**alue **F**unctions), addressing the typical drawbacks of HJR methods in settings where there is no adversary, but simply non-adversarial stochastic noise, as is often the case in robotics scenarios, wind disturbances acting on an aerial vehicle being a representative example. By learning a value function that expresses, for each state, the probability of reaching the target set without ever entering the failure set, we are able to assess safe target reaching in a graded fashion: states that succeed under most disturbance realizations, and fail only under a highly unfavourable sequence of disturbances, are distinguished from states that fail under typical conditions.

Crucially, the entire procedure treats the system as a black box: it requires only the ability to sample transitions from the environment, with no access to an analytical model, no differentiability of the dynamics, and no knowledge of the disturbance distribution. This makes the approach directly applicable to contact-rich and hybrid systems, such as legged robots, for which a closed-form model is either unavailable or non-differentiable, and where PDE-based methods cannot be applied. Fig. 1 provides an overview of the proposed method.

Additionally, our value function provides interpretable probabilities, as opposed to those computed by HJR methods, which provide either the worst-case constraint violation the trajectory will incur, or the value of the target-set membership function at the time the target is reached.

Summarizing, the contributions are the following: (i) a novel self-supervised method for computing probabilistic reach-avoid sets under stochastic disturbances with potentially unbounded support, without assuming worst-case behavior; (ii) a comparison against HJR-based baselines on systems of increasing complexity, up to a Hopper locomotion task, showing that the resulting sets are more interpretable, less conservative and more accurate under non-adversarial disturbances.

II. Learning the Probabilistic Value Function (PVF)

A. Problem formulation

We consider stochastic discrete-time dynamical systems of this form:

$$\begin{aligned} &x' = f(x, u, w), \\ &x \in \mathcal{X} \subseteq \mathbb{R}^n,\ u \in \mathcal{U} \subseteq \mathbb{R}^m,\ w \sim p_{\mathcal{W}}, \end{aligned} \tag{1}$$

where x is the current state, x' the state reached applying the control u, and the stochastic disturbance w, drawn from a possibly unbounded distribution $p_{\mathcal{W}}$, to the system. The transition function $f : \mathcal{X} \times \mathcal{U} \times \mathcal{W} \rightarrow \mathcal{X}$ is treated as a *black box*: it is unknown in closed form and can only be queried through a simulator or real-world rollouts, without access to its analytical form, gradients, or the explicit distribution $p_{\mathcal{W}}$.

Given a fixed (possibly stochastic) policy $\pi(\cdot) : \mathcal{X} \rightarrow \mathcal{U}$, the objective is to compute, starting from each state x, the

probability of reaching a target set $\mathcal{T}$ without passing through a constraint set $\mathcal{C}$, i.e. the reach-avoid probability:

$$V(x) \triangleq \mathbb{P}(\tau_{\mathcal{T}} < \tau_{\mathcal{C}} \mid X_0 = x)\,, \tag{2}$$

where $\tau_{\mathcal{T}}$ and $\tau_{\mathcal{C}}$ denote the first hitting times of the target and constraint sets, respectively. Given this probability function, we can define probabilistic reach-avoid sets, for a given probability threshold, as the superlevel set of the function.

Our approach does not require an explicit expression for the target and constraint sets; it suffices to have a method that checks whether a given state belongs to them. To this end, we define two indicator functions, $\mathcal{I}_{\mathcal{T}}$ and $\mathcal{I}_{\mathcal{C}}$, associated with the target set $\mathcal{T}$ and the constraint set $\mathcal{C}$, respectively:

$$\mathcal{I}_{\mathcal{T}}(x) = \begin{cases} 1 & x \in \mathcal{T},\ \text{target reached} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$\mathcal{I}_{\mathcal{C}}(x) = \begin{cases} 1 & x \in \mathcal{C},\ \text{constraints violated} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The indicator functions can be negated with an overline bar, such as $\overline{\mathcal{I}}$.

B. Method

A state can be either in the target set, in the constraint set, or in neither of those. Assuming without loss of generality that target and constraint set have empty intersection[1], these three cases are mutually exclusive. Let us now define the following three symbols to represent these three events, where x_i is the state at the step i in an episode starting from step 0:

$$\begin{aligned} T_i &: \quad x_i \in \mathcal{T} \\ C_i &: \quad x_i \in \mathcal{C} \\ N_i &: \quad x_i \notin \mathcal{T} \wedge x_i \notin \mathcal{C} \end{aligned} \tag{5}$$

Starting from a given state at timestep $t_0 = 0$ we define the event S_0 of eventually reaching the target set before violating any constraint:

$$\begin{aligned} S_0 &= \bigvee_{t \geq 0} \left[T_t \wedge \left(\bigwedge_{i \in [0, t-1]} N_i \right) \right] \\ &= T_0 \vee (N_0 \wedge T_1) \vee (N_0 \wedge N_1 \wedge T_2) \vee \dots \end{aligned} \tag{6}$$

Since the event N_0 is common to all the terms after the first one, we can collect it to express this event recursively:

$$\begin{aligned} S_0 &= T_0 \vee [N_0 \wedge \overbrace{(T_1 \vee (N_1 \wedge T_2) \vee \dots)}^{S_1}] \\ &= T_0 \vee (N_0 \wedge S_1) \end{aligned} \tag{7}$$

Exploiting the fact that T_0 and N_0 are mutually exclusive, we can translate this expression in terms of probabilities as:

$$\begin{aligned} \mathrm{P}(S_0) &= \mathrm{P}(T_0) + \mathrm{P}(N_0 \wedge S_1) \\ &= \mathrm{P}(x_0 \in \mathcal{T}) + \mathrm{P}(x_0 \notin \mathcal{C} \wedge x_0 \notin \mathcal{T} \wedge S_1) \end{aligned} \tag{8}$$

[1]If that is not the case we can always redefine the target set by removing its intersection with the constraint set.

After the state x_0 is observed, the above probabilities on x_0 become binary and can be expressed using the indicator functions defined above:

$$\mathrm{P}(S_0 | X_0 = x_0) = \mathcal{I}_{\mathcal{T}}(x_0) + \overline{\mathcal{I}_{\mathcal{C}}}(x_0)\,\overline{\mathcal{I}_{\mathcal{T}}}(x_0)\,\mathrm{P}(S_1 | X_0 = x_0) \tag{9}$$

This expression is our starting point for defining a recursive evaluation of the reach-avoid probability, which we want to learn as a Value function $V(x)$. Based on (9) we write the following expression for the Reach-Avoid value function:

$$V(x_0) = \mathcal{I}_{\mathcal{T}}(x_0) + \overline{\mathcal{I}_{\mathcal{C}}}(x_0)\,\overline{\mathcal{I}_{\mathcal{T}}}(x_0)\,\mathbf{E}[V(x_1) | X_0 = x_0] \tag{10}$$

Evaluating the expectation of $V(\cdot)$ directly is intractable because the transition probability distribution $\mathrm{P}(x' | x, u)$ is unknown. However, given access to a simulator, it is possible to sample from the distribution and approximate this expectation without explicit knowledge of the transition dynamics using Temporal-Difference (TD) learning [12]:

$$V(x) \leftarrow V(x) + \alpha\big(V_{\text{target}}(x, x') - V(x)\big), \tag{11}$$

where $\alpha \in [0, 1]$ is the user-defined learning rate. The target can be computed by evaluating (10) in the three cases where x belongs to $\mathcal{C}$, to $\mathcal{T}$, or to neither of those:

$$V_{\text{target}}(x, x') := \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ 1 & \text{if } x \in \mathcal{T} \\ V(x') & \text{otherwise} \end{cases} \tag{12}$$

As typically done in TD learning, the expectation of $V(\cdot)$ over the stochastic system dynamics in (10) is replaced by the current estimate of $V(\cdot)$ for the observed next state x'.

Assumption 1. *Let us define the time at which the state enters either the target or the constraint set for the first time:* $\tau(x) = \inf\{t \geq 0 : x_t \in \mathcal{T} \cup \mathcal{C}, x_0 = x\}$. *We assume that eventually the system reaches* $\mathcal{T} \cup \mathcal{C}$ *almost surely, that is:* $P(\tau(x) < \infty) = 1$ *for every* $x \in \mathcal{X}$.

This assumption rules out closed regions in which the system can remain indefinitely without reaching either $\mathcal{T}$ or $\mathcal{C}$. In the presence of such regions, the undiscounted Bellman equation may admit multiple fixed points because no terminal boundary value is propagated to them.

By using a neural network as a function approximator, the TD update (11) can be extended to the learning of the continuous reach-avoid probability. The loss function to be minimized is

$$\min_{\theta} \sum_{(x, x') \in \mathcal{P}} \left\| V_\theta(x) - V_{\text{target}}(x, x') \right\|^2 \tag{13}$$

where $\mathcal{P}$ is the set of the stored pair transitions, obtained by rolling out the system using the given policy π.

C. Algorithmic implementation

To train a value function, the procedure is reported in Algorithm 1. First, trajectories are generated in a data collection phase by sampling random initial states and executing the given policy. Subsequently, transition pairs are extracted from the trajectories and inserted into the paired dataset $\mathcal{P}$.

Algorithm 1 Probabilistic Value Function (PVF) Training

Require: Simulator $\mathcal{S}$, control policy π, number of episodes n_{episodes}, network parameters θ, target network parameters θ_{target}, soft update coefficient τ_θ, stopping threshold ϵ, maximum number of epochs $E_{\max}$

Ensure: Trained parameters θ

$\mathcal{D} \leftarrow \emptyset$ ▷ trajectory buffer
for $i = 1$ to n_{episodes} **do**
  Sample initial state $x_0 \sim \mathcal{X}$
  $\tau \leftarrow \text{Rollout}(\mathcal{S}, \pi, x_0)$, stopping early if $\mathcal{C}$ or $\mathcal{T}$ are reached
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau\}$
$\mathcal{P} \leftarrow \text{ExtractTransitionPairs}(\mathcal{D})$
for $e = 1$ to $E_{\max}$ **do**
  $\theta_{\text{old}} \leftarrow \theta$
  Randomly split $\mathcal{P}$ in minibatches B_i, creating batch set $\mathcal{B} = \{B_1, ..., B_n\}$
  for each batch $B_i \in \mathcal{B}$ **do**
    for each pair $(x, x') \in B_i$ **do**
      if $x \in \mathcal{C}$ **then**
        $y \leftarrow 0$
      else if $x \in \mathcal{T}$ **then**
        $y \leftarrow 1$
      else
        $y \leftarrow V_{\theta_{\text{target}}}(x')$
      Label pair (x, x') with target y
    Update θ: $\min_\theta \sum_{(x,x') \in B_i} \|V_\theta(x) - y\|^2$
    Update target network: $\theta_{\text{target}} \leftarrow \tau_\theta\, \theta + (1 - \tau_\theta)\, \theta_{\text{target}}$
  rel_norm $\leftarrow \|\theta - \theta_{\text{old}}\| \,/\, \|\theta_{\text{old}}\|$
  if rel_norm $< \epsilon$ **then**
    break
return θ

Finally, the training phase is executed. Each pair is used to compute the target for $V_\theta(x)$ as described in Eq. 12, and the network learns the reach-avoid probability by minimizing the mean squared error with respect to the target. The training is stopped when the change of the network parameters θ is small and lower than a chosen threshold ϵ. We noticed that introducing another network to compute the target when needed, improves the stability of the learning process. After each optimization step, the target network parameters θ_{target} are updated using Polyak averaging [13], with small soft update coefficient τ_θ, in the order of 0.001.

While our theory discusses an infinite-horizon hitting probability, the algorithm uses a finite episode length. However, this timeout is only a data-collection cutoff for an infinite-horizon quantity because the value targets are computed based on the current estimate of the reach-avoid probability. Therefore, as the number of training epochs grows to infinity, so does the horizon considered by the computed value function.

D. Comparison with related methods

We compare our approach with two representative learning-based safety methods: Lipschitz Continuous Reachability Learning (LCRL) [7], which learns robust reach-avoid sets, and Safety Q-functions for Reinforcement Learning (SQRL) [10], which estimate the future failure risk of a safety-constrained policy. Both methods additionally synthesize or adapt a controller. In contrast, our method focuses on estimating the reach-avoid probability of a given closed-loop system. We therefore restrict the following discussion to the value functions learned by the three approaches.

LCRL belongs to the family of Hamilton–Jacobi reachability methods and adopts a robust formulation: its value function determines whether the target can be reached safely for every disturbance realization within a prespecified bounded set. Our method instead models disturbances stochastically and estimates the probability of reaching the target before entering the constraint set. Consequently, the two value functions encode different notions of feasibility. LCRL provides a worst-case, binary characterization through the sign of its value function, whereas our value function quantifies the likelihood of task success under the disturbance distribution. The latter can distinguish states having different success probabilities even when neither state admits a worst-case guarantee. Moreover, our formulation does not require a bounded disturbance set and therefore also accommodates distributions with unbounded support. To enable a direct empirical comparison, we consider bounded stochastic disturbances, for which LCRL can be applied by treating every disturbance in the support as adversarial. Such a comparison highlights the conservatism introduced when disturbance likelihoods are discarded. SQRL is closer to our approach because its safety critic is also defined through an expectation over stochastic transitions. Its objective, however, differs from reach-avoid feasibility. SQRL estimates, for each state–action pair, the discounted risk of eventually entering a failure set, while task completion is handled separately through the policy reward. It therefore does not include an explicit target set in the safety-critic formulation.

This distinction is particularly relevant over long or infinite horizons. If a system remains exposed to a nonzero failure risk indefinitely, its eventual failure probability may equal one over a large portion of the state space, providing little information about task feasibility. SQRL mitigates this issue through a finite horizon and a safety discount factor $\gamma_{\text{safe}} < 1$, which emphasize near-term failures but cause the resulting critic to represent discounted risk rather than an undiscounted hitting probability. Our target set instead defines successful termination explicitly. Together with the constraint set, it supplies the two boundary conditions needed to distinguish successful and unsuccessful task completion. Under the standard Assumption 1, these boundary conditions uniquely determine the reach-avoid probability and provide terminal supervision for TD learning. Thus, the two approaches are complementary: SQRL is designed to constrain failure risk during policy learning, whereas our method is

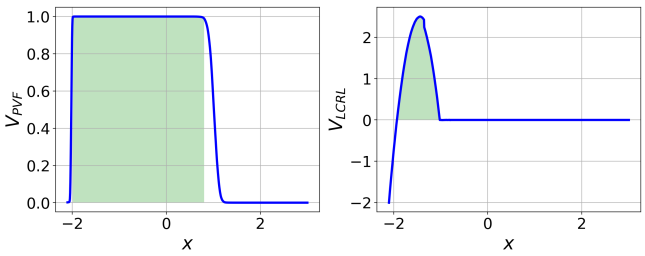


Fig. 2. Reach-avoid value functions on the scalar toy problem. Comparison between the proposed Probabilistic Value Function (V_{PVF}, left) and the adversarial Lipschitz Continuous Reachability Learning baseline (V_{LCRL}, right). Green regions indicate states identified as safe (i.e., able to reach the target set without violating constraints).

designed to quantify the probability that a specified task can be completed safely.

III. Results

To assess the accuracy and characteristics of our method[2], we compared it against the two above-mentioned state-of-the-art methods: LCRL [7], and SQRL [10] without discounting.

A. Introductory example: Toy problem

To illustrate the difference between the proposed method and HJR methods, we first consider a simple scalar toy problem, taken from [7]. Let $x \in \mathcal{X} \subseteq \mathbb{R}$ denote the scalar state, $u \in \mathcal{U} \subset \mathbb{R}$ the control input, and $w \in \mathcal{W} \subset \mathbb{R}$ the stochastic disturbance. The dynamics, target set $\mathcal{T}$, and constraint set $\mathcal{C}$ are defined as:

$$\begin{aligned} x' &= 1.01\,x + 0.01\,(u + w), \\ \mathcal{T} &= \{x \in \mathbb{R} \mid -2 < x \leq -1\}, \\ \mathcal{C} &= \{x \in \mathbb{R} \mid x \leq -2\}, \\ \text{s.t.} \quad & |u| \leq 1, \quad |w| \leq 4, \end{aligned} \tag{14}$$

where the disturbance bound $|w| \leq 4$ corresponds to a zero-mean Gaussian distribution with $\sigma = 1$ truncated at 4σ. To generate the dataset $\mathcal{D}$ for evaluating our method, we collect transitions under the control policy $\pi(x)$ computed by LCRL [7]. The resulting value functions obtained via our method (V_{PVF}) and LCRL (V_{LCRL}) are reported in Fig. 2.

With this level of noise, if the worst-case disturbance is always chosen, the system can never reach the target (except when it is already inside the target), as shown in the right panel of Fig. 2. In contrast, our method marks a larger portion of the state space as successful (i.e., states from which the target can be reached with high probability). This example illustrates that our method is substantially different from reach-avoid HJR, and that in situations where the noise is not adversarial, it can be significantly less conservative.

B. Value function accuracy: LCRL comparison

We assessed the accuracy of our method on two other systems of increasing difficulty: Drones Racing [7] and Hopper [14], [15]. Since PVF requires a policy, for a fair comparison we always tested all methods with the same policy. For the Drone Racing experiment, we used the policy learned directly via LCRL [7]. For Hopper, where LCRL struggled to optimize a control policy, we instead provided LCRL with a pre-trained locomotion policy obtained using the standard RL algorithm Proximal Policy Optimization (PPO) [16]. This isolates LCRL's role to learning the worst-case disturbance and the corresponding reach-avoid set under a fixed policy. The PVF network was a 5-layer MLP with 256 hidden units per layer and GELU activations, with a linear (unactivated) scalar output. LCRL's actor (present in Drones Racing only), disturbance, and critic networks used a 4-layer, 512-unit MLP with ReLU activations, taking the state (plus action, for the critic) as input, with role-dependent outputs: a scalar estimate for the critic, and a bounded control for the actor/disturbance networks.

In all tests, LCRL was run until the critic loss stabilized below a chosen threshold (10^{-4} for Drones Racing and 10^{-6} for Hopper), or until the maximum number of epochs (160) was reached.

To obtain a statistically grounded estimate of the true reach-avoid probability of the policy π at a given state x, we treat each closed-loop rollout as an independent Bernoulli trial and estimate the reach-avoid probability $p(x)$ via Monte Carlo sampling with an adaptive stopping rule based on the Wilson score confidence interval. This is known to give substantially better coverage than the naive normal-approximation (Wald) interval, especially for small sample sizes or proportions near 0 or 1 [17].

For each sampled state x_i, we run a fixed budget of $n = 4000$ rollouts and form the estimate $\hat{p}_i = s_i/n$ from the observed successes s_i, together with its Wilson interval $[p_{lo,i}, p_{hi,i}]$ at confidence level $1 - \beta = 99.5\%$. The state is classified *truly successful* if $p_{lo,i} \geq \ell$, *truly unsuccessful* if $p_{hi,i} < \ell$, and *unclear* if the interval straddles the target safety level $\ell = 0.99$. This provides the Monte Carlo reference against which the learned reach-avoid classifications are evaluated. The ground-truth probability level $\ell = 0.99$ was chosen as a practical trade-off: it is high enough to clearly discriminate unsuccesful states, while remaining statistically tractable, since higher levels require substantially more rollouts to certify at the confidence chosen. Certifying a success probability of exactly 1, by contrast, is fundamentally problematic, as no finite number of trials can rule out the possibility that an unfavorable event has simply not yet been observed. The value functions approximations $V_{PVF}(x)$ and V_{LCRL} are scored against this ground truth certificates over the clear states: a *true positive* (TP) is a truly unsuccessful state that V also predicts unsuccessful, a *false negative* (FN) a truly unsuccessful state that V predicts successful, a *false positive* (FP) a truly successful state that V predicts unsuccessful, and a *true negative* (TN) a truly successful state that V predicts successful. For each system, these metrics were evaluated on 1000 states reached after a random number of policy steps from randomly sampled initial states, discarding those that ended inside the target or the constraint set. With $n = 4000$ rollouts per state, the fraction of unclear

[2]Code will be made available upon acceptance

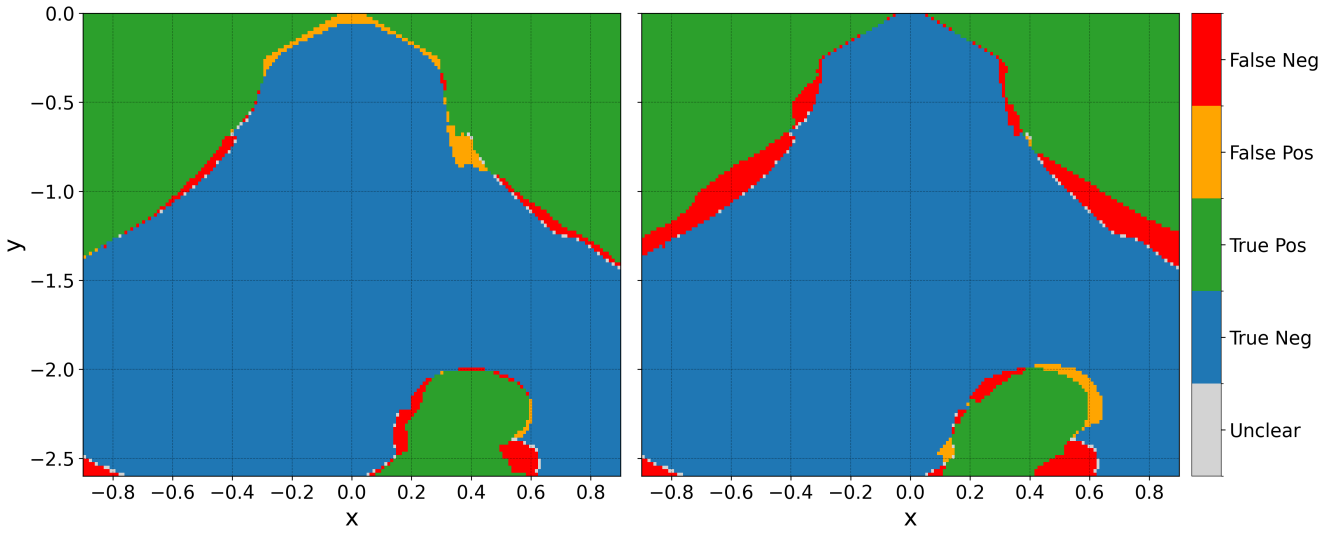


Fig. 3. Visual comparison between PVF (left) and LCRL (right) in terms of accuracy, for Drones racing experiment. x and y indicate the initial position of the ego drone in the horizontal plane. The opposing drone starts at $x = 0.4$ m and $y = 0.0$ m

cases remained below 4.5% across all experiments.

1) Drones racing: This scenario, adapted from the LCRL paper [7], formulates a drone take-over maneuver as a reach-avoid problem involving two quadrotors racing through a gate, where the ego drone must overtake safely its opponent. The opposing drone heads straight for the gate under a fixed LQR controller, but its acceleration is not perfectly known to the ego drone; this uncertainty enters the model as a disturbance bounded in norm by 2 m/s^2. The noise distribution is assumed to be a zero-mean Gaussian truncated at 3σ, with $\sigma = \frac{2}{3}$m/s^2. Our goal is to compute the probability that the ego drone can safely overtake the opponent drone and reach the gate first, regardless of this disturbance.

The joint system state is 12-dimensional, with each drone described by its position and velocity along the three spatial axes. Along each axis, the drones follow double-integrator dynamics, driven by acceleration inputs. While the dynamics themselves are linear, the problem is nonlinear due to the constraints defining the avoid set. In particular, the safety constraint requires the ego drone to stay outside the airflow generated by the other drone, combining a squared norm with a max operator.

The results of three full comparisons under different seeds are presented in Fig. 4, where the FP and FN of each method are reported, showing minimum, maximum and median errors across runs. We also report the classification threshold used by PVF and LCRL. At the nominal thresholds (0.99 for PVF and 0 for LCRL), PVF performed remarkably better in terms of FN and moderately better in terms of FP. The number of FN, the most dangerous classification failure due to a missed alarm, was consistently smaller for PVF, except when the LCRL threshold was increased over 0.1, which however increased the FP (i.e., false alarms) above 6%. Fig. 3 shows a visual comparison between the reach-avoid sets computed by PVF and LCRL, tested against

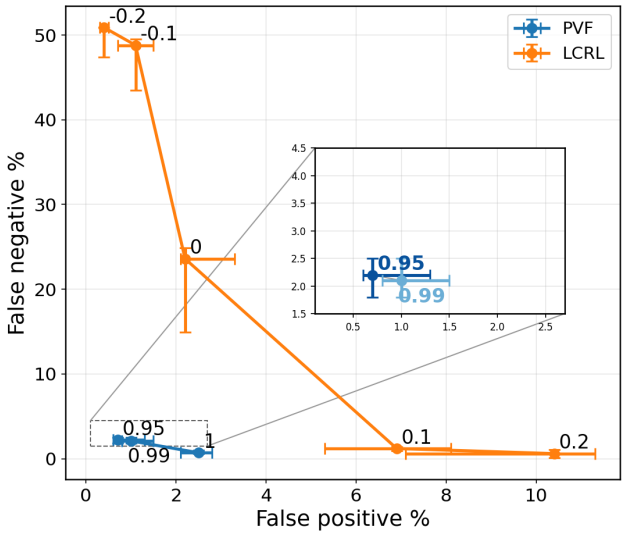


Fig. 4. Comparison between PVF and LCRL in terms of accuracy, for Drones racing experiment. Thresholds for fail detection reported near the medians.

Wilson confidence intervals, on a particular section of the state space, as in [7], by varying only the xy position of the ego drone in the horizontal plane. This figure highlights the higher number of FN for LCRL, particularly for $y > -1.5$ m.

2) Hopper: This test involves the common RL benchmarking system Hopper [15], shown in Fig. 5, characterized by non-smooth contact dynamics and a 12-dimensional state. The robot started at a random state, with the horizontal position $x_h \in [-10, 60]$ m, and has to reach $x_h \geq 50$ m without falling, while subjected to disturbances in horizontal and vertical velocity. The velocity disturbances are modeled as a zero-mean Gaussian truncated at $4\sigma = 0.15$ m/s. The



Fig. 5. Snapshot of the Hopper legged system.

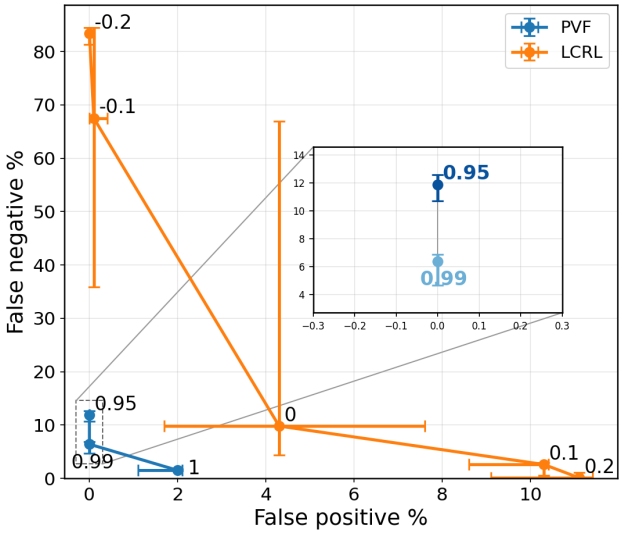


Fig. 6. Comparison between PVF and LCRL in terms of accuracy, for Hopper experiment. Thresholds for fail detection reported near the medians.

comparison between PVF and LCRL is reported in Fig. 6. Even in this case PVF was more accurate on average than LCRL both in terms of FN and FP at the nominal thresholds. Under nominal conditions, the median FN rate for LCRL was about 4% higher than for PVF, although the minimum FN rates across runs were very close, whereas PVF performed markedly better in terms of FP, achieving nearly 0%. This could be due to the fact that the disturbance policy learned by LCRL was not truly worst-case, and that PVF encountered more adversarial disturbances during data collection.

C. Computation times

Table I reports the mean computation time for each method. In our implementation, PVF required more computation, primarily because of its larger transition dataset and repeated stochastic rollouts. For Hopper, LCRL was run with a fixed policy, which explains its comparatively low computation time on this system w.r.t. to Drones racing; nonetheless, allowing LCRL additional computation time did not lead to any improvement in performance. For PVF, the size of the generated dataset is the main driver of computation time. We believe that the method could potentially work well even with smaller datasets, yielding significantly shorter data collection and training times than those reported in Table I, and that implementing a technique that concentrates the training where the learned probability error relative to the targets is larger could further lower the times. However, more work is needed to validate such hypothesis.

TABLE I

MEAN COMPUTATION TIME ACROSS THE DIFFERENT RUNS.

	DRONES RACING	HOPPER
LCRL	3175 s	361 s
PVF data collection	677 s	7145 s
PVF training	9367 s	23502 s
PVF total	10044 s	30647 s

TABLE II

FAILURES-ONLY NETWORK RESULTS FOR HOPPER.

METRIC	VALUE (%)
True Positive	74.6
False Positive	23.6
True Negative	0.0
False Negative	0.0
Unclear Cases	1.8

D. Ablation: undiscounted avoid-only formulation

To validate the importance of the target set (as discussed in Section II-D), even in cases where our main interest is avoiding the constraint set, we train a PVF on Hopper, considering only failures. This approach is equivalent to SQRL [10] without discounting. If Hopper falls within 500 steps the episode fails. We computed the TD target as:

$$V_{target}(x, x') := \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ V(x') & \text{otherwise} \end{cases} \tag{15}$$

We validated the accuracy of the learned value function computing the Wilson confidence intervals, as in the previous tests. The classification threshold was set to 0.99, corresponding to a 1% failure probability.

The results reported in Table II show that this approach was not able to detect the safe states, because the network always classified the states as unsafe, acting extremely cautiously. Intuitively, this can be explained by the fact that, since the training is self-supervised, except for the failure signals, the network is only forced to go towards the target associated to failures.

E. Application: model predictive shielding

As an example of a potential application, we have used the learned PVF and the associated (backup) policy as a shield for another policy [1]. By selecting a desired probability level, the sublevel-set of $V(\cdot)$ approximates the associated probabilistic reach-avoid set. Testing membership to this set, a so-called safety filter can decide when to trigger the switch from a nominal policy to the backup policy. The settings of

TABLE III

MEAN $\pm$ STANDARD DEVIATION OF SUCCESS COUNTS (OUT OF 100 TRIALS) ACROSS 11 SEEDS, BY NOISE LEVEL.

	NOISE LEVEL (4σ)		
	0.10	0.15	0.20
SHIELDED	99.1 ± 1.6	97.2 ± 4.2	91.0 ± 8.3
UNSHIELDED	97.1 ± 5.2	90.8 ± 7.7	83.1 ± 10.0

the reach-avoid task are the same as in Section III-B.2. For this test we use two policies:

- the *nominal policy*, trained on Hopper without disturbances, with a reward focused on forward speed;
- the *backup policy*, trained on Hopper with disturbances, using a reward that assigns a lower weight to forward speed; among the checkpoints saved during RL training, we retain the one with the highest mean episode length. The probabilistic reach-avoid set is computed using this policy.

Starting from the same initial state, whose estimated probability of success exceeds 0.99, the Hopper's task is to move forward 6 m without falling. We performed 100 rollouts of the nominal policy and, under the same disturbance realizations, 100 rollouts of the nominal policy shielded by the backup policy. The shield triggers whenever the mean predicted probability of success at the next state falls below 0.99. To estimate this probability, we query the simulator n times from the current state and average the PVF estimate over the resulting n successor states. Three noise levels were tested, even though the used PVF network was trained only on rollouts subject to zero-mean Gaussian noise truncated at 4σ, with a maximum disturbance of $4\sigma = 0.15$. The test was repeated for 11 different seeds, using $n = 100$ with identical disturbance realizations applied to both the shielded and unshielded cases. Mean and standard deviation of the success rate across all runs are reported in Table III. With shielding active, the number of failures was consistently lower than in the unshielded case, even for noise levels exceeding those used to compute the PVF. At the disturbance bound $4\sigma = 0.15$, the shielded execution achieved, on average, 6.4% more successes than the unshielded policy, with roughly half its standard deviation. At the higher noise bound $4\sigma = 0.20$, the gap between the two methods widened, with shielding achieving 7.9% more successes on average. We believe that the gap between the desired success probability (99%) and the lower observed success rate is due to the distribution mismatch between the training-set states and the states visited by the nominal policy at deployment.

IV. CONCLUSIONS

In this article we presented PVF, a novel method for computing probabilistic reach-avoid sets. We compared it against LCRL, a recent and promising HJ-reachability method for computing robust reach-avoid sets. Under random disturbances, PVF achieved better classification accuracy when evaluated against a ground truth certified via Wilson confidence intervals at a high confidence level. Despite these encouraging results, the proposed approach still presents some limitations that would require further attention. Most importantly, our method currently assumes that a policy is available for controlling the system. We plan to extend our method to jointly learn the probabilistic value function and a policy to maximize the associated success probability. Another critical point is computational efficiency, which is currently a weakness of the method. In this respect we want to investigate ways to intensify training on state-space regions where the probability estimate is more uncertain. Finally, we plan to apply and validate our method on larger systems, such as quadruped robots navigating obstacle-rich environments, and investigate the use of PVF for shielding in safety critical applications.

REFERENCES

[1] O. Bastani, S. Li, and A. Xu, "Safe Reinforcement Learning via Statistical Model Predictive Shielding," in *Robotics Science and Systems*, 2021.

[2] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.

[3] L. Grüne and J. Pannek, *Nonlinear model predictive control*. Springer, 2017.

[4] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *IEEE Conference on Decision and Control (CDC)*, 2017.

[5] S. Bansal and C. J. Tomlin, "Deepreach: A deep learning approach to high-dimensional reachability," in *IEEE International Conference on Robotics and Automation*, 2021.

[6] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, "Bridging hamilton-jacobi safety analysis and reinforcement learning," in *IEEE International Conference on Robotics and Automation*, 2019, pp. 8550–8556.

[7] 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 (RA-L)*, vol. 10, no. 4, 2025.

[8] A. Abate, M. Prandini, J. Lygeros, and S. Sastry, "Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems," *Automatica*, vol. 44, no. 11, 2008.

[9] S. Summers and J. Lygeros, "Verification of discrete time stochastic hybrid systems: A stochastic reach-avoid decision problem," *Automatica*, vol. 46, no. 12, 2010.

[10] K. Srinivasan, B. Eysenbach, S. Ha, J. Tan, and C. Finn, "Learning to be safe: Deep rl with a safety critic," arXiv:2010.14603, 2020.

[11] A. La Rocca, M. Saveriano, and A. Del Prete, "VBOC: Learning the Viability Boundary of a Robot Manipulator using Optimal Control," *IEEE Robotics and Automation Letters*, 2023.

[12] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, no. 1, 1988.

[13] 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," 2019. [Online]. Available: https://arxiv.org/abs/1509.02971

[14] K. Zakka, B. Tabanpour, Q. Liao, M. Haiderbhai, S. Holt, J. Y. Luo, A. Allshire, E. Frey, K. Sreenath, L. A. Kahrs, C. Sferrazza, Y. Tassa, and P. Abbeel, "Mujoco playground: An open-source framework for gpu-accelerated robot learning and sim-to-real transfer." 2025. [Online]. Available: https://github.com/google-deepmind/mujoco_playground

[15] H. Durrant-Whyte, N. Roy, and P. Abbeel, "Infinite-horizon model predictive control for periodic tasks with contacts," in *Robotics: Science and Systems VII*, 2012.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv:1707.06347, 2017.

[17] L. D. Brown, T. T. Cai, and A. DasGupta, "Interval estimation for a binomial proportion," *Statistical Science*, vol. 16, no. 2, 2001.