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

Deep Reinforcement Learning for Reach-Avoid-Stay Problems

Gabriel Chenevert, Jingqi Li, Achyuta Kannan, Sangjae Bae, and Donggun Lee

***Abstract*— Reach-Avoid-Stay (RAS) tasks are essential in applications where systems must safely reach a target set and remain within it under all bounded disturbances. Existing approaches either struggle to compute the maximal robust RAS set—the set of all states from which the RAS task is achievable—or are limited in handling general dynamic systems. To address these challenges, this paper proposes a two-step deep reinforcement learning framework that jointly learns the maximal robust RAS set and the corresponding control policy. The first step identifies the maximal robust control-invariant set within the target set and derives a policy that ensures the system remains within it. The second step computes the maximal robust reach-avoid (RA) set using this invariant set as the target, and it is proven that this RA set is equivalent to the maximal robust RAS set. Leveraging this result, a switching policy is constructed from the two step-wise policies, which constitutes a valid policy guaranteeing completion of the RAS task. Simulation results demonstrate that the proposed framework (1) computes the exact maximal robust RAS set in the absence of training errors, yielding the least restrictive RAS policy, and (2) identifies the RAS set with high accuracy while outperforming baseline methods on RAS tasks.**

I. Introduction

Modern robotic motion planning and control methods aim to generate safe trajectories in complex environments. A prominent approach is reachability analysis [1], which determines the set of states from which a control policy exists to safely accomplish a given task. Reachability analysis has been applied to diverse systems, including aircraft collision avoidance [2], human-robot interaction [3], and robot manipulation [4]. Among reachability formulations, the *reach-avoid* (RA) problem is one of the most widely used: the goal is to drive a system to a target while satisfying safety constraints. Recent advances in RA methods involve leveraging deep learning [5] and reinforcement learning (RL) [6]–[13] to enhance computational efficiency for high-dimensional systems. Other work has extended RA to stochastic processes [14].

However, RA methods can cause the system to leave the target set [15]. This occurs when the target strictly contains and is not equivalent to the robust viability kernel, the maximal robust control-invariant subset of the target set under all bounded disturbances. If the system reaches the target set but not the robust viability kernel, it will exit the target set. In vehicle applications, for example, this tends to occur when the system enters the target set with excessive velocity and lacks sufficient control to slow and remain within it. RA methods [15] are unable to determine which states are in the robust viability kernel, and are therefore insufficient for tasks where staying within the target set is critical.

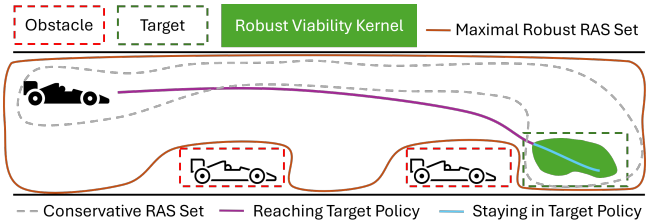

Fig. 1. The reach-avoid-stay (RAS) task involves guiding a system from an initial state, to a target set where it remains while safely avoiding all obstacles against bounded disturbances. The maximal robust RAS set (orange region) is the set of all states where the RAS task can be performed. The robust viability kernel, the solid green region, is the set of all states which can safely stay within the target. A RAS policy must first direct the system from within the RAS set to the robust viability kernel, and then remain within the robust viability kernel. If a RAS method has only identified the conservative RAS set (gray dashed line), it may encounter false negative safety classifications, restricting the range of the control policies.

Reachability tasks that require staying within the target set have motivated the introduction of the *reach-avoid-stay* (RAS) problem. The RAS problem seeks the set of initial states and a corresponding policy such that the system can be driven safely to the target set and kept there indefinitely despite all bounded disturbances. This set is called the *maximal robust RAS set*. Together with its associated policy, the maximal robust RAS set provides a guarantee that the RAS task can be achieved from the *largest* set of states. Without this set, the policy may exclude feasible states or mistakenly include unsafe ones. False positives, states incorrectly classified as belonging to the RAS set, are dangerous, as the task can not be safely completed from those states, leading to safety violations. False negatives exclude feasible states, leading to unnecessarily conservative policies.

Several methods have been developed for the RAS problem; however, each approach encounters distinct challenges. Control-Lyapunov-Barrier function (CLBF)-based methods [16] require the design of CLBFs, which is challenging for high-dimensional or nonlinear systems. Reinforcement learning-based approaches [7], [17] attempt to address the challenges of designing CLBFs, but fail to identify the maximal robust RAS set, resulting in inaccurate safety classification and conservative policies. Funnel-based control

Gabriel Chenevert, and Donggun Lee are with the Department of Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, USA. {gechenev,donggun_lee}@ncsu.edu. Jingqi Li is with the Department of Aerospace Engineering and Engineering Mechanics, at University of Texas, Austin, USA. jingqi.li@austin.utexas.edu. Achyuta Kannan is with the Department of Computer Science , North Carolina State University, Raleigh, USA. ackannan@ncsu.edu. Sangjae Bae is with Honda Research Institute, sbae@honda-ri.com

This research is supported by the Honda Research Institute.

methods [18], [19] circumvent the challenging CLBF design process, but assume that the system can stop instantly. These limitations cause current methods for the RAS problem to be either difficult to design, unable to identify the maximal RAS set, or dependent on restrictive assumptions about dynamical systems that limit their general applicability.

To address the above limitations, we propose a two-step deep-RL framework for RAS problems that computes the maximal robust RAS set and its corresponding policy. First, we compute the robust viability kernel within the target set. Second, we apply the RA method [15], replacing the target set with the robust viability kernel. We introduce a value function transformation that yields a function strictly positive within the robust viability kernel and non-positive elsewhere, enabling it to serve as a reward function for the RA method. Finally, we design a switching policy for RAS tasks by combining the policies from both steps.

The primary contributions of our learning framework are:

1) **Theoretical Contribution:** We prove that our RAS framework identifies the maximal robust RAS set in the absence of training errors, and that the resulting RAS policy achieves the RAS task for all states in this set under all bounded disturbances. These results are enabled by our proposed value function transformation, without which the integration of the two prior reachability frameworks cannot solve the RAS problem.
2) **Learning an Accurate Maximal Robust RAS Set Via Deep-RL**: We develop a deep-RL framework to learn the maximal robust RAS set for high-dimensional systems, and demonstrate that even with the training error introduced by deep-RL methods, our framework characterizes the maximal robust RAS set with high accuracy.
3) **Practical Applicability**: Our approach handles complex environments for which designing CLBFs [16] is difficult, and does not rely on the instant stopping assumption made by funnel-based methods [18], [19], enabling broader applicability.
4) **Empirical Validation**: We conduct simulations to validate our framework's effectiveness. Our approach outperforms CLBF-based methods [16] at identifying the maximal robust RAS set, reducing false negatives, yielding a less cautious policy while retaining safety. It also achieves higher success rates for the RAS task compared to a reach-avoid method [13].

II. Background: Reach-Avoid Analysis

We consider nonlinear dynamical systems, described by

$$\mathrm{x}_{t+1} = f(\mathrm{x}_t, u_t, d_t), \tag{1}$$

where $\mathrm{x}_t \in \mathbb{R}^n$ is the state, $u_t \in U \subset \mathbb{R}^{m_u}$ is control, and $d_t \in D \subset \mathbb{R}^{m_d}$ is a disturbance at time t. For robust analysis, we consider an adversarial disturbance [11], which can model environmental uncertainty and bounded discrepancies between the simulation and the physical system. If our control policy succeeds under such conditions, it is guaranteed to succeed under less severe disturbances.

Consider an open set $T \subseteq \mathbb{R}^n$ representing the target set. We define a Lipschitz continuous bounded reward function g which indicates if a state within the target set:

$$g(x) > 0 \Leftrightarrow x \in T. \tag{2}$$

Similarly, we consider an open set $C \subseteq \mathbb{R}^n$ representing the constraint set (safe region). We define an additional Lipschitz continuous bounded reward function exists such that:

$$l(x) > 0 \Leftrightarrow x \in C. \tag{3}$$

We present the RA problem from prior work in [13] and [15]. The objective is to identify the maximal robust RA set along with its corresponding control policy $\pi_u : \mathbb{R}^n \to U$ such that for all disturbance policies, $\pi_d : \mathbb{R}^n \to D$, the system reaches the target set while remaining within the constraint set. The resulting maximal robust RA set is defined as:

$$\mathcal{RA} \coloneqq \left\{ x \,\middle|\, \begin{array}{l} \exists \pi_u \text{ s.t. } \forall \pi_d, \exists \tau \in \{0, ...\} \text{ s.t.} \\ 1)\ \mathrm{x}_t \in C, t = \{0, ..., \tau\} \text{ and} \\ 2)\ \mathrm{x}_\tau \in T \end{array} \right\}, \tag{4}$$

where $\mathrm{x}_{t+1} = f(\mathrm{x}_t, \pi_u(\mathrm{x}_t), \pi_d(\mathrm{x}_t))$, and $\mathrm{x}_0 = x.$. While the RA task only requires that the trajectory reaches T momentarily, the RAS task requires it to remain in T for all time after the reaching time τ. Therefore, we extend the RA problem to RAS, which is the core problem of our work.

III. Problem Definition: Reach-Avoid-Stay

Consider the dynamical system described in (1), along with the target set T defined in (2), and the constraint set C defined in (3). The RAS problem is to identify the maximal robust RAS set and corresponding control policy, $\pi_u : \mathbb{R}^n \to U$ such that for all disturbance policies, $\pi_d : \mathbb{R}^n \to D$, the system remains within the constraint set and stays in the target set after reaching it. The maximal robust RAS set is defined as:

$$\mathcal{RAS} \coloneqq \left\{ x \,\middle|\, \begin{array}{l} \exists \pi_u \text{ s.t. } \forall \pi_d, \\ 1)\ \mathrm{x}_t \in C, t = \{0, ...\} \text{ and} \\ 2)\ \exists \tau \in \{0, ...\} \text{ s.t. } \mathrm{x}_t \in T, t = \{\tau, ...\} \end{array} \right\}, \tag{5}$$

where x_t is governed by a RAS policy π_u and a disturbance policy π_d:

$$\mathrm{x}_{t+1} = f(\mathrm{x}_t, \pi_u(\mathrm{x}_t), \pi_d(\mathrm{x}_t)), \text{ and } \mathrm{x}_0 = x. \tag{6}$$

The first condition in (5) encodes the safety condition, while the second encodes the reach and stay conditions.

IV. Two-Step Deep Deterministic Policy Gradient for RAS

In this section, we propose a two-step RL framework to compute $\mathcal{RAS}$ and the corresponding policy. Our approach integrates the RL method for RA tasks [13] and the robust viability kernel [20]. In the first step, we identify the robust viability kernel set within the target set and a control policy for remaining within it. In the second step, we apply the

RA formulation, but replace the target set with the robust viability kernel specified in the first step and identify a policy for reaching the robust viability kernel. We prove that the resulting $\mathcal{RA}$ is equivalent to $\mathcal{RAS}$ and that this $\mathcal{RAS}$ is maximal and robust; no state outside $\mathcal{RAS}$ subject to adversarial disturbances can achieve the RAS task. Section IV-A presents the robust viability kernel analysis. Building on this result, in Section IV-B, we propose our main framework: the RAS formulation, along with its proof and algorithms.

A. Computing the Robust Viability Kernel

In this subsection, we leverage the discounted formulation from [13] to find the largest robust control-invariant set within the target set while avoiding obstacles. This set is referred to as the robust viability kernel:

$$\mathcal{AS} \coloneqq \left\{ x \;\middle|\; \begin{array}{l} \exists \pi_u \text{ s.t. } \forall \pi_d, \\ \mathrm{x}_t \in C \cap T, t = \{0, ...\} \end{array} \right\}. \tag{7}$$

We define a discounted value function, H, that characterizes the robust viability kernel within the target set while remaining in the constraint set:

$$H(x) \coloneqq \sup_{\pi_u} \inf_{\pi_d} \inf_{t=\{0,...\}} \gamma^t \bar{g}(\mathrm{x}_t), \tag{8}$$

where $\bar{g}$ characterizes the target set excluding the avoid region:

$$\bar{g}(x) \coloneqq \min\left(g(x), l(x)\right), \tag{9}$$

and $\gamma \in (0, 1)$ is the discount factor. The value of $\bar{g}$ is positive if and only if the state is within both the target set and the constraint set. H evaluates the discounted value of $\bar{g}$ across an infinite time horizon. If at any state is outside the target set or the constraint set, H will be negative, indicating that the initial state was outside of $\mathcal{AS}$.

The following remark presents the Bellman equation for H that characterizes $\mathcal{AS}$ and the corresponding policy that enables the system to safely stay in the target if the state is in $\mathcal{AS}$.

Remark 1 *The discounted-value function H, defined in* (8)*, is the unique solution to the following Bellman equation [21]:*

$$H(x) = \min\left\{ \bar{g}(x), \gamma \max_{u \in U} \min_{d \in D} H(f(x, u, d)) \right\}, \tag{10}$$

since the corresponding Bellman operator is a contraction mapping [22]. The robust viability kernel set within the target set while avoiding unsafe regions is the level-zero set of H:

$$\mathcal{AS} = \{x \mid H(x) = 0\}. \tag{11}$$

Note that H is non-positive for all $x \in \mathbb{R}^n$: $H(x) = 0$ for $x \in \mathcal{AS}$ and $H(x) < 0$ for $x \notin \mathcal{AS}$. Also, for $x \in \mathcal{AS}$, any control that satisfies the following equation enables staying in the target set and constraint set for any bounded disturbance:

$$\pi_H(x) \in \arg\max_{u \in U} \min_{d \in D} H(f(x, u, d)), \tag{12}$$

where $\pi_H : \mathbb{R}^n \to U$ is a policy that keeps trajectories safe indefinitely if they start in $\mathcal{AS}$.

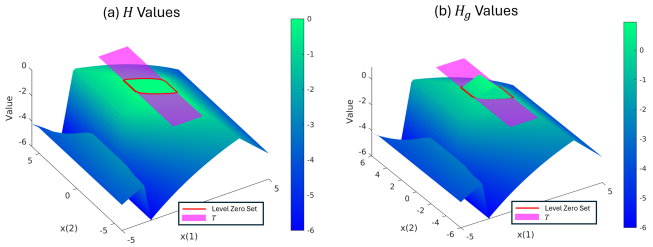

Fig. 2. These figures show the transformation from H shown in (a) to H_g shown in (b) for the cart example whose dynamics and target set formulation are available in Section V-A. The zero-level set of H is the same as the super-zero-level set of H_g. H_g is strictly positive in $\mathcal{AS}$ as demonstrated in Theorem 1.

B. Computing the Maximal Robust RAS Set

In this subsection, we utilize the RA formulation from [13] to specify $\mathcal{RAS}$ and its corresponding policy by replacing T in (4) with $\mathcal{AS}$. For this process, we need to use a value function whose super-zero-level set is $\mathcal{AS}$ to provide an incentive to reach $\mathcal{AS}$ as quickly as possible. Without this, in RA analysis the system receives the same reward whether it reaches $\mathcal{AS}$ in finite time or only when time is very large approaching infinity. In such a case, the policy does not actively drive the system toward $\mathcal{AS}$. Therefore, since H is identically zero on $\mathcal{AS}$, it is not a suitable candidate.

To provide a reaching incentive, we design a new value function $H_g : \mathbb{R}^n \to \mathbb{R}$ as follows.

$$H_g(x) \coloneqq \begin{cases} H(x) \text{ in (8)}, & \text{if } H(x) < 0, \\ g(x) \text{ in (2)}, & \text{otherwise.} \end{cases} \tag{13}$$

Theorem 1 shows that the super-zero-level set of H_g represents the same robust viability kernel $\mathcal{AS}$, while the values inside the set are strictly positive.

Theorem 1 *H_g is positive if and only if x is within $\mathcal{AS}$.*

Proof: Consider two cases: $x \in \mathcal{AS}$ and $x \notin \mathcal{AS}$. Case 1: $x \in \mathcal{AS}$. Since Remark 1 implies $H(x) = 0$, we have $H_g(x) = g(x)$ by the definition of H_g in (13). By the definition of $\mathcal{AS}$ in (7), $g(x) > 0$. Thus, $H_g(x) = g(x) > 0$. Case 2: $x \notin \mathcal{AS}$. By Remark 1 and (8), it follows that $H(x) < 0$. Then, by (13), we conclude that $H_g(x) = H(x) < 0$. ■

An example of H_g constructed for the cart example in Section V-A can be found in Fig. 2 (b).

To characterize $\mathcal{RAS}$, we use H_g to define another discounted-value function V:

$$V(x) \coloneqq \sup_{\pi_u} \inf_{\pi_d} \sup_{t=\{0,...\}} \min\{\gamma^t H_g(\mathrm{x}_t), \inf_{s=\{0,...,t\}} \gamma^s l(\mathrm{x}_s)\}. \tag{14}$$

Subsequently, the Bellman equation for V and the corresponding policy can be derived as described by the following remark.

Remark 2 *The value function V, defined in* (14)*, is the*

unique solution to the following Bellman equation [13]:

$$V(x) = \min\{l(x), \max\{H_g(x), \gamma \max_{u\in U} \min_{d\in D} V(f(x,u,d))\}\}, \tag{15}$$

since the corresponding Bellman operator is a contraction mapping [13]. Any policy that satisfies the following equation enables reaching the robust viability kernel within the target set $\mathcal{AS}$ while satisfying the constraint set:

$$\pi_V(x) \in \arg\max_{u\in U} \min_{d\in D} V(f(x,u,d)), \tag{16}$$

where $\pi_V : \mathbb{R}^n \to U$ is the corresponding RA policy [13].

Now, we present our main theorem, which states that the value function V characterizes the maximal robust RAS set $\mathcal{RAS}$ (5).

Theorem 2 *The super-zero-level set of V (14) is the maximal robust RAS set, $\mathcal{RAS}$ (5).*

$$\mathcal{RAS} = \{x \mid V(x) > 0\} \tag{17}$$

Proof: (i) Suppose $x \in \mathcal{RAS}$. Then there exists π_u^* such that, for all π_d, $l(\mathrm{x}_t^*) > 0$, $t = 0, ...,$ and there exists τ^* such that, $g(\mathrm{x}_t^*) > 0$, $t = \tau, ...,$ where x^* solves (6) for π_u^* and π_d. Note that τ^* and x^* depend on π_d. These conditions can be rewritten as follows: for all π_d, there exists τ^* such that $l(\mathrm{x}_t^*) > 0$ for $t = 0, ..., \tau^*$ and $\bar{g}(\mathrm{x}_t^*) > 0$ for $t = \tau^*, ...,$ which implies that $\mathrm{x}_{\tau^*}^* \in \mathcal{AS}$. By (13), $H_g(\mathrm{x}_{\tau^*}^*) = g(\mathrm{x}_{\tau^*}^*) > 0$. Therefore, we have

$$\inf_{\pi_d} \sup_{\tau=\{0,...\}} \min\{\gamma^\tau H_g(\mathrm{x}_\tau^*), \min_{t=\{0,...,\tau\}} \gamma^t l(\mathrm{x}_t^*)\} > 0. \tag{18}$$

Since π_u^* may not be a maximizer for V,

$$V(x) \geq \inf_{\pi_d} \sup_{\tau=\{0,...\}} \min\{\gamma^\tau H_g(\mathrm{x}_\tau^*), \inf_{s=\{0,...,\tau\}} \gamma^s l(\mathrm{x}_s^*)\} > 0. \tag{19}$$

(ii) Suppose $V(x) > 0$. Then, there exists π_V such that, for any π_d, there exists τ^* such that

$$H_g(\mathrm{x}_{\tau^*}^*) > 0, l(\mathrm{x}_t^*) > 0, t = \{0, ..., \tau^*\}, \tag{20}$$

where x^* solves (6) for π_V, π_d. Since $H_g(\mathrm{x}_{\tau^*}^*)$ is positive, there exists another policy π_H such that, for any π_d,

$$g(\mathrm{x}_t^{H*}) > 0, l(\mathrm{x}_t^{H*}) > 0, \ t = \{\tau^*, ...\}, \tag{21}$$

where x_t^{H*} solves (6) for π_H, π_d, and $\mathrm{x}_{\tau^*}^H = \mathrm{x}_{\tau^*}$. Combining π_V and π_H into a unified policy $\pi_{\mathcal{RAS}}$ yields:

$$\pi_{\mathcal{RAS}}(x) = \begin{cases} \pi_V(x) & \text{if } H_g(x) \leq 0 \\ \pi_H(x) & \text{otherwise.} \end{cases} \tag{22}$$

Using $\pi_{\mathcal{RAS}}$, for each π_d, the corresponding state trajectory $\bar{\mathrm{x}}$ solving (6) is

$$\bar{\mathrm{x}}_t = \begin{cases} \mathrm{x}_t^* & t = \{0, ..., \tau^*\} \\ \mathrm{x}_t^{H*} & t = \{\tau^*, ...\}. \end{cases} \tag{23}$$

Therefore, $\pi_{\mathcal{RAS}}$ satisfies, for any π_d, $l(\bar{\mathrm{x}}_t) = l(\mathrm{x}_t) > 0$ for $t \leq \tau^*$ by (20), $l(\bar{\mathrm{x}}_t) = l(\mathrm{x}_t^{H*}) > 0$ for $t > \tau^*$ by (21), and $g(\bar{\mathrm{x}}_t) = g(\mathrm{x}_t^{H*}) > 0$ for $t \geq \tau^*$ by (21). Thus, $x \in \mathcal{RAS}$, and $V(x) > 0$. ■

The proof of Theorem 2 implies that if the state is within $\mathcal{RAS}$, the system can accomplish the RAS task under the switching policy (22); otherwise, the task cannot be achieved.

Remark 3 *Within $\mathcal{RAS}$, π_V drives the system to $\mathcal{AS}$, but it cannot guarantee safety once inside. In contrast, π_H guarantees safe invariance in $\mathcal{AS}$, which necessitates switching from π_V to π_H. The switch occurs when $H_g(x) > 0$, which by Theorem 2 indicates that the system is in $\mathcal{AS}$. If $x \notin \mathcal{RAS}$, π_V will be unable to safely reach $\mathcal{AS}$, thus the system will be unable to perform the RAS task.*

C. A Two-step Deep-RL Framework for Learning the Maximal Robust RAS Set and the Associated Policies

We summarize our two-step framework in Algorithm 1. First, we learn a value function H that characterizes $\mathcal{AS}$, along with the corresponding stay policy π_H. Second, to encourage the system to reach the target set, we construct H_g according to (13), which assigns positive values within $\mathcal{AS}$ and negative values outside it. Third, we learn a value function V that characterizes the RAS set and derive the associated reach-avoid policy π_V.

To address general high-dimensional nonlinear systems, we implement Algorithm 1 using Deep Deterministic Policy Gradient (DDPG) [23]–[25]. For low-dimensional systems, where the curse of dimensionality has less impact, DDPG can be replaced with tabular Q-learning [26] or similar optimization methods.Guidance on training specific to our examples is provided in Section V-B.3.

V. Simulation

In this section, we demonstrate the following:

- **Claim 1:** Our framework identifies the maximal robust RAS set, $\mathcal{RAS}$, if there is no training error. We verify that this is a super-set of the RAS set identified by the CLBF baseline [16] resulting in more accurate safety classification and less conservative policies. (Section V-A)
- **Claim 2:** Even with training error, our framework identifies $\mathcal{RAS}$ with over 95% accuracy. (Section V-C)
- **Claim 3:** Our framework achieves a higher success rate than a baseline RA method [15] in high-dimensional systems. (Section V-C)

For Claim 1, to remove training error induced by neural networks, we consider a low-dimensional system that is tractable for our framework using tabular Q-learning [27]. We construct a CLBF [16] to identify a RAS set for comparison in Section V-A.

To validate Claims 2 and 3, we present two high-dimensional systems, the descriptions of which are provided in Section V-B. The analysis of our framework's performance in comparison to a baseline RA method [15] is presented in Section V-C. We are unable to use a CLBF-based or funnel-based method as baselines for the high-dimensional

Algorithm 1: A Two-step Framework for RAS Problems:

```
// This training can be performed
   using DDPG or any value iteration
   based method.
```

1 **Train an approximation function H^ϕ, a control policy π^θ_H, and a disturbance policy $\pi^\rho_{d_H}$:** Update H^ϕ by minimizing the Bellman error:

$$\|H^\phi(x) - \min\{\bar{g}(x), \gamma H^\phi(f(x, \pi^\theta_H(x), \pi^\rho_{d_H}(x)))\}\|_2. \tag{24}$$

Use $H^\phi\Big(f\big(x, \pi^\theta_H(x), \pi^\rho_{d_H}(x)\big)\Big)$ to train π^θ_H and $\pi^\rho_{d_H}$ such that π^θ_H maximizes H^ϕ and $\pi^\rho_{d_H}$ minimizes H^ϕ.

2 **Create H^ϕ_g:** Combine H^ϕ with g according to (13).

3 **Train an approximation function V^φ, a control policy for π^ϑ_V, and a disturbance policy $\pi^o_{d_V}$:** Update V^φ by minimizing the Bellman error:

$$\|V^\varphi(x) - \min\{l(x), \max\{H^\phi_g(x), \gamma V^\varphi(f(x, \pi^\vartheta_V(x), \pi^o_{d_V}(x)))\}\}\|_2. \tag{25}$$

Use $V^\varphi\Big(f\big(x, \pi^\vartheta_V(x), \pi^o_{d_V}(x)\big)\Big)$ to train π^ϑ_V and $\pi^o_{d_V}$ such that π^ϑ_V maximizes V^φ and $\pi^o_{d_V}$ minimizes V^φ.

systems since designing a CLBF found in [16] would be impractical, and they violate the funnel method assumption that the system can instantly stop [18].

A. Two-Dimensional Example: Double Integrator

This subsection demonstrates that our framework provides the maximal robust RAS set compared to another set computed by a CLBF baseline method [28].

Consider a two-dimensional cart system:

$$\mathrm{x}_{t+1}(1) = \mathrm{x}_t(1) + \Delta t \mathrm{x}_t(2) + \Delta t^2(u_t + d_t), \tag{26}$$
$$\mathrm{x}_{t+1}(2) = \mathrm{x}_t(2) + \Delta t(u_t + d_t), \tag{27}$$

where $\mathrm{x}_t(1)$ is the position of the cart along a track at time t, $\mathrm{x}_t(2)$ is the velocity of the cart at time t, and both control ($u_t \in [-3, 3]$) and disturbance ($d_t \in [-2, 2]$) are acceleration. A target is located at $x(1) = 0$, and an obstacle is located at $x(1) = -3$. Both the obstacle and the target have a radius of one. We design one reward function g and one constraint function l as follows:

$$g(\mathrm{x}_t) = 1 - |\mathrm{x}_t(1) - 0|, \quad l(\mathrm{x}_t) = |\mathrm{x}_t(1) - 3| - 1. \tag{28}$$

Demonstration of Claim 1: Our framework provides $\mathcal{RAS}$, a superset of the RAS set provided by the CLBF in [16] as supported by Theorem 2. The volume of $\mathcal{RAS}$ shown in Fig. 3 is 28.67, compared to the CLBF baseline of 1.98.

Demonstration of Switching Control Policy $\pi_{\mathcal{RAS}}$: $\pi_{\mathcal{RAS}}$ switching control policy achieves the RAS task despite being opposed by an adversarial disturbance as can be observed from the trajectory shown in Fig. 3. The trajectory starts from within $\mathcal{RAS}$, and is driven by π_V safely to $\mathcal{AS}$. Upon reaching $\mathcal{AS}$, H_g becomes positive, and the control policy is changed to π_H which successfully keeps the system within $\mathcal{AS}$.

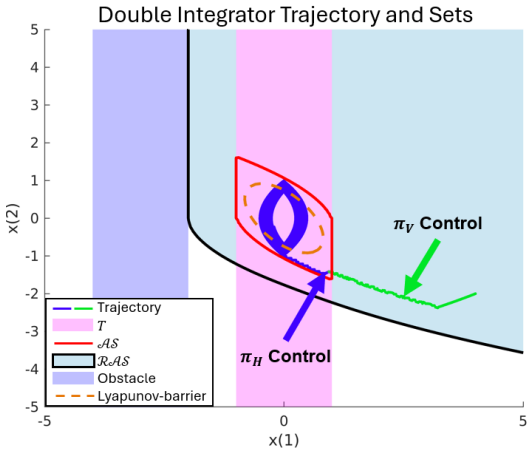

Fig. 3. This figure illustrates $\mathcal{RAS}$ (within the black lines) and a trajectory driven by $\pi_{\mathcal{RAS}}$ in green and blue. The trajectory successfully achieves the RAS task under adversarial disturbances. π_V is used to reach $\mathcal{AS}$ where control is switched to π_H to enable staying.

B. Complex System Experiments

Here we present two high-dimensional systems to demonstrate Claim 2 that our framework identifies $\mathcal{RAS}$ with over 95% accuracy, and Claim 3, our framework achieves a higher success rate than a baseline RA method [15].

1) VTOL Taxi (VTOL): Consider a VTOL taxi in a city. The taxi must avoid collisions with buildings and other aircraft while moving to a drop-off location. Throughout this process, wind disrupts the taxi's motion. In this demonstration, there is one ego VTOL taxi and two other VTOLs which the ego VTOL must avoid.

We model this system as an 18-dimensional state $\mathrm{x}_t = [\mathrm{p}^e_t; \mathrm{v}^e_t; \mathrm{p}^1_t; \mathrm{v}^1_t; \mathrm{p}^2_t; \mathrm{v}^2_t]$, where $\mathrm{p}^e_t \in \mathbb{R}^3$ and $\mathrm{v}^e_t \in \mathbb{R}^3$ represent the ego VTOL's position and velocity, respectively, at time t. $\mathrm{p}^1_t \in \mathbb{R}^3$ and $\mathrm{v}^1_t \in \mathbb{R}^3$ denote the position and velocity of the first autonomous VTOL at time t, and $\mathrm{p}^2_t \in \mathbb{R}^3$ and $\mathrm{v}^2_t \in \mathbb{R}^3$ represent those of the second autonomous VTOL. The ego VTOL uses the following dynamics.

$$\mathrm{p}^e_{t+1} = \mathrm{p}^e_t + \Delta t \mathrm{v}^e_t, \tag{29}$$
$$\mathrm{v}^e_{t+1} = v^e_t + \Delta t(u_t + d_t), \tag{30}$$

where $u_t \in \mathbb{R}^3$ is a velocity input to the VTOL, and the disturbance $d_t \in \mathbb{R}^2$ represents wind and dynamics modeling errors. Each of the two obstacle VTOLs is modeled with double integrator dynamics, where the control input is designed as PD control to follow a fixed target, with the acceleration bounded between -0.5 and 0.5. We use the following l and g equations:

$$l(\mathrm{x}_t) = \min\Bigg(\|\mathrm{p}_t^e - \mathrm{p}_t^1\|_2 - 2\bar{r}_d, \|\mathrm{p}_t^e - \mathrm{p}_t^2\|_2 - 2\bar{r}_d, \|[\mathrm{p}_t^e(1); \mathrm{p}_t^e(2)] - \bar{b}\|_2 - \bar{r}_d\Bigg),$$
$$g(\mathrm{x}_t) = \bar{r} - \|\mathrm{p}_t^2 - \bar{\mathrm{p}}\|_2,$$

where $\bar{r}_d$ is the radius of the VTOL, $\bar{b} \in \mathbb{R}^2$ is the position of the building, $\bar{p}$ is the position of the target, and $\bar{r}$ is the radius of the target.

2) Nonlinear Harvester Unloading (Harvester): Consider a tractor unloading crops from a combine harvester in motion. The tractor must pull alongside the harvester and stay there while unloading the harvester without running over crops or hitting the harvester. At any moment, the harvester may speed up or slow down depending on terrain, field conditions, or crop density.

We model this nonlinear system as follows:

$$\begin{aligned}
\mathrm{x}_{t+1}(1) &= \mathrm{x}_t(1) + \Delta t(\cos(\mathrm{x}_t(3))\mathrm{x}_t(4) - \mathrm{x}_t(5) - v_c),\\
\mathrm{x}_{t+1}(2) &= \mathrm{x}_t(2) + \Delta t \sin(\mathrm{x}_t(3))\mathrm{x}_4(t),\\
\mathrm{x}_{t+1}(3) &= \mathrm{x}_t(3) + \Delta t u_t(1),\\
\mathrm{x}_{t+1}(4) &= \mathrm{x}_t(4) + \Delta t u_t(2),\\
\mathrm{x}_{t+1}(5) &= \mathrm{x}_t(5) + \Delta t d_t,
\end{aligned} \tag{31}$$

where $\mathrm{x}_t(1)$ is the horizontal distance from the tractor to the harvester, $\mathrm{x}_t(2)$ is the vertical distance from the tractor to the harvester, $\mathrm{x}_t(3)$ is the angle of the tractor, $\mathrm{x}_t(4)$ is the speed of the tractor, $\mathrm{x}_t(5)$ is the portion of the harvester's velocity set by the disturbance, and v_c is a constant velocity of the harvester. The control $u_t(1)$ is the angular velocity of the tractor and $u_t(2)$ is the acceleration of the tractor. Disturbance d_t is the acceleration of the harvester. g is defined as:

$$g(\mathrm{x}_t) = \bar{r}_1 - \|[\mathrm{x}_t(1); \mathrm{x}_t(2)] - [0, \bar{o}_c]\|_2, \tag{32}$$

where $\bar{o}_c$ is the location of the target at -2, and $\bar{r}_1$ is the target radius of 1.5.

Avoid includes the field of crops yet to be harvested, the harvester, and the crops in front of the harvester. It is characterized by: $l_1(\mathrm{x}_t), l_2(\mathrm{x}_t), l_3(\mathrm{x}_t) < 0$, where

$$\begin{aligned}
l_1(\mathrm{x}_t) &= -\mathrm{x}_t(2) - \bar{r}_2 + 2\bar{r}_3,\\
l_2(\mathrm{x}_t) &= \min(|\mathrm{x}_t(1) - 50| - 50, |\mathrm{x}_t(2)| - (\bar{r}_3 + \bar{r}_2)),\\
l_3(\mathrm{x}_t) &= \|[\mathrm{x}_t(1); \mathrm{x}_t(2)]\|_2 - (\bar{r}_2 + \bar{r}_3),
\end{aligned} \tag{33}$$

where $\bar{r}_2$ is the radius of the tractor and $\bar{r}_3$ is the radius of the harvester. These can be combined into a single constraint inequality: $l(\mathrm{x}_t) = \min\{l_1(\mathrm{x}_t), l_2(\mathrm{x}_t), l_3(\mathrm{x}_t)\} > 0$.

3) Two-Step Deep Reinforcement Learning Training: When implementing our method with DDPG for the VTOL taxi and harvester examples, we found the following general principles helpful for achieving a well-trained policy:

- Low Critic Loss: A well-trained policy typically exhibits a critic loss ranging from 0.01 to 0.001, with lower values indicating greater accuracy. Critic loss must converge before effective policy training can occur.

Hyperparameter	VTOL	Harvester
Control Layers (Actor, Critic)	512x4, 768x4	512x3, 512x4
Batch Size (H, V)	512, 512	512, 256
Epochs	100	100
Environment Steps Per Epoch	40,000	40,000
Training Time (H / V) [min]	123.15 / 162.93	135.88 / 114.27

TABLE I

TRAINING HYPERPARAMETERS

Method	VTOL	Harvester
Our RAS Policy RAS Task Success Rate	93.0	99.3
Baseline RA Policy RAS Task Success Rate	0	73.5
Our Identified $\mathcal{RAS}$ Accuracy	95.8	99.4

TABLE II

OUR RAS FRAMEWORK ACHIEVES HIGHER SUCCESS RATES THAN THE BASELINE RA METHOD, AND PROVIDES AN ACCURATE IDENTIFIED $\mathcal{RAS}$.

- Training Duration: We found it advantageous to continue training even after the loss and reward values appear stable. This additional training reduces the likelihood of outliers or edge cases during evaluation.
- Importance of training H: A well trained H value function is critical for training V. Any training error from H will be amplified in V. Generally, when the level-zero set of H is completely inside the target set, it can be used for training V.
- Episode Lengths: H trains best with episodes twice as long as those for V. At minimum, both H and V should have sufficiently long episodes to allow the agent to move from any state to the target set.

During offline training, we used one NVIDIA RTX A6000, and a AMD Ryzen Threadripper PRO 5975WX CPU, along with 256 GB of RAM. The training time of H and V depends on the batch and layer sizes, but were under three hours for both experiments. Exact values for the final tuned VTOL and Harvester models can be found in Table V-B.3. Once trained, the RAS framework is more than capable of running in real time. The final VTOL RAS policy is able to execute at 963.4 Hz, and the Harvester RAS policy executes at 1,366.1 Hz on the same hardware used for training. Hyperparameter settings for training the VTOL and Harvester environments are provided in Table V-B.3.

4) Justification of the Two Examples for Demonstrating the RAS task: The VTOL and Harvester environments have large differences between T and $\mathcal{AS}$ enabling a clear demonstration of how RA policies fail the RAS task. In VTOL, the high maximum velocity compared to low maximum acceleration results in many states being unable to slow within target. In Harvester, the tractor must match the speed and direction of the Harvester, limiting the number of directions from which it can enter T and stay.

C. Claim Validation

*1) **Demonstration of Claim 2**:* To validate that our identified $\mathcal{RAS}$ was accurate, we used the policy and adversarial disturbances from our framework to generate 2,000

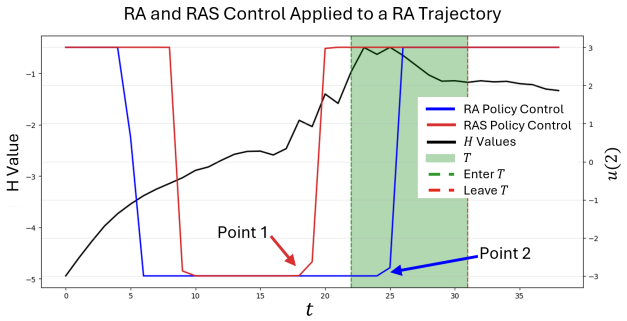

Fig. 4. The blue and red lines represent the acceleration control outputs of the RA and RAS policies along the trajectory generated by the RA policy used in Fig. 5 for the Harvester example.The black line represents H values across the trajectory. The H values are negative, indicating that RA has not entered the $\mathcal{AS}$ and is therefore unable to stay. Additionally, the RAS policy indicates that the system should start to slow at point 1 when the tractor is outside T, but the RA policy waits until point 2 when the system is in the middle of T.

trajectories, each 500 time steps long. Half of the trajectories were sampled from within our framework's identified $\mathcal{RAS}$, and the others were from outside the identified $\mathcal{RAS}$. A trajectory was considered to have performed the RAS task if it safely entered T and remained there without leaving even momentarily. We measure accuracy as the fraction of states for which our identified $\mathcal{RAS}$ correctly predicts the ability to safely reach and stay within the target set.

Our framework was highly successful at correctly identifying $\mathcal{RAS}$, the results of which can be found in the last row of Table II. Our framework achieved 95.8% accuracy in VTOL and 99.4% accuracy in Harvester.

*2) **Demonstration of Claim 3**:* To compare our RAS framework with a baseline RA method [15], we sampled 1,000 initial states from within the identified $\mathcal{RAS}$. We then used each state as a starting point for the RA and RAS policies. We defined the success rate as the percentage of trajectories that successfully performed the RAS task. Our framework achieved a 93% success rate in VTOL and 99.3% success rate in Harvester compared to the baseline RA method's success rates of 0% and 73.5% in VTOL and Harvester, as shown in the second and third rows of Table II.[1]

D. Intuitive Explanation of RA Failure

The RA policy fails at the RAS task because it reaches T, but not $\mathcal{AS}$. An example of this is shown in Fig. 5. Although the tractor does pass through T, the plot of H values for this trajectory in Fig. 4 never reaches zero, confirming that it never enters $\mathcal{AS}$. This is because the RA policy does not start to slow the tractor until it has reached Point 2 in Fig. 4 as shown by the blue line representing acceleration control from the RA policy.

To examine whether the system can slow earlier to reach $\mathcal{AS}$ and remain within T, we plot the acceleration control signal from the RAS policy along the same trajectory driven by the RA policy (shown in red in Fig. 4). We refer to this as

[1]The RA policy achieved high success rates for safely reaching T of 98.9%, and 100% across the VTOL, and harvester experiments.

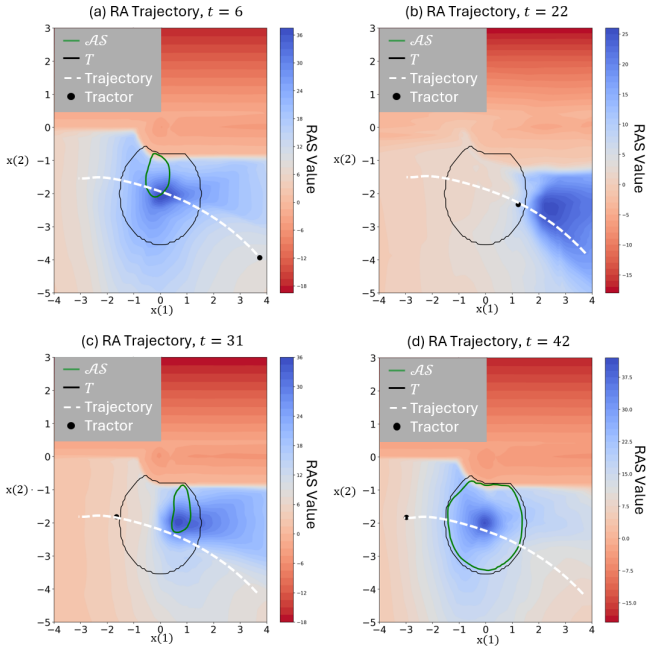

Fig. 5. A Sequence of states for a Harvester example trajectory generated by the RA policy at t = 9, 22, 31, and 42. The black outline represents T. The green outline represents the $\mathcal{AS}$. The trajectory never enters $\mathcal{AS}$, and exits T.

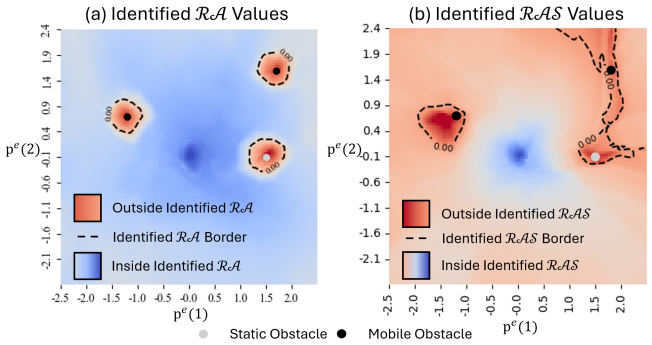

Fig. 6. The identified $\mathcal{RAS}$ and $\mathcal{RA}$ for the VTOL example. The black circles represent the mobile obstacles, and the gray circle represents the static obstacle. The identified $\mathcal{RA}$ in (a) is a super-set of the identified $\mathcal{RAS}$ in (b) due to the limits the RAS policy places on the system to enable it to stay.

the RAS policy control. This indicates that the tractor should begin to slow before entering T at Point 1 to reach $\mathcal{AS}$.

The control difference between the RA and RAS policies is a result of the RA policy using T as its target and the π_V using $\mathcal{AS}$ as its target. $\mathcal{AS}$ has speed bounds which inform π_V what speeds it must reach within T for π_H to be able to stay. However, T has no such restrictions, leading the RA policy to reach T as quickly as possible, at maximum speed. For static targets, unless the dynamics of a specific system allow it to reverse direction from maximum speed within half of T, the RA policy will miss $\mathcal{AS}$ and leave T.

E. $\mathcal{RA}$ and $\mathcal{RAS}$ Set Differences:

In theory, with identical target and constraint sets, the RA set will be a super set of the RAS set. In practice, even with learning errors, our results largely confirm this. The identified $\mathcal{RA}$ and $\mathcal{RAS}$ for the VTOL environment can be seen in

Fig. 6. It can be observed that the RAS policy is generally more cautious around both the static and mobile obstacles.

VI. Conclusion

We propose a two-step deep reinforcement learning method to solve the Reach-Avoid-Stay (RAS) problem. Our framework addresses the limitations of multiple prior methods including the difficulty of designing a CLBF [16], and the funnel method [18] assumption that the system can instantly stop. We demonstrate that without training error, our approach yields the maximal robust RAS set, which is a super-set of prior method's RAS sets. Even when facing training error, we identify the maximal robust RAS set with over 95% accuracy. We demonstrate a higher success rate than a reach-avoid baseline [13] in the RAS task, and explain how fundamental differences between the RA policy and RAS policy leads to the RA policy being unable to perform the RAS task.

The primary limitation of our framework lies in its reliance on prior knowledge of the environment, including target sets, constraint sets, and system dynamics. When this information does not accurately reflect real-world conditions, the trained policies cannot guarantee successful completion of the RAS task. This challenge has recently been addressed in safety applications by approaches such as reinforcement learning that introduces additional parameters to capture unknown environment information [29], or by integrating safety analysis with online control methods [30]. However, these approaches still lack rigorous theoretical analysis of safety and performance assurances. In future work, we aim to close this gap by extending reachability-based RL and incorporating real-world sensor data into our framework.

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)*. IEEE, 2017, pp. 2242–2253.

[2] M. Prandini and J. Hu, "Application of reachability analysis for stochastic hybrid systems to aircraft conflict prediction," in *2008 47th IEEE conference on decision and control*. IEEE, 2008, pp. 4036–4041.

[3] S. Bansal, A. Bajcsy, E. Ratner, A. D. Dragan, and C. J. Tomlin, "A Hamilton-Jacobi reachability-based framework for predicting and analyzing human motion for safe planning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7149–7155.

[4] S. Jauhri, J. Peters, and G. Chalvatzaki, "Robot learning of mobile manipulation with reachability behavior priors," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8399–8406, 2022.

[5] 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)*. IEEE, 2021, pp. 1817–1824.

[6] R. E. Allen, A. A. Clark, J. A. Starek, and M. Pavone, "A machine learning approach for real-time reachability analysis," in *2014 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2014, pp. 2202–2208.

[7] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe nonlinear control using robust neural lyapunov-barrier functions," in *Conference on Robot Learning*. PMLR, 2022, pp. 1724–1735.

[8] K.-C. Hsu, V. Rubies-Royo, C. J. Tomlin, and J. F. Fisac, "Safety and liveness guarantees through reach-avoid reinforcement learning," *arXiv preprint arXiv:2112.12288*, 2021.

[9] K.-C. Hsu, H. Hu, and J. F. Fisac, "The safety filter: A unified view of safety-critical control in autonomous systems," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, 2023.

[10] K.-C. Hsu, A. Z. Ren, D. P. Nguyen, A. Majumdar, and J. F. Fisac, "Sim-to-lab-to-real: safe reinforcement learning with shielding and generalization guarantees (abstract reprint)," in *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, 2024, pp. 22 699–22 699.

[11] K.-C. Hsu, D. P. Nguyen, and J. F. Fisac, "Isaacs: Iterative soft adversarial actor-critic for safety," in *Learning for Dynamics and Control Conference*. PMLR, 2023, pp. 90–103.

[12] 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)*. IEEE, 2019, pp. 8550–8556.

[13] 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*, vol. 10, no. 4, pp. 3582–3589, 2025.

[14] S. Summers, M. Kamgarpour, J. Lygeros, and C. Tomlin, "A stochastic reach-avoid problem with random obstacles," in *Proceedings of the 14th international conference on Hybrid systems: computation and control*, 2011, pp. 251–260.

[15] J. Li, D. Lee, S. Sojoudi, and C. J. Tomlin, "Infinite-horizon reach-avoid zero-sum games via deep reinforcement learning." [Online]. Available: http://arxiv.org/abs/2203.10142

[16] Y. Meng and J. Liu, "Lyapunov-barrier characterization of robust reach–avoid–stay specifications for hybrid systems," *Nonlinear Analysis: Hybrid Systems*, vol. 49, p. 101340, 2023.

[17] 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)*. IEEE, 2024, pp. 11 532–11 539.

[18] R. Das and P. Jagtap, "Funnel-based control for reach-avoid-stay specifications," in *2024 Tenth Indian Control Conference (ICC)*. IEEE, 2024, pp. 54–59.

[19] ——, "Prescribed-time reach-avoid-stay specifications for unknown systems: A spatiotemporal tubes approach," *IEEE Control Systems Letters*, 2024.

[20] J.-P. Aubin, A. M. Bayen, and P. Saint-Pierre, *Viability theory: new directions*. Springer Science & Business Media, 2011.

[21] F. L. Lewis, D. L. Vrabie, and V. L. Syrmos, *Optimal control*, 3rd ed. Wiley.

[22] E. V. Denardo, "Contraction Mappings in the Theory Underlying Dynamic Programming," *SIAM Review*, vol. 9, no. 2, pp. 165–177, 1967, publisher: Society for Industrial and Applied Mathematics. [Online]. Available: https://www.jstor.org/stable/2027440

[23] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell, "Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4213–4220.

[24] 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.

[25] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. PMLR, 2014, pp. 387–395.

[26] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[27] C. J. C. H. Watkins and P. Dayan, "Q-learning," vol. 8, no. 3, pp. 279–292. [Online]. Available: https://doi.org/10.1007/BF00992698

[28] J.-J. E. Slotine, W. Li, and others, *Applied nonlinear control*. Prentice hall Englewood Cliffs, NJ, vol. 199, number: 1.

[29] A. Lin, S. Peng, and S. Bansal, "One filter to deploy them all: Robust safety for quadrupedal navigation in unknown environments," *arXiv preprint arXiv:2412.09989*, 2024.

[30] J. Borquez, L. Raus, Y. U. Ciftci, and S. Bansal, "Dualguard mppi: Safe and performant optimal control by combining sampling-based mpc and hamilton-jacobi reachability," *IEEE Robotics and Automation Letters*, 2025.