Safety and Liveness Guarantees through Reach-Avoid Reinforcement Learning

Kai-Chieh Hsu*§, Vicenç Rubies-Royo†§, Claire J. Tomlin†, Jaime F. Fisac*
*Department of Electrical and Computer Engineering, Princeton University, United States
{kaichieh, jfisac}@princeton.edu
†Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, United States
{vrubies, tomlin}@berkeley.edu

Abstract*—Reach-avoid optimal control problems, in which the system must reach certain goal conditions while staying clear of unacceptable failure modes, are central to safety and liveness assurance for autonomous robotic systems, but their exact solutions are intractable for complex dynamics and environments. Recent successes in the use of reinforcement learning methods to approximately solve optimal control problems with performance objectives make their application to certification problems attractive; however, the Lagrange-type objective (cumulative costs or rewards over time) used in reinforcement learning is not suitable to encode temporal logic requirements. Recent work has shown promise in extending the reinforcement learning machinery to safety-type problems, whose objective is not a sum, but a minimum (or maximum) over time. In this work, we generalize the reinforcement learning formulation to handle all optimal control problems in the reach-avoid category. We derive a time-discounted reach-avoid Bellman backup with contraction mapping properties and prove that the resulting *reach-avoid Q-learning* algorithm converges under analogous conditions to the traditional Lagrange-type problem, yielding an arbitrarily tight conservative approximation to the reach-avoid set. We further demonstrate the use of this formulation with deep reinforcement learning methods, retaining zero-violation guarantees by treating the approximate solutions as untrusted oracles in a model-predictive supervisory control framework. We evaluate our proposed framework on a range of nonlinear systems, validating the results against analytic and numerical solutions, and through Monte Carlo simulation in previously intractable problems. Our results open the door to a range of learning-based methods for safe-and-live autonomous behavior, with applications across robotics and automation. See https://github.com/SafeRoboticsLab/safety_rl for code and supplementary material.

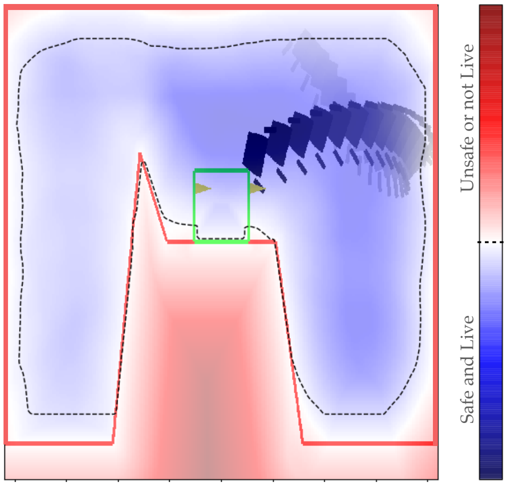


Fig. 1: Snapshots of the OpenAI Gym Lunar Lander benchmark system executing the control policy learned through Reach-Avoid Double Deep Q-Network: the vehicle avoids the failure set (terrain and screen edges) and reaches the square (green) target region. The overlaid value function slice and corresponding zero level set (dashed line) indicate the computed safe-and-live positions for the vehicle's initial attitude and velocities.

I. Introduction

Recent years have seen a significant increase in the capabilities of robotic systems, driven by rapid advances in sensing and computing hardware paired with novel optimization algorithms and data-driven methodologies. These improvements have led to an unprecedented growth in the number and variety of robots operating autonomously in the world, from autonomous airborne delivery of medical supplies [1] to the first driverless vehicle services becoming available to the public in the last year [2].

The correct functioning of the decision-making components governing the behavior of these systems is an integral part of their safety and reliability, and indeed their incorrect functioning has led to high-impact system-wide failures, including a number of recent deadly accidents involving malfunctioning advanced driver assistance systems [3, 4, 5]. Yet, while ensuring the correct functioning of robotic systems is critical to their viable deployment in high-stakes settings, these are precisely the settings in which it is often most challenging to provide strong operational assurances, because they tend to involve particularly complex, dynamic, and uncertain environments. This is not made easier by the fact that much of the recent progress on data-driven decision-making yields "black box" components whose operation after (or during) training cannot readily be certified, requiring computationally intensive *post hoc* verification [6] or supervisory control policies to restrict their outputs at runtime [7, 8]. Tractably computing decision-making policies that can enforce meaningful properties for

[§] Denotes equal contribution in alphabetical order.

autonomous robotic operation is therefore an important open problem in the field.

While the last half-decade has seen increasing interest in guaranteeing safety (preventing undesired conditions, such as collisions or severe rule violations) in robot planning and learning-based control, comparatively little attention has been paid to ensuring liveness, that is, the eventual completion of specified goals (such as delivering passengers or cargo at the desired location). Most real-world systems must meet a combination of safety and liveness properties, since enforcing one without the other tends to result in undesired behavior (a robot can typically preserve safety by indefinitely remaining at rest, or more easily achieve liveness—at least in theory—by violating safety constraints). These two properties are highly complementary, and in fact they are known to jointly encode arbitrarily sophisticated temporal logic specifications [9].

Reachability analysis is a central mathematical tool enabling the synthesis and certification of controllers with safety and liveness properties for continuous-state dynamical systems. Given a dynamical system whose evolution can be influenced through a control input, it can determine the set of initial conditions *and* the appropriate control policy for which the state of the system will be driven to some desired *target* configuration while avoiding undesired *failure* conditions. Specifically, a reach-avoid problem can be seen as the conjunction of two reachability problems: one in the positive, where the controller's goal is to reach the target set, and one in the negative, where the controller seeks to *not* reach the failure set. However, because the optimal choice of control input is coupled between the two problems, it is *not* enough to solve each of them separately and then combine the computed solutions.

Hamilton-Jacobi analysis [10] provides a general methodology to compute the optimal solution to reach-avoid problems and, thereby, robust certificates on the controlled behavior of dynamical systems. While the theory applies to general nonlinear dynamics, the central *value function* that encodes the optimal controller is the solution of a Hamilton-Jacobi partial differential equation (or in some cases a variational inequality) that, with few exceptions, cannot be obtained analytically in closed form. As a result, general Hamilton-Jacobi methods ultimately rely on state space discretization and are computationally intense, requiring computation and memory exponential in the state dimension, which limits their practical use to systems with up to about 5 continuous state variables [11].

Given the computation requirements, these synthesis methods are not commonly run online; instead, the most frequent approach is to precompute a controller (and its associated certificate) offline and store it in memory for online lookup. This approach has been successfully applied in a number of different settings, from robust trajectory tracking [12] to safe learning-based control [7].

Recent efforts have sought to use deep learning to find approximate solutions to reachability problems by directly trying to enforce satisfaction of the associated Hamilton-Jacobi partial differential equation or variational inequality [13, 14]. These methods work by repeatedly sampling random points over the state space and time horizon and computing the associated local Hamilton-Jacobi equation error. By enforcing this error to be zero, these methods effectively propagate information in backward time from the boundary condition. In this setting, a dynamical model of the system, usually control-affine, is required in order to compute the Hamiltonian and compute the error. In a similar spirit, a different line of work represents reachability controllers as neural network classifiers by directly exploiting the "bang-bang" nature of optimal reachability controllers for control-affine systems [15].

A successful recent approach for Hamilton-Jacobi *safety* analysis relies on the contraction mapping property commonly exploited in the reinforcement learning literature to deliver a converging temporal learning scheme for safety problems [16]. In this work the authors exploit said contraction mapping using several reinforcement learning frameworks, and most notably, deep Q-learning.

Contribution. Our central theoretical contribution is the derivation of a novel, time-discounted formulation of the reach-avoid optimal control problem that lends itself to reinforcement learning methods thanks to the contraction mapping induced by the associated dynamic programming equation. Unlike the previously derived safety-only analog, this discounted reach-avoid solution is guaranteed to produce a conservative under-approximation of the set of conditions from which the system can succeed at its task. We further show that the approximation becomes arbitrarily tight as the time discounting becomes less pronounced.

Importantly, our guarantees can be extended to the use of deep reinforcement learning methods by treating the approximate optimal policy as an untrusted oracle and applying a supervisory control scheme: this insight is key, because it enables us to construct control policies with guaranteed safety and liveness for high-dimensional nonlinear systems that are intractable with classical approaches. To determine the reliability of the proposed method as a synthesis tool, we present comprehensive validation results against analytic and numerical solutions when these are obtainable, and by exhaustive Monte Carlo simulation in two high-dimensional nonlinear systems for which reach-avoid solutions have, until now, not been obtainable. Fig. 1 showcases one of the high-dimensional examples we tested.

II. Background

A. Dynamical System Safety and Liveness Analysis

Consider a dynamical system with state $s \in \mathcal{S} \subset \mathbb{R}^n$ and control input $u \in \mathcal{U} \subset \mathbb{R}^m$, which evolves according to the ordinary differential equation

$$\dot{s} = f(s, u) \,. \tag{1}$$

We assume the set $\mathcal{U}$ to be compact and the dynamics f to be bounded and Lipschitz continuous; under these conditions, system trajectories $\xi : \mathbb{R}_+ \to \mathcal{S}$ are well defined, and continuous in time, for all measurable control inputs [17, Ch. 2]. We use the notation $\xi_s^{\mathbf{u}}(\cdot)$ for a time trajectory starting at state s under control signal $\mathbf{u}$. For discrete-time approximations, we denote the time step $\Delta t > 0$.

Safety and liveness specifications for dynamical systems can be succinctly formulated in terms of a *target set* $\mathcal{T} \subset \mathbb{R}^n$, the collection of states where we want to steer trajectories, and a *constraint set* $\mathcal{K} \subset \mathbb{R}^n$, the set of allowable states during the evolution of a trajectory. The complement of the constraint set is the *failure set* which we denote as $\mathcal{F} = \mathcal{K}^c$. By convention, we let $\mathcal{T}$ and $\mathcal{K}$ be closed sets; no other assumptions (convexity, connectedness, etc.) are made.

The *safe set* with respect to the failure set $\mathcal{F}$ (or equivalently the *viability kernel* of the constraint set $\mathcal{K}$), is defined as the collection of initial states from which the controller can indefinitely keep the system away from the failure set:

$$\Omega(\mathcal{F}) \;:=\; \{s \in \mathcal{S} \mid \exists \mathbf{u} \in \mathbb{U}, \forall \tau \geq 0, \xi_s^{\mathbf{u}}(\tau) \notin \mathcal{F}\}, \qquad (2)$$

where $\mathbb{U}$ represents the collection of all measurable control signals $\mathbf{u} : \mathbb{R}_+ \to \mathcal{U}$. Conversely, the *backward-reachable set* (or, more succinctly, the *reach set*) of the target $\mathcal{T}$ is

$$\mathcal{R}(\mathcal{T}) \;:=\; \{s \in \mathcal{S} \mid \exists \mathbf{u} \in \mathbb{U}, \exists \tau \geq 0, \xi_s^{\mathbf{u}}(\tau) \in \mathcal{T}\}. \qquad (3)$$

In many practical problems in robotics our goal is a combination of the above: namely, we want the system to reach a target configuration while maintaining safety. This requirement brings forth the notion of the *reach-avoid set*:

$$\begin{aligned}\mathcal{RA}(\mathcal{T};\mathcal{F}) \;:=\; \{&s \in \mathcal{S} \mid \exists \mathbf{u} \in \mathbb{U}, \tau \geq 0,\\ &\xi_s^{\mathbf{u}}(\tau) \in \mathcal{T} \wedge \forall \kappa \in [0,\tau]\; \xi_s^{\mathbf{u}}(\kappa) \notin \mathcal{F}\}, \qquad (4)\end{aligned}$$

which denotes the set of states for which some control signal exists that can drive the system to $\mathcal{T}$ while avoiding $\mathcal{F}$ at all prior times. Note that this set is in general *not* equal to the intersection of the previous two sets, because (a) there may exist states in $\mathcal{R}(\mathcal{T}) \cap \Omega(\mathcal{F})$ from which the controller can *either* drive the system to $\mathcal{T}$ *or* keep it away from $\mathcal{F}$, but not both (in other words, the control signals $\mathbf{u}$ exist in both cases but are mutually incompatible), and (b) there may exist states $s \notin \Omega(\mathcal{F})$ from which the system cannot be indefinitely kept away from $\mathcal{F}$, but it can *first* be driven to $\mathcal{T}$. The latter case is important for encoding complex operational requirements involving preconditions: for example, we may wish to prevent an autonomous vehicle from entering an intersection *before* reaching a state from which it is visible to any potential conflicting traffic.

B. Reach-Avoid Analysis via Dynamic Programming

In order to compute the reach-avoid set it is helpful to define two *implicit surface functions* which are used to delineate sets of states. For our problem of interest we will define two such functions $l(\cdot), g(\cdot) : \mathcal{S} \to \mathbb{R}$, which are Lipschitz-continuous and satisfy the following properties:

$$l(s) \leq 0 \iff s \in \mathcal{T} \qquad (5)$$

$$g(s) > 0 \iff s \in \mathcal{F}\ . \qquad (6)$$

Using these implicit surface function definitions, let us define the following payoff functional for trajectories of the system in *discrete-time*[1]:

$$\mathcal{V}^{\mathbf{u}}(s) = \min_{\tau\in\{0,1,\dots\}} \max\Big\{ l(\xi_s^{\mathbf{u}}(\tau)), \max_{\kappa\in\{0,\dots,\tau\}} g(\xi_s^{\mathbf{u}}(\kappa))\Big\}. \qquad (7)$$

The outer maximum in (7) acts as an "overwriting" mechanism. Given definition (6), if a trajectory remains inside $\mathcal{K}$ for all time, the inner maximum will be negative, and $\mathcal{V}^{\mathbf{u}}(s)$ will be negative if and only if the trajectory reaches $\mathcal{T}$. In contrast, if the constraints are ever violated the inner maximum will be positive, which ensures that the overall payoff must also be positive. In other words, the payoff can only be negative if $\mathcal{T}$ is reached without previously violating the constraints.

The goal from any initial state will be to minimize the payoff functional (7), which leads to the definition of the *value function* over the entire state space,

$$V(s) = \inf_{\mathbf{u}\in\mathbb{U}} \mathcal{V}^{\mathbf{u}}(s). \qquad (8)$$

The *sign* of the value function encodes crucial information on the safety and liveness of our problem. Concretely,

$$V(s) \leq 0 \iff s \in \mathcal{RA}(\mathcal{T};\mathcal{F}). \qquad (9)$$

It can be shown that the value function must satisfy the fixed-point *reach-avoid Bellman equation* (RABE) below, where $s_+^u = \xi_s^u(\Delta t)$. For details on the derivation we refer the reader to Appendix A.

$$V(s) = \max\Big\{ g(s), \min\big\{ l(s), \inf_{u\in\mathcal{U}} V(s_+^u) \big\} \Big\}. \qquad (10)$$

Running Example (2-D point particle)**.** *Consider a point particle with simple motion à la Isaacs [19] that is required to reach a (yellow) target box without crossing any of three (magenta) obstacle boxes or leaving its rectangular domain, as shown in Fig. 2. The dynamics of the particle are described by the equation*

$$\dot{x} = uv_x, \;\; \dot{y} = v_y, \qquad (11)$$

where $v_x, v_y > 0$ are horizontal and vertical speed constants and u is the control input. The state is the particle's position, $s = [x, y]^T$ and the allowed control is discrete, $u \in \{-1, 0, 1\}$. The dynamics in (11) describe a particle that moves continually upward at a rate v_y while only being able to control its horizontal speed through u.

Here each box (obstacles, boundary and target) is specified by its center position and dimensions, i.e., (p, L), with $p = [p_x, p_y]^T$ and $L = [L_x, L_y]^T$. The safety margin function is expressed as the point-wise maximum of signed

[1] We follow the discrete-time formulation consistently with the standard reinforcement learning framework; we direct readers to [18] for the continuous-time analog.

distance functions. Thus, $g(s)$ is computed by the margin to the boundary ($g_b(s)$) and the obstacles ($g_o(s)$) as below

$$
\begin{aligned}
g_b(s) &:= \max_i \Delta(s, p^{c_b})[i] - L^{c_b}[i]/2, \\
g_o(s) &:= \max_j \max_i L^{c_j}[i]/2 - \Delta(s, p^{c_j})[i], \\
g(s) &:= \max\left\{g_b(s),\ g_o(s)\right\},
\end{aligned}
$$

where $\Delta(s,p) := [|x - p_x|,\ |y - p_y|]^T$. The target margin function can be expressed as

$$l(s) := \max_i \Delta(s, p^t)[i] - L^t[i]/2.$$

C. Q-Learning and Approximate Dynamic Programming

Reinforcement learning, also known as direct adaptive optimal control [20], provides a framework for solving dynamic decision problems represented as Markov Decision Processes (MDPs) where the underlying objective is to maximize a Lagrange-type functional

$$\mathcal{V}^{\mathbf{u}}(s) = \sum_{\tau} \gamma^{\tau} r(s_{\tau}, u_{\tau}), \tag{12}$$

where $r(s,u) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is known as the *reward function* and $\gamma \in [0,1)$ is the time-discount parameter. Tabular Q-learning, or asynchronous dynamic programming [21], is a type of model-free reinforcement learning scheme which does not require knowledge of the dynamics of the system. It maintains a state-action value matrix, known as the Q-values, and updates each entry in the following manner: $Q(s,u) \xleftarrow{\alpha} r(s,u) + \gamma \max_{u'} Q(s_+^u, u')$, where the operator $\xleftarrow{\alpha}$ implies $x \xleftarrow{\alpha} y \equiv x \leftarrow x + \alpha(y - x)$. Under some mild conditions [22], tabular Q-learning converges to the optimal solution, meaning that the optimal policy and state value function are given by $u^* := \arg\max_u Q(s,u)$ and $V(s) = Q(s,u^*)$. Unfortunately, the discretization of the state space in tabular Q-learning leads to an exponential growth in storage requirements, an issue which is commonly referred to as the *curse of dimensionality*.

Deep reinforcement learning shows potential to alleviate the curse of dimensionality in optimal control problems by using deep neural networks to find approximately optimal value functions and policies [23, 24, 25]. In particular, double deep Q-Network (DDQN) [23] uses two deep neural networks (NNs) to approximate the Q-values. One NN is called the online network with parameter w and the other network is called the target network with parameter $\tilde{w}$. The Q-value is updated by $Q_w(s,u) \xleftarrow{\alpha} r(s,u) + \gamma Q_{\tilde{w}}(s_+^u, \max_{u'} Q_w(s_+^u, u'))$.

In [16], the authors show how to use reinforcement learning to approximate the solution to the Hamilton-Jacobi-Bellman (HJB) equation for the special case where the cost functional is given in the form $\mathcal{V}^{\mathbf{u}} = \min_{\tau \in \{0,1,\ldots\}}\ g(\xi_s^{\mathbf{u}})$. In particular, they use the following Q-learning update

$$
\begin{aligned}
Q_w(s,u) \leftarrow \gamma \min\left\{g(s),\ Q_{\tilde{w}}(s_+^u,\ \max_{u'} Q_w(s_+^u, u'))\right\} \\
+(1-\gamma)\ g(s),
\end{aligned} \tag{13}
$$

which can be used to approximate solutions for safety or liveness problems, but not both. In the next section we introduce a more general update for the broader class of reach-avoid problems.

III. The Time-Discounted Reach-Avoid Bellman Equation

Unlike the Lagrange-type dynamic programming equation commonly used in reinforcement learning, the reach-avoid optimality condition (10) does not induce a contraction mapping on V. This contraction mapping property is a crucial requirement for the convergence of the value iteration scheme and related temporal learning approaches, such as Q-learning [21, 22]. In what follows we provide a principled modification of (10) that induces a contraction mapping in the space of value functions through the introduction of a time discount parameter $\gamma \in [0,1)$. This enables the use of reinforcement learning methods based on temporal learning, in the same spirit as [16]. In addition, we will see that for any choice of γ, the fixed-point solution $V_\gamma(x)$ and its corresponding reach-avoid set will be an under-approximation of the true (undiscounted) reach-avoid set $\mathcal{RA}_\gamma \subseteq \mathcal{RA}$, which (under mild technical assumptions) becomes arbitrarily tight as $\gamma \to 1$.

A. Probabilistic Interpretation of Time Discounting

As previously stated, (10) is not guaranteed to induce a contraction in the space of value functions. This can be addressed by introducing a discount factor $\gamma \in [0,1)$, which can be interpreted as a probability of episode continuation. In "traditional" (Lagrange-type) reinforcement learning, for example, the value of a state can be written using this interpretation as follows

$$V(s) = \max_{u \in \mathcal{U}}\ (1-\gamma) r(s,u) + \gamma\big(r(s,u) + V\big(s_+^u\big)\big), \tag{14}$$

with $s_+^u = s + f(s,u)\Delta t$. The term $1 - \gamma$ can be seen as representing the probability of the episode terminating in one step, therefore only accounting for the immediate reward, with γ conversely representing the probability of the episode continuing, hence accounting for the immediate reward and the expectation of future returns, as encoded in the value function. Maintaining this interpretation, and noting that truncating (7) to a single-step problem leaves $\mathcal{V}^{\mathbf{u}}(s) = \max\{g(s), l(s)\}, \forall \mathbf{u}$, we can modify (10) with an analogous "probabilistic time discounting":

$$
\begin{aligned}
B_\gamma[V](s) &:= \gamma \max\Big\{\min\big\{\min_{u \in \mathcal{U}} V(s_+^u),\ l(s)\big\},\ g(s)\Big\} + \\
&\qquad (1-\gamma)\max\{l(s), g(s)\}, \\
V_\gamma(s) &= B_\gamma[V_\gamma](s).
\end{aligned} \tag{15}
$$

We denote (15) the *discounted reach-avoid Bellman equation* (DRABE). Unlike its undiscounted counterpart (10), this fixed-point equation does induce a contraction mapping in the space of value functions. We show this next.

B. Contraction Mapping and Reach-Avoid Q-learning

Based on the time-discounted value function defined in (15), we can define the γ-discounted reach-avoid set, by analogy with (9), as $\mathcal{RA}_\gamma := \{s \in \mathcal{S} | V_\gamma(s) \leq 0\}$. We

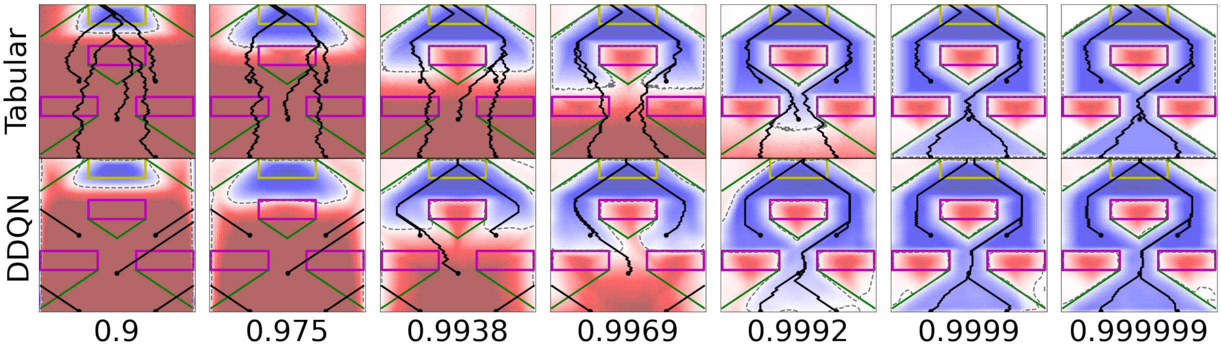


Fig. 2: A convergent family of under-approximations that asymptotically approaches the undiscounted reach-avoid set as $\gamma \to 1$. The red region indicates positive state value, while blue region indicates negative state value. The dashed gray line specifies the zero level set or the discounted reach-avoid set boundary, while Green lines specify analytic reach-avoid set boundary. The solid black lines show trajectory rollouts from five initial states.

present three propositions for the proposed DRABE and leave their proofs in Appendix A. We first show that DRABE induces a contraction mapping. With the important contraction mapping property at hand, Q-learning can be guaranteed to converge if the learning rate satisfies the assumptions in [22]. In practice, this contraction mapping property also enables the use of approximate ("deep") Q-learning methods to be directly applied, as demonstrated in Section IV, even though these methods lack the theoretical convergence guarantees of their exact (tabular) counterpart. Finally, we show that the fixed-point solution of DRABE converges to the fixed-point solution of RABE when γ approaches to 1.

Proposition 1 (Contraction Mapping)**.** *The discounted reach-avoid Bellman equation* (15) *induces a contraction mapping under the supremum norm for any* $\gamma \in [0, 1)$*.*

Proposition 2 (Convergence of Reach-Avoid Q-Learning)**.** *Let* $\mathbf{S} \subseteq \mathcal{S}$ *and* $\mathbf{U} \subseteq \mathcal{U}$ *be finite discretizations of the state and action spaces, and let* $\mathbf{f} : \mathbf{S} \times \mathbf{U} \to \mathbf{S}$ *be a discrete transition function approximating the system dynamics. The Q-Learning scheme, applied to the Discounted Reach-Avoid Bellman Equation* (15) *and executed on the above discretization converges, with probability* 1*, to the optimal state-action safety value function*

$$Q(\mathbf{s}, \mathbf{u}) := (1-\gamma)\max\{l(\mathbf{s}), g(\mathbf{s})\} + \gamma \max\Big\{g(\mathbf{s}),\ \min\Big\{l(\mathbf{s}), \max_{\mathbf{u}' \in \mathbf{U}} Q\big(\mathbf{f}(\mathbf{s}, \mathbf{u}), \mathbf{u}'\big)\Big\}\Big\}\ ,$$

in the limit of infinite exploration time and given partly-random episode initialization and learning policy with full support over $\mathbf{S}$ *and* $\mathbf{U}$ *respectively. Concretely, the temporal difference learning rule is:*

$$Q_{k+1}(\mathbf{s}, \mathbf{u}) \leftarrow Q_k(\mathbf{s}, \mathbf{u}) + \alpha_k \Big[(1-\gamma)\max\{l(s), g(s)\} + \gamma \max\Big\{g(s), \min\big\{ \max_{\mathbf{u}' \in \mathbf{U}} Q\big(\mathbf{f}(\mathbf{s}, \mathbf{u}), \mathbf{u}'\big),\ l(s)\big\}\Big\} - Q_k(\mathbf{s}, \mathbf{u})\Big],$$

for learning rate $\alpha_k(\mathbf{s}, \mathbf{u})$ *satisfying*

$$\sum_k \alpha_k(\mathbf{s}, \mathbf{u}) = \infty \quad \sum_k \alpha_k^2(\mathbf{s}, \mathbf{u}) < \infty\ ,$$

for all $\mathbf{s} \in \mathbf{S}, \mathbf{u} \in \mathbf{U}$.

Proposition 3 (Convergent Value Function)**.** *In the limit of* $\gamma \to 1$*, the fixed point of DRABE* (15) *converges to the fixed point of RABE* (10) *and is the solution of* (8)*.*

C. Discounted Reach-Avoid Sets

In Section III-B, we have shown that Q-learning methods converge to a fixed point. Another crucial property is the connection between the discounted ($0 \leq \gamma < 1$) and the exact ($\gamma = 1$) reach-avoid set. The following result shows that the discounted reach-avoid set defined by the discounted reach-avoid value function is always an under-approximation and asymptotically approaches the true reach-avoid set as γ approaches 1.

Theorem 1 (Conservativeness of Discounted Reach-Avoid Set)**.** *For any discount factors* $0 \leq \gamma_1 \leq \gamma_2 < 1$*, the reach-avoid set associated with* γ_1 *is a subset of the reach-avoid set associated with* γ_2*. Moreover, as long as* $\mathcal{RA}$ *has a nonempty interior and* V *is nowhere locally constant on its zero level set,* $\mathcal{RA}_\gamma$ *asymptotically approaches* $\mathcal{RA}$*, under the Hausdorff set distance, as* $\gamma \to 1$*.*

The proof can be found in Appendix A. Theorem 1 states that any increasing sequence of discount factors approaching 1 has an associated sequence of reach avoid-sets where each set contains all of its predecessors. In addition, as γ tends to 1, the sequence of discounted reach-avoid sets approaches the true reach-avoid set. These features are particularly appealing for *safe learning* since any discount factor will yield a conservative but safe under-approximation of the reach-avoid set. In other words, the value function should never wrongly predict that it can safely reach the target when in fact it collides. This property makes the value function for any γ a good candidate for *shielding* methods [7, 26], where the agent is allowed to explore the environment until safety or liveness are at stake. Thus, any discounted reach-avoid set yields a permissible region of exploration and an associated emergency controller through the value function.

Running Example (2-D point particle)**.** *Fig. 2 demonstrates the convergent family of under-approximate reach-avoid sets obtained by different Q-Learning methods, which are tabular Q-learning (TQ) [21] and double deep Q-network (DDQN) [23]* [2]*. First, we look at the discounted reach-avoid sets obtained by TQ as shown in the first row of Fig. 2. We observe the boundary of the discounted reach-avoid set is very close to the target set initially. However, as γ gradually annealing to 1, it approaches the undiscounted reach-avoid set. More importantly, the discounted reach-avoid sets are always under-approximations, which means the proposed method guarantees safety and liveness more conservatively. Secondly, we compare the reach-avoid sets by DDQN with TQ. Similarly, the discounted reach-avoid sets are generally under-approximations with some errors near the boundary. This result suggests that deep RL methods manage to approximate reach-avoid games' value function as long as we slowly anneal discount rate to 1.*

IV. Results and Validation

In this section we present the results of implementing our proposed Discounted Reach-Avoid Bellman Equation on a 2-D point particle problem, 3-D Dubins car problem, a 6-D Lunar Lander problem and a 6-D attack-defense game. We use DDQN to learn the approximate value function and we describe the architecture, training hyper-parameters and initialization of the NN in Appendix B.

A. 2-D point particle: Sum of Costs versus Reach-Avoid

This first problem employs the same dynamic system in our running example. We compare reinforcement learning directly optimizing the discounted reach-avoid cost (RA) against reinforcement learning minimizing the usual sum of discounted costs (Sum). We consider a sparse cost for the usual reinforcement learning, i.e., $c(s,u) = -1$ if $s_+^u \in \mathcal{T}$, $c(s,u) = \rho$ if $s_+^u \in \mathcal{F}$, and zero elsewhere. Here, we fix γ to 0.95 in Sum training and 0.9999 in RA training. Rollouts end when the agent hits the outer square boundary. This termination condition we denote as "End".

Fig. 3 shows the learned state value function as well as the training progress. The state value function shows that RA conforms with the safety and liveness specification. Thus, the state value is the same if, from that state, the control policy can navigate to the center of the target set. In addition, the trajectory rollouts show that the policy keeps the distance from the failure set since that distance influences the outcome. On the other hand, Sum can have a decent reach-avoid set only if the penalty is adequately specified ($\rho = 1$). Even if the state value is similar to RA's, the trajectory rollouts do often come close to the failure set, which is not desirable. The training progress suggests that RA has a better (asymptotic) success ratio than Sum with an adequate penalty, and RA converges much faster than Sum with a low penalty ($\rho = 0.1$). RA provides an easy way to formulate the cost because it only requires l and g having the property in (5) and (6), while Sum requires careful tuning of the cost structure.

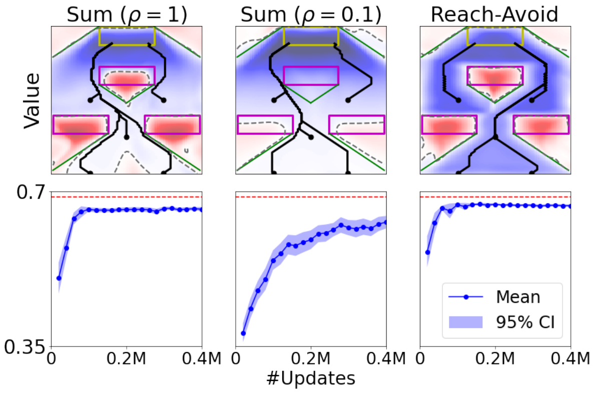


Fig. 3: First row: Value function after 400, 000 gradient updates. Second row: Fraction of episodes reaching the target set without visiting the failure set as training proceeds. Each run is an average over 1281 episodes with initial states from the discretized state space. The statistics is computed from 50 independent runs. The dashed red line specifies the success ratio of the chosen validation states based on the analytic reach-avoid set.

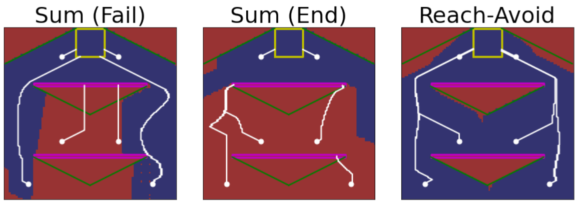


Fig. 4: Binary reach-avoid outcome from executing the learned policies, for different objective formulations, after 400, 000 gradient updates. White lines show trajectory rollouts from six initial states. The red region indicates failed rollouts from the corresponding position states, while blue region indicates rollouts that reach the target. The first two figures correspond to reinforcement learning formulations with different rollout termination conditions. The last figure corresponds to our reach-avoid formulation.

To show that an appropriate cost structure is sometimes challenging to find, we construct a new environment with two thin obstacles and one target while keeping the system dynamics the same. Fig. 4 shows the binary reach-avoid outcomes of trajectory rollouts. The rightmost figure shows that RA has the correct rollout outcomes except for some regions around the top reach-avoid set boundary. However, Sum with "End" termination only has the correct rollout outcomes for the region near the target. In order to enforce safety to a larger degree in the Sum formulation, we terminate the training episode if the agent collides with the obstacles. We denote this new rollout termination condition as "Fail", shown

[2] For TQ, We use a grid of 81 × 241 cells, while for DDQN, we describe the architecture and activation in Appendix B.

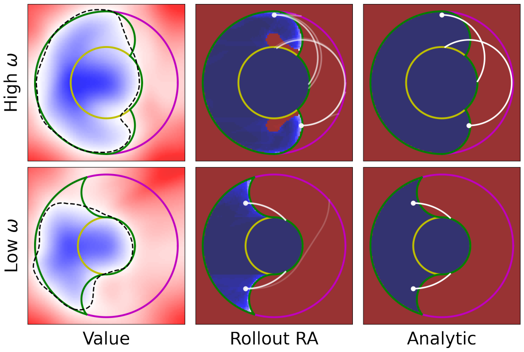


Fig. 5: Value function learned after $400,000$ gradient updates. The reach-avoid set in the second column is the average success rate over 50 models trained on different random seeds. White lines show trajectory rollouts of first 10 models from two different initial states. The reach-avoid set in the third column is derived from geometric analysis and the trajectories are simply by constant angular velocity.

in the leftmost figure. We observe that Sum (Fail) has more correct outcomes than Sum (End), especially on the two sides of the environment. However, in the central region, blocked by two thin obstacles, the learned policies fail to evade obstacles. This result is possibly due to the abrupt value transition from the top boundary of the obstacles, so the negative cost is difficult to be propagated to the region under the obstacles. The reason we observe these issues with the Sum formulation is that these objectives are only *proxies* for the property we need our system to satisfy, whereas the reach-avoid objective is an exact encoding of it. In other words, regardless of the environment configuration, the reach-avoid policy will be optimizing for "the right thing". In this sense, the reach-avoid objective provides a more robust generalization across environments.

B. 3-D Dubins Car: Q-Learning as an Untrusted Oracle

The dynamics of the Dubins car are described by the following equation

$$\dot{x} = v\cos\theta,\ \dot{y} = v\sin\theta,\ \dot{\theta} = u, \tag{16}$$

where $u \in \{\omega, 0, -\omega\}$. In this environment, the state of the system is $s = [x, y, \theta]$, with x, y encoding position and θ encoding the heading. In this environment we constrain the car to lie within the outer (magenta) circle and the goal is for the car to reach the inner (yellow) circle as shown in Fig. 5. The implicit surface functions here are defined as $l(s) := \|s\| - r$ and $g(s) := \|s\| - R$, where r and R are the inner and outer radii, respectively. We construct two types of Dubins cars: (1) one with high turning rate ($r \geq 2v/\omega - R,\ v = 0.5, \omega = 0.833, R = 1, r = 0.5$) and (2) a second one with low turning rate ($r < 2v/\omega - R,\ v = 0.5, \omega = 0.667, R = 1, r = 0.4$). The results are shown in the first and the second row of Fig. 5, respectively.

The left column of Fig. 5 shows a slice of the learned value function for a 0° heading angle. We first observe that the learned value function does not accurately approximate the analytic reach-avoid value function (rightmost column of Fig. 5). We take a closer look at the misclassified states by computing the confusion matrix between the predicted value function and the obtained outcome when rolling out the learned policy. The condition is "success" ("failure") when the learned policy rollout value is negative (positive) and the predicted condition is "success" ("failure") when the DDQN value is negative (positive). We obtain 6.6% false-success rate (FSR) and 5.1% false-failure rate (FFR) for the car with a high turning rate, while we get 7.9% FSR and 2.5% FFR for the car with a low turning rate. Here we emphasize FSR because these are states from which we predict to reach the target set without entering the failure set, but the learned policy fails to deliver on this promise.

While the value function computed through deep Q-learning may not, by itself, constitute a reliable safety-liveness indicator, we can instead treat it as an *untrusted oracle* encoding a best-effort reach-avoid policy. Provided that we have the ability to simulate the dynamics of the system accurately (or with a nontrivial error bound), we can obtain an improved value prediction for any query state by rolling out the learned policy and observing its outcome. In fact, this *rollout value* has two practically useful properties: first, it is accurate (to the degree of fidelity allowed by the simulator); and second, it encodes the outcome achieved by the *actually available* best-effort policy. If this policy is substantially worse than the optimal one (due to poor learning, for example) the resulting rollout value will predict worse performance than could otherwise have been achieved, but it will nonetheless predict it *accurately* for the learned policy. Even when small inaccuracies exist in the simulator, we can generally expect the FSR to be drastically lower than for the DDQN value prediction.

We minimize DDQN-predicted state-action values to obtain the DDQN "untrusted oracle" policy, and use this policy to obtain the rollout value, hence the approximate reach-avoid set. The central column of Fig. 5 shows the average reach-avoid outcome across 50 models trained on different random seeds if we use rollout value, while the rightmost column shows the ground truth analytic reach-avoid set obtained by geometric means.

This untrusted oracle approach enables us to translate our learned reach-avoid solution into a supervisory control law with a recursive feasibility guarantee, by employing a "shielding" scheme [26]. Suppose that we start inside the rollout reach-avoid set (blue region in the central column of Fig. 5). Before we execute a candidate action from any policy (e.g. from a performance-driven reinforcement learning agent), we simulate a short trajectory forward to see if we would remain in the reach-avoid set after this action. If the simulated rollout is successful, then we are free to execute the candidate action; otherwise, we can use the reach-avoid policy to steer away from the reach-avoid set boundary, and thus maintain safety and liveness. Using the untrusted oracle policy, we get a FFR of 4.8% for high turn rate and 2.5% for low turn rate, and a FSR of exactly 0, by construction (thereby ensuring the success of shielding schemes provided we have an accurate simulator available at runtime).

C. 6-D Lunar Lander

In this section we will showcase the algorithm on the lunar lander environment from OpenAI gym [27]. In this environment the state of the system is given by the six dimensional vector $s := [x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}]$, and the lander has four available actions at its disposal: to activate the left, right or main engine, or to apply no action.

Let us define the minimum signed L_2 distance between a point $[x, y]^T \in \mathbb{R}^2$ and a generic set $\mathcal{S}$ as

$$L_2(x, y; \mathcal{S}) = c\Big(\min_{[\hat{x},\hat{y}]\in\partial\mathcal{S}} \sqrt{(x-\hat{x})^2 + (y-\hat{y})^2}\Big), \quad (17)$$

where $c = -1$ if $[x, y]^T \in \mathcal{S}$ and $c = +1$ otherwise, and $\partial\mathcal{S}$ is the boundary of the set.

The safety margin is defined by the minimum signed L_2 distance from the center of the lander to the moon surface, as well as the minimum distance to the left, right and top boundaries. Together, these four components define a polygon $\mathcal{P}_\mathcal{C} \subset \mathbb{R}^2$ which demarcates the constraint set. The *safety margin* is thus defined as $g(s) = -L_2(x, y; \mathcal{P}_\mathcal{C})$.

The target set $\mathcal{T}$ is defined as a rectangular region within the constraint set (i.e. $\mathcal{T} \subset \mathcal{P}_\mathcal{C}$) that we want to reach without colliding. The *target margin* is similarly defined as the minimum signed L_2 distance to $\mathcal{T}$, that is $l(s) = L_2(x, y; \mathcal{T})$.

In our experiments we run the environment for 5 million gradient updates. Fig. 6 shows slices of the value function for different values of v_x and v_y, while keeping θ and $\dot{\theta}$ constant. In this case the system is too high-dimensional to be able to compare the value function to a ground-truth. However, Fig. 6 shows the correct qualitative behavior of our solution, with leftward and rightward initial speeds causing the value function to extend horizontally from the boundary and the obstacle. Similarly, initial vertical speeds cause the value function to be positive near the bottom and top of the environment. For these experiments we find the FSR and FFR to be 20% and 4% respectively.

D. 6-D Attack-Defense Game with Two Dubins Cars

In this experiment, we consider attack-defense games between two identical Dubins cars. One car, *the attacker*, wants to reach the yellow target without hitting the magenta failure set boundary. On the other hand, the other car, *the defender*, attempts to catch the attacker. We consider the game is won if the attacker manages to reach their goal before they are caught by the defender. An example is shown in Fig. 7. In the following paragraphs, we will use subscript A to denote variables related to the attacker and D to the defender. The state of the system in this experiment is $s = (s_A, s_D)$ and the dynamics of the Dubins car follow (16) with $v = 0.75, \omega = 3, R = 1, r = 0.5$.

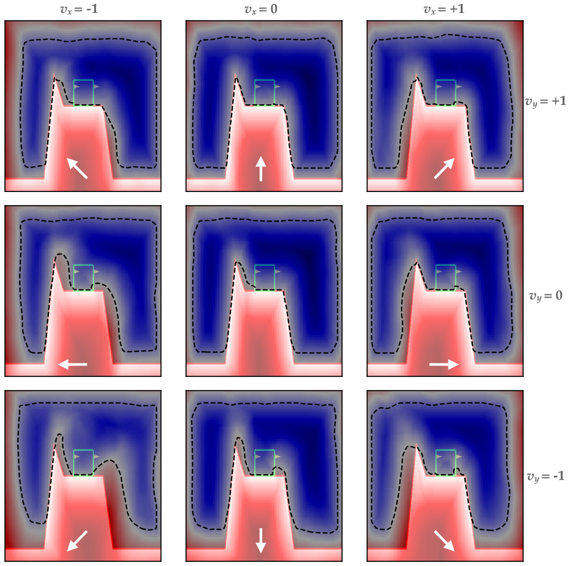
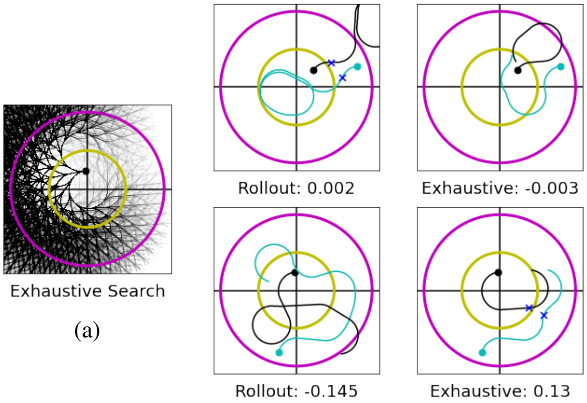


Fig. 6: Slices of the value function for different values of v_x and v_y, with $\theta = 0$ and $\dot{\theta} = 0$. Top, middle and bottom rows correspond to $v_y = 1, 0, -1$ respectively. First, second and last column represent $v_x = -1, 0, 1$ respectively. The dashed line denotes the zero level set, and the arrows denote the direction of the initial speed.



(a)

(b)

Fig. 7: Validation of the *defender's* policy. (a): exhaustive trajectories. (b): DDQN rollout and the worst case in exhaustive trajectories. First row: false failure; Second row: false success. Black lines show the defender's trajectories, while green lines show the attacker's trajectories. Blue x's specify the instant when the attacker is captured by the defender.

The target set and failure set for this experiment are defined below.

$$\mathcal{T} := \{(s_A, s_D) \mid s_A \in \mathcal{T}_A\}, \quad (18)$$

$$\mathcal{F} := \{(s_A, s_D) \mid s_A \in \mathcal{F}_A \lor (s_A, s_D) \in \mathcal{F}_J\}, \quad (19)$$

where $\mathcal{F}_J := \{(s_A, s_D) \mid \|s_A - s_D\|_2 \leq \beta\}$ and β is the capture range. We use $\beta = 0.25$ in this experiment. Their implicit surface functions are as follows.

$$\begin{aligned} l(s) &= \|s_A\|_2 - r \\ g(s) &:= \max\Big\{\|s_A\|_2 - R,\ \beta - \|s_A - s_D\|_2\Big\}. \end{aligned}$$

We learn the minimax player policies by replacing $\min_{u\in\mathcal{U}} V(s_+^u)$ in (15) with $\min_{u_A\in\mathcal{U}} \max_{u_D\in\mathcal{U}} V(s_+^{u_A,u_D})$. With this modification, the output of the DDQN becomes a matrix indexed by controls from the attacker and defender. We evaluate the minimax player policies by DDQN values in two ways: (1) *estimation error*, how well we learn from the data, and (2) *approximation error*, how closed the minimax policy is to the (conservatively) optimal policy.

For estimation error, we sample 225 attacker positions uniformly in the ring and 225 defender positions uniformly inside the constraint set. In addition, we uniformly sample 15 heading angles in $[0,\ 2\pi]$ for both attacker and defender. For these 15^6 state samples, we execute the policy by DDQN values for 100 time steps. We compare this rollout value with the DDQN predicted minimax value and got 6.7% FFR and 23.3% FSR.

For approximation error, we want to check if the defender's policy is optimal by two-round validation. In each round, we divide 50 time steps into ten intervals, and within each interval, the defender adopts the same action. Therefore, we have 3^{10} different action sequences, as shown in Fig. 7 (a). We obtain the rollout by letting the attacker follow the DDQN policy, but the defender follows the artificial action sequences. We rank exhaustive trajectories by (1) crossing the constraint or captured, (2) unfinished, and (3) succeeding. We pick the worst trajectory in the attacker's viewpoint and record the rollout value as the exhaustive value. If the worst trajectory from the first-round is unfinished, we conduct the second-round validation from the end-point of that trajectory.

We sample 500 initial states from the states with negative and positive DDQN rollout value, and we obtain 33% and 1.2% error rate, respectively. Fig. 7 (b) shows trajectories by DDQN rollout and exhaustive search with their payoff, which is defined in (7). The first row of Fig. 7 (b) shows a rare example in which the attacker succeeds in validation but fails in the DDQN rollout. This discrepancy is due to restricting the defender to maintaining the same action within each short time interval. However, we note that we have very low error rate for states of positive DDQN rollout value. In this case, even though the defender fails in all exhaustive trajectories, the attacker barely escapes with -0.02 average exhaustive rollout value. The second row of Fig. 7 (b) shows that the attacker fails in validation but succeeds in the DDQN rollout. We make the following two observations: (1) the defender in the DDQN rollout chooses to chase the attacker right away, while in validation, the defender chooses to block the path for the attacker to enter into the target set. This result suggests that the learned policy may not be optimal. (2) it seems possible that the attacker can reach the target set before the defender can get in the way. However, the defender adopts some unintuitive action sequence, e.g., a clockwise turn in the bottom left. This action results in a infrequently visited joint state, at which the neural network does not produce an accurate approximate value. This indicates that our "exhaustive" validation is able to expose vulnerabilities in the learned attacker policy.

V. Conclusion

In this paper we have presented a learning approach for computing solutions to reach-avoid optimal control problems. We first presented the optimality condition for the value function of reach-avoid problems and how it does not naturally induce a contraction mapping in the space of value functions. Upon introducing a discount term, we then constructed a discounted version of the optimality condition, or DRABE, which does induce a contraction. Notably, we show that using the DRABE operator we are able to recover conservative approximations of the exact reach-avoid set for any $\gamma \leq 1$. We then proceeded to use this discounted update in tabular and deep Q-learning in order to compute approximately optimal reachable sets and controllers for a variety of non-linear systems. Our results are a first step in showing how Hamilton-Jacobi reachability can be coupled with reinforcement learning methodologies to yield solutions for problems where one wishes to optimize for both safety and liveness.

Appendix A
Fixed-Point RABE Derivation and Other Proofs

In what follows we show how to use Bellman's principle of optimality to obtain the optimality condition for the value function:

$$\begin{aligned} V(s) &= \inf_{\mathbf{u}} \min_{\tau\in\{0,1,\ldots\}} \max\big\{l(\xi_s^{\mathbf{u}}(\tau)),\ \max_{\kappa\in\{0,\tau\}} g(\xi_s^{\mathbf{u}}(\kappa))\big\} \\ &= \min\Big\{ \max\big\{l(s), g(s)\big\}, \inf_{\mathbf{u}} \min_{\tau\in\{\mathbf{1},\ldots\}} \\ &\qquad \max\big\{l\big(\xi_{s_1}^{\mathbf{u}}(\tau)\big),\ \max_{\kappa\in\{0,\tau\}} g\big(\xi_s^{\mathbf{u}}(\kappa)\big)\big\}\Big\} \qquad (20) \\ &= \min\Big\{ \max\big\{l(s), g(s)\big\}, \inf_{\mathbf{u}} \min_{\tau\in\{1,\ldots\}} \\ &\qquad \max\big\{g(s), \max\{l(\xi_{s_1}^{\mathbf{u}}(\tau)),\ \max_{\kappa\in\{1,\tau\}} g(\xi_{s_1}^{\mathbf{u}}(\kappa))\}\big\}\Big\} \\ &= \min\Big\{ \max\big\{l(s), g(s)\big\}, \inf_{u} \max\big\{g(s), \\ &\qquad \inf_{\mathbf{u}_1} \min_{\tau\in\{1,\ldots\}} \max\{l(\xi_{s_1}^{\mathbf{u}_1}(\tau)),\ \max_{\kappa\in\{1,\tau\}} g(\xi_{s_1}^{\mathbf{u}_1}(\kappa))\}\big\}\Big\} \\ &= \min\{\max\{l(s), g(s)\}, \inf_{u} \max\{g(s), V(s_+^u)\}\} \\ &= \min\{\max\{l(s), g(s)\}, \max\{g(s), \inf_{u} V(s_+^u)\}\}, \end{aligned}$$

where $s_+^u = s + f(s,u)\Delta t$, and $\mathbf{u}_1$ refers to the control signal without the first control input. Noting that $\min\{\max\{a,b\},\max\{b,c\}\} = \max\{b,\min\{a,c\}\}$ we obtain the fixed-point RABE:

$$V(s) = \max\Big\{g(s),\ \min\big\{l(s), \inf_{u\in\mathcal{U}} V(s_+^u)\ \big\}\ \Big\}. \qquad (21)$$

Now we prove the proposed discounted (D)RABE induces a contraction mapping under the supremum norm, as described in Proposition 1.

TABLE 1: The hyper-parameters used in different environments. PP: point particle, DC: Dubins car, LL: Lunar Lander, AD: attack-defense.

Environment	PP		DC	LL	AD
	Fig. 2	Fig. 3, 4			
state dimension	2		3	6	6
action set dimension	3		3	4	9
# gradient updates (T)	12M	$400,000$		5M	4M
NN architecture	(100, 20)			(512, 512, 512)	
NN activation	Tanh				
learning rate	$\max\{0.001 \times 0.8^{\lfloor 20x/T \rfloor},\ 0.0001\}^*$				
optimizer	Adam		AdamW		
discount rate (γ)	§	0.9999		§	
exploration-exploitation	$\max\{0.95 \times 0.6^{\lfloor 20x/T \rfloor},\ 0.05\}^*$				
DDQN soft-update	0.01				
replay buffer size	10000			50000	
initialization†	without		$\max\{l, g\}$	g	$\max\{l, g\}$

* x: the number of updates so far, T: maximal number of updates.
† with uniform samples in the $x - y$ plane
§ $\min\{1 - 0.2 \times 0.5^{\lfloor 20x/T \rfloor},\ 0.999999\}$

Proof of Proposition 1. Observing that $|\max\{a, b\} - \max\{a, c\}| \leq |b - c|$, $\forall a, b, c \in \mathbb{R}$, we have

$$\begin{aligned}
&\Big|\max\Big\{\min\{\min_{u \in \mathcal{U}} V_\gamma(s_+^u),\ l(s)\},\ g(s)\Big\} - \\
&\quad \max\Big\{\min\{\min_{u \in \mathcal{U}} \tilde{V}_\gamma(s_+^u),\ l(s)\},\ g(s)\Big\}\Big| \\
\leq &\left|\min\{\min_{u \in \mathcal{U}} V_\gamma(s_+^u),\ l(s)\} - \min\{\min_{u \in \mathcal{U}} \tilde{V}_\gamma(s_+^u),\ l(s)\}\right| \\
\leq &\left|\min_{u \in \mathcal{U}} V_\gamma(s_+^u) - \min_{u \in \mathcal{U}} \tilde{V}_\gamma(s_+^u)\right|.
\end{aligned}$$

Without loss of generality, suppose $\min_{u \in \mathcal{U}} V_\gamma(s_+^u) > \min_{u \in \mathcal{U}} \tilde{V}_\gamma(s_+^u)$ and let $u^* := \arg\min_{u \in \mathcal{U}} \tilde{V}_\gamma(s_+^u)$. Thus, $V_\gamma(s_+^{u^*}) \geq \min_{u \in \mathcal{U}} V_\gamma(s_+^u) \geq \tilde{V}_\gamma(s_+^{u^*})$. Then, we have

$$\begin{aligned}
&\left|B_\gamma[V_\gamma](s) - B_\gamma[\tilde{V}_\gamma](s)\right| \\
\leq &\gamma \left|\min_{u \in \mathcal{U}} V_\gamma(s_+^u) - \min_{u \in \mathcal{U}} \tilde{V}_\gamma(s_+^u)\right| \leq \gamma \left|V_\gamma(s_+^{u^*}) - \tilde{V}_\gamma(s_+^{u^*})\right| \\
\leq &\gamma \max_{u \in \mathcal{U}} \left|V_\gamma(s_+^u) - \tilde{V}_\gamma(s_+^u)\right| \leq \gamma \max_{s \in \mathcal{S}} \left|V_\gamma(s) - \tilde{V}_\gamma(s)\right|. \quad \square
\end{aligned}$$

Further, we prove the convergence result for the Reach-Avoid Q-Learning Scheme, as presented in Proposition 2.

Proof of Proposition 2. Our proof follows from the general proof of Q-learning convergence for finite-state, finite-action Markov decision processes presented in [22]. Our transition dynamics f, initialization and policy randomization, and learning rate α_k satisfy Assumptions 1, 2, and 3 in [22] in the standard way. The only critical difference in the proof is the contraction mapping, which we obtain under the supremum norm by Proposition 1: with this, Assumption 5 in [22] is met, granting convergence of Q-learning by Theorem 3 in [22]. □

In addition, we prove that the fixed-point solution of DRABE is the solution of RABE when the discount rate approaches 1, as stated in Proposition 3.

Proof of Proposition 3. Let l_t and g_t stand for the target margin and safety margin at t-th time step of a discrete-time trajectory. The discounted value of this trajectory is

$$\begin{aligned}
\mathcal{V}_\gamma^{\mathbf{u}}(s) \ = \ &(1 - \gamma)\max\{l_0, g_0\} + \gamma \max\Big\{g_0, \min\big\{l_0, \\
&(1 - \gamma)\max\{l_1, g_1\} + \gamma \max\{g_1, \min\{l_1, \cdots\}\}\big\}\Big\}, \quad (22)
\end{aligned}$$

which is the explicit form of the objective minimized in (15). By taking $\gamma \to 1$, we recover

$$\begin{aligned}
\lim_{\gamma \to 1} \mathcal{V}_\gamma^{\mathbf{u}}(s) &= \max\Big\{g_0, \min\big\{l_0, \max\big\{g_1, \min\{l_1, \cdots\}\big\}\big\}\Big\} \\
&= \min \quad \big\{\max\{g_0, l_0\}, \max\big\{g_0, g_1, \min\{l_1, \cdots\}\big\}\big\} \\
&= \min \quad \big\{\max\{g_0, l_0\},\ \max\{g_0, g_1, l_1\}, \\
&\qquad\qquad \max\{g_0, g_1, g_2, \min\{l_2, \cdots\}\}\big\} \\
&= \min_{\tau \in \{0, 1, \cdots\}} \max\Big\{l_\tau,\ \max_{\kappa \in \{0, 1, \cdots, \tau\}} g_\kappa\Big\} = \mathcal{V}^{\mathbf{u}}(s).
\end{aligned}$$

Thus,

$$\lim_{\gamma \to 1} V_\gamma(s) = \lim_{\gamma \to 1} \min_{\mathbf{u} \in \mathbb{U}} \mathcal{V}_\gamma^{\mathbf{u}}(s) = \min_{\mathbf{u} \in \mathbb{U}} \mathcal{V}^{\mathbf{u}}(s) = V(s). \quad \square$$

Finally, we prove that any choice of γ induces a conservative approximation to the exact reach-avoid set, as stated in Theorem 1.

Proof of Theorem 1. For brevity, let $u^* := \arg\min_u V(s_+^u)$. From (7) and (8), we know $\max\{l(s), g(s)\}$ is always bigger or equal than $V(s)$, so we have $V_{\gamma_1}(s) \geq V_{\gamma_2}(s)$ for any $\gamma_1 \leq \gamma_2$. Here we enumerate all cases as follows

1) $V(s_+^{u^*}) \geq l(s)$:

$$\begin{aligned}
&\max\{l(s), g(s)\} - \max\big\{\min\big\{V(s_+^{u^*}),\ l(s)\big\},\ g(s)\big\} \\
&= \ \max\{l(s), g(s)\} - \max\{l(s), g(s)\} = 0.
\end{aligned}$$

2) $g(s) \geq V(s_+^{u^*})$ and $g(s) \geq l(s)$:

$$\begin{aligned}
&\max\{l(s), g(s)\} - \max\big\{\min\big\{V(s_+^{u^*}),\ l(s)\big\},\ g(s)\big\} \\
&= \ g(s) - g(s) = 0.
\end{aligned}$$

3) $l(s) > V(s_{+}^{u^*}) \geq g(s)$:

$$\max\{l(s), g(s)\} - \max\big\{\min\big\{V(s_{+}^{u^*}),\ l(s)\big\},\ g(s)\big\} \\ =\ l(s) - V(s_{+}^{u^*}) > 0.$$

4) $l(s) > g(s) > V(s_{+}^{u^*})$:

$$\max\{l(s), g(s)\} - \max\big\{\min\big\{V(s_{+}^{u^*}),\ l(s)\big\},\ g(s)\big\} \\ =\ l(s) - g(s) > 0.$$

Since $\forall \gamma_1 \leq \gamma_2,\ \forall s : V_{\gamma_1}(s) \leq 0 \rightarrow V_{\gamma_2}(s) \leq 0$, we have $\mathcal{RA}_{\gamma_1} \subseteq \mathcal{RA}_{\gamma_2}$. Also, taking the limit of (15) as $\gamma \rightarrow 1$ we recover $\lim_{\gamma \rightarrow 1} V_\gamma(s) = V(s)$.

Finally, we consider (Hausdorff) set convergence: we seek to show $\forall \delta > 0, \exists \gamma \in [0, 1) \mid d_H(\mathcal{RA}_\gamma, \mathcal{RA}) < \delta$. Since $\mathcal{RA}_\gamma \subseteq \mathcal{RA}$, we have: $d_H(\mathcal{RA}_\gamma, \mathcal{RA}) = \sup_{x \in \mathcal{RA}} \inf_{y \in \mathcal{RA}_\gamma} \|x - y\|$ (for an arbitrary norm $\|\cdot\|$ on $\mathcal{S}$). Provided that V is nowhere locally constant on its zero level set and the interior of $\mathcal{RA}$ is nonempty, we have that for any $\delta > 0$, there exists $\epsilon > 0$ such that near any x with $V(x) \leq 0$ there exists a point y, $\|x - y\| \leq \delta$ for which $V(y) < -\epsilon$. From the limit of (15), we also have that for any such ϵ, there exists a sufficiently large $\gamma \in [0, 1)$ for which $|V_\gamma(y) - V(y)| < \epsilon$, and hence $V_\gamma(y) < 0$. Therefore, for any $\delta > 0$ we can find $\gamma \in [0, 1)$ such that any point $x \in \mathcal{RA}$ has a corresponding point $y \in \mathcal{RA}_\gamma$ within distance δ. This concludes the proof. □

Appendix B
Practical Training Considerations

Table 1 lists the state dimension, action set dimension, DDQN hyper-parameters and NN training method in each environment. Among these hyper-parameters, we find that the discount rate, memory buffer size and NN architecture are the most crucial ones. If we start with very high γ, say 0.99, the convergence becomes extremely slow and DDQN cannot learn the value function well. For more complicated environment, we need deeper NN and larger memory buffer, so we have stronger representability and more diverse state-action pairs.

References

[1] E. Ackerman and M. Koziol. "In the Air with Zipline's Medical Delivery Drones". *IEEE Spectrum: Technology, Engineering, and Science News* (2019). (Accessed on 2021/02/08.)

[2] F. Siddiqui. "Waymo Becomes First Company to Launch Driverless Ride-Hailing to Public - The Washington Post". *The Washington Post* (Oct. 2020).

[3] NTSB. *Collision Between a Car Operating With Automated Vehicle Control Systems and a Tractor-Semitrailer Truck Near Williston, Florida, May 7, 2016.* Tech. rep. NTSB/HAR-17/02. United States National Transportation Safety Board, 2017.

[4] NTSB. *Collision Between a Sport Utility Vehicle Operating With Partial Driving Automation and a Crash Attenuator, Mountain View, California, March 23, 2018.* Tech. rep. NTSB/HAR-20/01. United States National Transportation Safety Board, 2020.

[5] NTSB. *Collision Between Car Operating with Partial Driving Automation and Truck-Tractor Semitrailer, Delray Beach, Florida, March 1, 2019.* Tech. rep. NTSB/HAB-20/01. (Accessed on 2021/02/08.) United States National Transportation Safety Board, 2020.

[6] C. Liu, T. Arnon, C. Lazarus, et al. "Algorithms for Verifying Deep Neural Networks". *Foundations and Trends in Optimization* 4 (2021).

[7] J. F. Fisac*, A. K. Akametalu*, M. N. Zeilinger, et al. "A general safety framework for learning-based control in uncertain robotic systems". *IEEE Transactions on Automatic Control (in press)* (2018).

[8] S. Li and O. Bastani. "Robust Model Predictive Shielding for Safe Reinforcement Learning with Stochastic Dynamics". *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 7166–7172.

[9] B. Alpern and F. B. Schneider. "Defining Liveness". *Information Processing Letters* 21.4 (Oct. 1985), pp. 181–185.

[10] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. "Hamilton-Jacobi Reachability: A Brief Overview and Recent Advances". *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2017, pp. 2242–2253.

[11] I. M. Mitchell. "The flexible, extensible and efficient Toolbox of Level Set Methods". *Journal of Scientific Computing* 35.2 (2008), pp. 300–329.

[12] S. L. Herbert, M. Chen, S. Han, et al. "FaSTrack: a Modular Framework for Fast and Guaranteed Safe Motion Planning". *Conference on Decision and Control (CDC) (submitted)* (Mar. 2017). arXiv: 1703.07373.

[13] V. Rubies-Royo and C. Tomlin. "Recursive Regression with Neural Networks: Approximating the HJI PDE Solution". 2017. arXiv: 1611.02739 `[cs.LG]`.

[14] S. Bansal and C. Tomlin. "DeepReach: A Deep Learning Approach to High-Dimensional Reachability". 2020. arXiv: 2011.02082 `[cs.RO]`.

[15] V. Rubies-Royo, D. Fridovich-Keil, S. Herbert, and C. J. Tomlin. "A Classification-based Approach for Approximate Reachability". *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 7697–7704.

[16] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, et al. "Bridging hamilton-jacobi safety analysis and reinforcement learning". *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8550–8556.

[17] E. A. Coddington and N. Levinson. "Theory of ordinary differential equations". Tata McGraw-Hill, 1955.

[18] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry. "Reach-Avoid Problems with Time-Varying Dynamics, Targets and Constraints". *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. HSCC '15. Seattle, Washington: Association for Computing Machinery, 2015, pp. 11–20.

[19] R. Isaacs. "Differential Games". New York: John Wiley and Sons, 1965.

[20] R. S. Sutton, A. G. Barto, and R. J. Williams. "Reinforcement learning is direct adaptive optimal control". *IEEE Control Systems Magazine* 12.2 (1992), pp. 19–22.

[21] C. J. C. H. Watkins and P. Dayan. "Q-learning". *Machine Learning* 8.3-4 (May 1992), pp. 279–292.

[22] J. N. Tsitsiklis. "Asynchronous Stochastic Approximation and Q-Learning". *Machine Learning* 16.3 (1994), pp. 185–202.

[23] H. Van Hasselt, A. Guez, and D. Silver. "Deep Reinforcement Learning with Double Q-Learning". *arXiv preprint arXiv:1509.06461* (2015).

[24] T. P. Lillicrap, J. J. Hunt, A. Pritzel, et al. "Continuous control with deep reinforcement learning." *ICLR (Poster)*. 2016.

[25] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". *PMLR*. Vol. 80. July 2018, pp. 1861–1870.

[26] O. Bastani. "Safe reinforcement learning via online shielding". *arXiv preprint arXiv:1905.10691* (2019).

[27] G. Brockman, V. Cheung, L. Pettersson, et al. "Openai gym". *arXiv preprint arXiv:1606.01540* (2016).