Ablation Study of How Run Time Assurance Impacts the Training and Performance of Reinforcement Learning Agents

Nathaniel Hamilton
Parallax Advanced Research
Beavercreek, OH, USA
nathaniel.hamilton@parallaxresearch.org

Kyle Dunlap
Parallax Advanced Research
Beavercreek, OH, USA
kyle.dunlap@parallaxresearch.org

Taylor T Johnson
Department of Computer Science
Vanderbilt University
Nashville, TN, USA
taylor.johnson@vanderbilt.edu

Kerianne L Hobbs
Autonomy Capability Team (ACT3)
Air Force Research Laboratory
Wright-Patterson Air Force Base, USA
kerianne.hobbs@us.af.mil

***Abstract*—Reinforcement Learning (RL) has become an increasingly important research area as the success of machine learning algorithms and methods grows. To combat the safety concerns surrounding the freedom given to RL agents while training, there has been an increase in work concerning *Safe Reinforcement Learning* (SRL). However, these new and safe methods have been held to less scrutiny than their unsafe counterparts. For instance, comparisons among safe methods often lack fair evaluation across similar initial condition bounds and hyperparameter settings, use poor evaluation metrics, and cherry-pick the best training runs rather than averaging over multiple random seeds. In this work, we conduct an *ablation study* using evaluation best practices to investigate the impact of *run time assurance* (RTA), which monitors the system state and intervenes to assure safety, on effective learning. By studying multiple RTA approaches in both on-policy and off-policy RL algorithms, we seek to understand which RTA methods are most effective, whether the agents become dependent on the RTA, and the importance of reward shaping versus safe exploration in RL agent training. Our conclusions shed light on the most promising directions of SRL, and our evaluation methodology lays the groundwork for creating better comparisons in future SRL work.**

***Index Terms*—Deep Reinforcement Learning, Safe Reinforcement Learning, Ablation Study, Run Time Assurance**

I. Introduction

Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) are fast-growing fields with growing impact, spurred by success in agents that learn to beat human experts in games like Go [1] and Starcraft [2]. However, these successes are predominantly limited to virtual environments. An RL agent learns a behavior policy that is optimized according a reward function. The policy is learned through interacting with/in the environment, making training on real-world hardware platforms prohibitively expensive and time-consuming. Additionally, RL allows agents to learn via trial and error, exploring *any behavior* during the learning process. In many cyber-physical domains, this level of freedom is unacceptable. Consider the example of an industrial robot arm learning to place objects in a factory. Some behaviors could cause the robot to damage itself, other elements in the factory, or nearby workers. To mitigate these set-backs, most RL training is performed in a simulated environment. After the training is completed in simulation, the learned policy can then be transferred to the real world via a *sim2real* transfer. However, this *sim2real* transfer can result in poor performance and undesirable behavior [3], [4].

To counteract these issues, the field of *Safe Reinforcement Learning* (SRL) has grown. Recent works demonstrate real-world online learning [5], optimal performance that does not require safety checking when deployed [6], and SRL approaches that work better than state-of-the-art DRL approaches [7]. Each new SRL paper claims to be the best, safest, most efficient, or least restrictive approach, but few prove these claims with valid demonstrations. When we tried to replicate studies using original source code, we found inconsistencies in their comparisons that made the SRL problem easier to solve. When these inconsistencies were accounted for, almost all the improvements claimed by the authors to come from the SRL method disappeared. These inconsistencies include (1) manipulating the initial conditions for only the SRL agents, so the RL methods have to learn how to solve the problem from a greater range of initial conditions, (2) forcing the "unsafe"

This work is supported by the Department of Defense (DoD) through the Air Force Office of National Defense Science & Engineering Graduate (NDSEG) Fellowship Program, the Air Force Research Laboratory Innovation Fund Program, and the Autonomy Technology Research (ATR) Center administered by Wright State University. The views expressed are those of the authors and do not reflect the official guidance or position of the United States Government, the Department of Defense or of the United States Air Force. This work has been approved for public release: distribution unlimited. Case Number AFRL-2023-1930.

RL agents to learn how to recover from unrecoverable unsafe conditions[1], and (3) tuning hyperparameters for their SRL methods while leaving the regular RL methods hyperparameters at a baseline. Any one of these inconsistencies can lead to results skewed in the SRL method's favor. Furthermore, many demonstrations fail to repeat trials across multiple random seeds. Because RL is a stochastic process, showing results from one random seed is not representative of the true performance of the algorithm. Only presenting the results of a singular trial allows for results to be selectively chosen to show a large improvement over existing methods. The work in [8], [9] highlights the importance of running experiments across at least 5 random seeds and averaging the results and showing the performance range in order to prove the trend of increased efficiency.

These issues in SRL publications bring rise to the need for better comparative studies and more *ablation studies*. An ablation study involves singling out and removing individual components of a complex system to understand their impact on the system as a whole. Ablation studies are used to determine causality and can prove which aspects of a system are actually the most important. In this work, we outline a better standard for comparing SRL approaches as we conduct a thorough ablation study on SRL approaches that use *Run Time Assurance* (RTA), an approach that monitors the output of the control policy for unsafe control actions and intervenes by modifying the output to assure system safety. RTA can be applied during training and after the training is complete.

Our contributions. This paper presents an in-depth investigation on how RTA configuration and usage choices impact RL training and final agent performance. The key contributions of this paper are as follows. **(1)** Evaluation across four different RTA approaches in addition to training with no RTA. **(2)** Evaluation of five different RTA training configurations that adapt how penalties are assigned during training and whether the RL agent has knowledge of a corrected action. **(3)** Evaluation of (1) and (2) on two different classes of RL: off-policy (SAC) and on-policy (PPO). **(4)** Evaluation of the true performance of each combination by training across 10 random seeds and averaging the results. **(5)** A large-scale (880 unique agents trained) experimental ablation study that covers (1), (2), and (3). **(6)** Analysis of the experimental results to provide practical insights and recommendations for training RL agents with RTA. In particular, answering these important questions: (VI-A) Do agents learn to become dependent on RTA? (VI-B) Which RTA configuration is most effective? (VI-C) Which RTA approach is most effective? (VI-D) Which works better with RTA, off-policy (SAC) or on-policy (PPO)? (VI-E) Which is more important, Reward Shaping or Safe Exploration?

[1]An example of an unrecoverable unsafe condition would include crashing the controlled vehicle, while a recoverable unsafe condition might include violating a set speed limit. While a low-fidelity simulation might allow recovery from a crash, that is usually not the case in reality. Therefore, when a crash occurs, the simulation should be terminated instead of allowed to continue. Otherwise, the agents trained to continue after such an event have to learn the optimal policy for that continuation in addition to the original task.

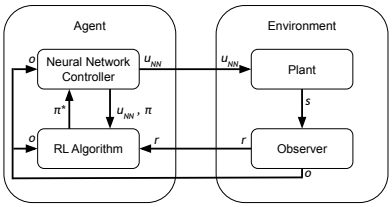


Fig. 1. DRL training interactions between agent and environment without RTA.

II. Deep Reinforcement Learning

Reinforcement Learning (RL) is a form of machine learning in which an agent acts in an environment, learns through experience, and increases its performance based on rewarded behavior. *Deep Reinforcement Learning* (DRL) is a newer branch of RL in which a neural network is used to approximate the behavior function, i.e. policy π. The basic construction of the DRL approach is shown in Fig. 1. The agent consists of the *Neural Network Controller* (NNC) and RL algorithm, and the environment consists of a plant and observer model. The environment can be comprised of any dynamical system, from Atari simulations ([10], [11]) to complex robotics scenarios ([3], [5], [8], [12]–[14]).

Reinforcement learning is based on the *reward hypothesis* that all goals can be described by the maximization of expected return, i.e. the cumulative reward [15]. During training, the agent chooses an action, u_{NN}, based on the input observation, o. The action is then executed in the environment, updating the internal state, s, according to the plant dynamics. The updated state, s, is then assigned a scalar reward, r, and transformed into the next observation vector. In all the examples shown in this work, we assume full observability, so all state information exists in the observation or can be reconstructed accurately from a single observation, i.e. the transformation by the observer is reversible. The observation is useful because it allows us to normalize the state values and change the input dimensions in order to ignore irrelevant variables and/or increase the importance of other variables by repeating them. The process of executing an action and receiving a reward and next observation is referred to as a *timestep*. Relevant values, like the input observation, action, and reward are collected as a data tuple, i.e. *sample*, by the RL algorithm to update the current NNC policy, π, to an improved policy, π^*. How often these updates are done is dependent on the RL algorithm.

In this work, we focus on model-free DRL algorithms, meaning the agent has no dependency on the environment model during training. Within model-free DRL algorithms, there are two main categories of training, *on-policy* and *off-policy*. On-policy algorithms use the learned policy to select the actions taken during training, while off-policy algorithms use a separate policy. This distinction will cause the RTA to have a varied impact on the learning process. Thus, we

repeat our experiments on two state-of-the-art DRL algorithms representing these two categories of training. *Proximal Policy Optimization* (PPO) is our on-policy algorithm[2] and *Soft Actor-Critic* (SAC) is our off-policy algorithm[3].

III. Safe Reinforcement Learning

When an RL agent explores states in a video game, the consequences of making a "wrong" move are limited. However, using RL in the real world has shown catastrophic results [3], [5]. The field of *Safe Reinforcement Learning* (SRL) was developed in response to RL's use on cyber-physical systems domain that interact with the real world in complex scenarios. In García and Fernández's comprehensive survey of SRL from 2015, they categorized the approaches into two main categories or styles: (1) modification of the optimality criterion and (2) modification of the exploration process [21]. In this work, we refer to these categories under the more general terms: (1) *reward shaping* and (2) *safe exploration*. Additionally, we introduce an emerging category of approaches, (3) *adversarial training/retraining*. Each are described in more detail in this section.

1) Reward Shaping: Reward shaping, the process of crafting a well-designed, optimal reward function, is essential for all forms of DRL since a poorly designed reward function can lead to unexpected and/or ineffective behavior [10]. Within SRL, reward shaping is used to reformulate the problem as a *Constrained Markov Decision Processes* (CMDP) [22]. Instead of optimizing performance according to a singular reward function, performance is optimized according to a task-oriented reward and a safety-focused cost [7], [23]–[27], so the agent learns a high-performing, safe policy. However, this style of SRL does not prohibit the agent from exploring unsafe behavior. Thus, it cannot be used to train on real hardware platforms. Instead, reward shaping techniques are limited to simulated environments and rely on high-quality transfer learning to be deployed in the real world.

2) Safe Exploration: Safe exploration approaches, which are often geared towards hardware deployment, ensure the agent remains 100% safe throughout the duration of training. Furthermore, this approach can be redesigned for deployment, ensuring the future safety of a static neural network that has completed training. Safe exploration techniques can be further broken down into the following three categories.

1) **Preemptive Shielding** where the set of actions the agent is allowed to choose from is preemptively reduced to only allow safe actions [11], [28].
2) **Safe-by-Construction** in which verification techniques are used, often on an abstraction of the learned policy, to verify safe behavior before being allowed to explore and develop further [29]–[31]. Alternatively, correct-by-construction can also be applied to a shielded RL solution [11].

[2]Other on-policy RL algorithms include A2C [16], TRPO [17], and ARS [13].

[3]Other off-policy RL algorithms include DQN [18], DDPG [19], and TD3 [20].

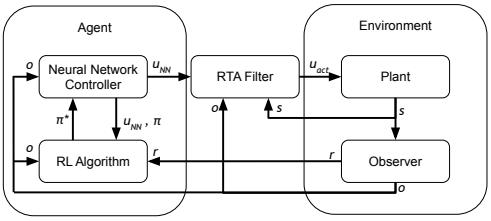


Fig. 2. DRL training interactions between the agent and the environment with RTA.

3) **Run Time Assurance (RTA) methods** filter the agent's desired actions, u_{NN}, to assure safety. In some cases, a monitor and/or decision module is used to determine whether the desired action provided by the learning agent is safe. In the event the agent's desired action is deemed unsafe, a different action that is determined to be safe is substituted and sent to the plant [5], [6], [32]–[39].

In this work, we focus solely on RTA methods for ensuring safe exploration, since they work across more examples with fewer scalability issues. An example of a general setup for safe exploration via RTA is shown in Fig. 2.

3) Adversarial Training/Retraining: The newest category of SRL approaches, *Adversarial Training/Retraining*, focuses on identifying unsafe behavior in the agent and then generating data to learn from and correct that behavior [40]–[42]. Most of the papers that use this approach focus on retraining an agent that already performs well in the environment. However, the approach can also be applied to an untrained network at the cost of requiring more training time.

IV. Run Time Assurance

One of the main contributions of this work is investigating how the RL training process is impacted by RTA approaches that filter unsafe control inputs to preserve system safety. For this paper, we focus on dynamical system plant models sampled discretely given by $s_{t+1} = f(s_t, u_t)$ where $s_t \in S$ is the state of the plant at timestep t, $S \subseteq \mathbb{R}^n$ is the real-valued state space, $u_t \in U$ is the control input to the plant at timestep t, with $U \subseteq \mathbb{R}^m$ the action space, and f is a function describing the state evolution from current state and control action.

For the dynamical system, inequality constraints $\varphi_i(s) : \mathbb{R}^n \rightarrow \mathbb{R}$, $\forall i \in \{1, ..., M\}$ can be used to define a set of M safety constraints, where the constraint is satisfied when $\varphi_i(s) \geq 0$. The admissible set $S_\varphi \subseteq S$, which is defined as the set of states where all constraints are satisfied, is then given by,

$$S_\varphi := \{s \in S \mid \varphi_i(s) \geq 0, \forall i \in \{1, ..., M\}\}. \quad (1)$$

Definition 1. *Safety and/or safe operation is achieved by always remaining within the admissible set, i.e. not violating any specified constraints. In the examples provided in this work, safety is defined on a finite time horizon, such that the operation is considered safe if* $\forall t \in [t_0, T], s_t \in S_\varphi$.

However, the ending time bound, T can be set to infinity for other systems that operate in perpetuity.

For RTA to ensure safe operation, we need to define a stricter subset of states to further constrain operations, known as the *control invariant* safe set, S_h. By operating in this stricter defined set, we avoid scenarios that can arise near the boundary of the admissible set, S_φ where, no matter the action executed, the next state will be outside the admissible set.

Definition 2. *The control invariant safe set, S_h, is a subset of the admissible set, S_φ, where $\forall s \in S_h, \exists u \in U, f(s,u) \in S_\varphi$.*

In this work, we first focus on two classes of RTA monitoring approaches, *explicit* and *implicit*, which define S_h differently. Explicit approaches use a pre-defined S_h, to determine when RTA intervention is necessary. To define S_h explicitly, we first define a set of M control invariant inequality constraints $h_i(s) : \mathbb{R}^n \to \mathbb{R}, \forall i \in \{1, ..., M\}$, where the constraints are satisfied when $h_i(s) \geq 0$. S_h is then given by,

$$S_h := \{s \in S \mid h_i(s) \geq 0, \forall i \in \{1, ..., M\}\}. \tag{2}$$

Implicit approaches use a defined backup control policy and the system dynamics to compute trajectories, which are used to determine when intervention is necessary. Implicitly, the S_h is defined as,

$$S_h := \{s \in S \mid \forall k \in [t_0, T], \phi_k^{u_\mathrm{b}}(s) \in S_\varphi\}, \tag{3}$$

where $\phi_k^{u_\mathrm{b}}$ represents a prediction of the state s for k timesteps under the backup control policy u_b. Because computing trajectories can be computationally expensive, explicit approaches tend to be more efficient. However, implicit approaches can be easier to implement since they do not require a precise definition of the control invariant safe set, which is difficult to define without being overly conservative.

Additionally, we split the RTA monitoring approaches further with two classes of intervention, *simplex* and *Active Set-Invariance Filter* (ASIF). The simplex approach switches from the primary control to a pre-defined backup controller if the system is about to leave the control invariant safe set [43]. The backup controller is usually less efficient at the desired control task, but meets desired safety and/or human-machine teaming constraints. One possible implementation for a simplex RTA filter is constructed as follows,

Simplex Filter

$$u_\mathrm{act}(s) = \begin{cases} u_\mathrm{NN}(o(s)) & \text{if} \quad \phi_k^{u_\mathrm{NN}}(s) \in S_h \\ u_\mathrm{b}(s) & \text{otherwise} \end{cases} \tag{4}$$

Here, $\phi_k^{u_\mathrm{NN}}(s)$ represents the predicted state if u_NN is applied for k discrete time intervals, and $o(s)$ is the observation created by state s.

ASIF approaches use barrier constraints to minimize deviations from the primary control signal while assuring safety [44]. One possible implementation for an ASIF RTA filter is constructed using a quadratic program as follows,

Active Set-Invariance Filter

$$u_\mathrm{act}(s, u_\mathrm{NN}) = \text{argmin} \, \|u_\mathrm{NN} - u_\mathrm{b}\| \quad \text{s.t.} \quad BC_i(s, u_\mathrm{b}) \geq 0, \quad \forall i \in \{1, ..., M\} \tag{5}$$

Here, $BC_i(s, u_\mathrm{b})$ represents a set of M barrier constraints [45] used to assure safety of the system. The purpose of barrier constraints is to enforce Nagumo's condition [46] and ensure $\dot{h}_i(s)$ is never decreasing $\forall i \in \{1, ..., M\}$ along the boundary of S_h. The function argmin finds the value of u_b closest to u_NN that still satisfies the barrier constraints. In this way, ASIF approaches apply the minimal change necessary to keep the system within S_φ at each timestep.

Using these defined approaches, we categorize our experiments in this paper according to the four derived classes of RTA monitoring approaches: *Explicit Simplex*, *Implicit Simplex*, *Explicit ASIF*, and *Implicit ASIF*.

V. Experiments

In order to answer the questions posed in the introduction, we have designed 880 experiments across multiple environments, RTA configurations, RTA approaches, and random seeds[4].

For each environment and DRL algorithm, we use established hyperparameters to limit the impact of tuning. We use 10 random seeds to generate our traces [8]. The evaluations run during training halt and freeze the NNC for the duration of the evaluation in order to better represent the performance of the agent at that point. After training is completed, the final learned policy is evaluated on the task 100 times to better approximate the expected performance if deployed. All evaluations are done in environments with and without the RTA active in order to identify any dependence on the RTA forming.

A. RTA configurations

In Fig. 2, we show a general method for including RTA in the training loop and purposefully left it vague. In this work, we have separated these various ways of connecting the RTA into the 5 configurations listed and explained below[5].The configurations are listed in order of increasing complexity. Each configuration builds on the previous ones, helping us observe the impact of each addition.

(1) Baseline (no RTA)

This configuration, demonstrated in Fig. 1, is used as a baseline comparison for all the RTA configurations. In this configuration, the agent is learning using the RL algorithm without any modifications. o is the input observation that led

[4]Code for recreating the experiments is available at https://github.com/act3-ace/rta-ablation-study.

[5]More detailed descriptions for the configurations are provided in the Appendix of the extended version available at https://arxiv.org/abs/2207.04117.

to the agent providing the desired action, u_{NN}, and r is the reward value computed by reaching the next state. The data tuple is $data = \{o, u_{\text{NN}}, r\}$.

(2) Baseline punishment

In this configuration, we assign a negative reward, i.e. punishment $p < 0$, if $unsafe?$ returns true, meaning at least one safety constraint was violated. This configuration adds SRL-style reward shaping to the problem. Instead of only maximizing the reward, the problem has two goals: (1) completing the task and (2) minimizing the punishment, or cost, incurred from violating constraints. In this work. p is constant; however, scaling p according to the severity of the safety violation can be done. The remaining configurations cannot factor in this kind of punishment because they rely on safe exploration which does not allow any violations of the safety constraints.

$$data = \begin{cases} \{o, u_{\text{NN}}, r + p\}, & \text{if } unsafe? \\ \{o, u_{\text{NN}}, r\}, & \text{otherwise} \end{cases}$$

(3) RTA no punishment

This configuration is the simplest form of safe exploration. Nothing changes from the baseline configuration except the agent remains safe throughout the training process because of the RTA depicted in Fig. 2. The data tuple is $data = \{o, u_{\text{NN}}, r\}$, regardless of whether or not the RTA is intervening.

(4) RTA punishment

This configuration adds an element of reward shaping to the previous configuration. Since we want the agent to learn the correct action to take in a scenario without the help of an RTA, we assign a punishment for having the RTA intervene. By adding this punishment, p, to the reward, the agent should better identify safe actions.

$$data = \begin{cases} \{o, u_{\text{NN}}, r + p\}, & \text{if } intervening? \\ \{o, u_{\text{NN}}, r\}, & \text{otherwise} \end{cases}$$

(5) RTA Corrected Action

In this configuration, we build on the idea of helping the agent identify the correct action to take in states near violating the safety constraints. Instead of punishing the agent for having the RTA intervene, we correct the agent's output to match that of the RTA's. In this manner, the agent only learns the actions actually taken in the environment. The data tuple is $data = \{o, u_{\text{act}}, r\}$.

B. Training Environments

We ran our experiments in three environments with varying levels of complexity. Additional information on these environments is provided in the Appendix[6].

[6]The Appendix is available in the extended version available at https://arxiv.org/abs/2207.04117.

1) Pendulum: The first environment is a modified version of OpenAI Gym's *Pendulum-v0* environment[7][12]. We chose this environment for its accessibility, simplicity, and previous use in [38] as a good indicator of the effectiveness of SRL over standard DRL. We use the same initial conditions and constraints described in their work.

The goal of the agent in this environment is to control an actuator that applies torque to the system to keep a frictionless pendulum upright and within the bounds of $\pm 1rad \approx \pm 46°$. Thus, the inequality constraint the RTA is designed to uphold can be written as,

$$\varphi_1(s) := 1 - |\theta|, \tag{6}$$

where θ is the displacement angle of the pendulum measured from the upright position. θ and the angular momentum of the pendulum, ω, are components of the environment state, $s = [\omega, \theta]$. The observation used as the input for the RL agent is $o = [\cos(\theta), \sin(\theta), \omega]$. If this constraint is violated, the episode is terminated.

The RTA design implemented in this environment is a simple implicit simplex design with the backup controller described by Equation 7.

$$u_b(s) = \min\left(\max\left(\frac{-32}{\pi}\theta, -15\right), 15\right) \tag{7}$$

The reward function was modified from the original used in [38] by adding a constant, 5, which was chosen in order to make a majority of the reward values positive. By keeping the reward positive, the agent is encouraged to not terminate the episode early. If the reward were mostly negative, the agent might learn that quickly terminating the episode maximizes the return better than remaining near upright. The resulting reward function, Equation 8, has a cumulative maximum of 1000 instead of 0. In the configurations which require a punishment, we use $p = -1$.

$$r_t = 5 - (\theta_t^2 + 0.1\omega_t^2 + 0.001u_t^2) \tag{8}$$

2) Spacecraft 2D Spacecraft Docking & 3D: The next two environments come from [47] and focus on the spacecraft docking problem[8]. In these environments, an active *deputy* spacecraft, under the control of the RL agent, approaches a passive *chief* spacecraft. This scenario is considered in 2D with only the x and y components, as well as 3D with the x, y, and z components. The goal of the agent in these environments is to use mounted thrusters that move the *deputy* spacecraft in the x, y, and z directions to a docking region around the *chief* spacecraft located at the origin. The state and observation are $s = o = [x, y, z, \dot{x}, \dot{y}, \dot{z}]$. If the *chief* spacecraft were moving, the observation might be modified to instead use the relative distances and velocities instead.

[7]The original Python implementation of this environment can be found at github.com/openai/gym/envs/classic_control/pendulum.py

[8]The implementations of these environments can be found at https://github.com/act3-ace/SafeRL

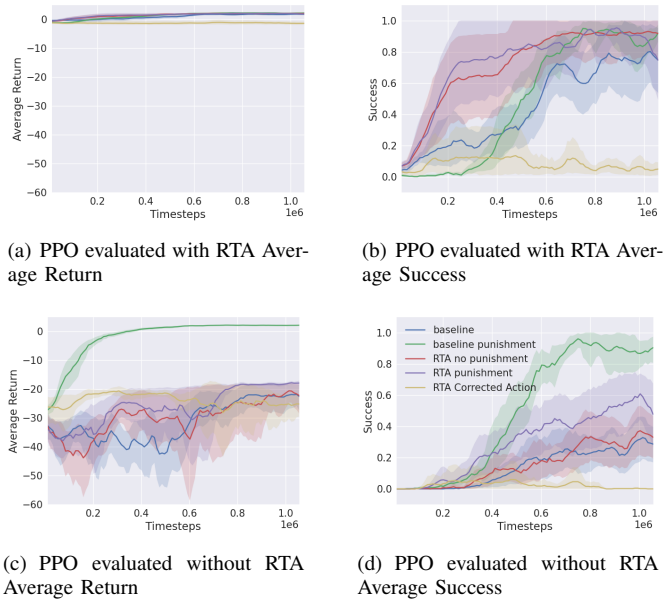
RTA is used in these environments to enforce a distance dependent speed limit $\varphi_1(s)$ and maximum velocity limits $\varphi_2(s)$-$\varphi_4(s)$ [48], [49]. Together, these constraints keep the deputy space craft controllable and prevent collisions caused by the deputy approaching too fast. The distance dependent speed limit is defined as,

$$\varphi_1(s) := \nu_D - \nu_{\mathrm{H}} + c d_{\mathrm{H}}, \tag{9}$$

where $\nu_D = 0.2m/s$ defines the maximum allowable docking velocity, $c = 2s^{-1}$ is a constant, $d_{\mathrm{H}} = (x^2 + y^2 + z^2)^{1/2}$ is the Euclidian distance between the deputy and chief spacecraft, and $\nu_{\mathrm{H}} = (\dot{x}^2 + \dot{y}^2 + \dot{z}^2)^{1/2}$ is the relative speed the deputy is approaching the chief. The maximum velocity, $v_{\max} = 10m/s$, limits can be written as inequality constraints,

$$\begin{aligned} \varphi_2(s) := v_{\max}^2 - \dot{x}^2, \quad \varphi_3(s) := v_{\max}^2 - \dot{y}^2, \\ \varphi_4(s) := v_{\max}^2 - \dot{z}^2. \end{aligned} \tag{10}$$

The reward functions for these environments are defined by sparse and dense components defined in Table I[9]. The sparsely defined terminal and safety reward components are only applied if the agent meets the specified requirements. In contrast, the dense reward component is computed after each timestep. In our experiments, the evaluation returns are computed using all the components defined in Table I. However, during training, the safety components are ignored unless the punishment is required by the configuration.



(a) PPO evaluated with RTA Average Return

(b) PPO evaluated with RTA Average Success

(c) PPO evaluated without RTA Average Return

(d) PPO evaluated without RTA Average Success

Fig. 3. Results collected from experiments run in the 2D Spacecraft Docking environment with an implicit simplex RTA. Each curve represents the average of 10 trials, and the shaded region is the 95% confidence interval about the mean. The large difference in return and success that is recorded with (a & b) and without (c & d) RTA shows that all agents trained with RTA learned to depend on it.

VI. Results and Discussion

In this section, we try to answer the questions posed in the introduction by analyzing the overarching trends found in our experiments. Including all the results collected would make this paper exceedingly long. Thus, we include select results that highlight the trends we found and provide all the results in the Appendix[6].

A. Do agents learn to become dependent on RTA?

Answer: Sometimes. Training RL agents with run time assurance *always* runs the risk of forming dependence. Furthermore, this phenomenon is more prevalent in our on-policy results than our off-policy results.

An agent is dependent on the RTA if the RTA is necessary for safe and successful behavior during deployment. We can determine if an agent has learned to be dependent on the RTA by evaluating performance with and without the RTA. We can identify when an agent has learned to form a dependence if the return and success drop significantly when evaluated without the RTA. If the agent is independent of the RTA, the performance metrics should be consistent when evaluated with and without the RTA.

We use gray to highlight examples where agents learned to form a dependency in Table II, which contains the final policy evaluations of our on-policy results across all the environments using the implicit simplex RTA approach. Note that not all agents in the highlighted rows (differentiated by the random seed used for training) learned a dependency, as evidenced by the increased standard deviation about the mean values. This further reinforces our answer that *sometimes* agents learn to become dependent on the RTA they are trained with. Instead of "always" or "never," whether the agent learns to become dependent on the RTA is a matter of chance, i.e. which random seed is used. Mania et al. show a great visualization in [13] of just how large an impact the random seed has on whether the agent learns a successful policy. The same is true here. If we had selected different random seeds, we would likely see different results for which agents learn to become dependent. However, it is the case that this can only happen if the agent is trained with RTA, and we have seen it is less likely to occur if the agent is punished for using the RTA as in the *RTA punishment* configuration. Note that the impact of the level/scale of punishment on whether dependence forms is left for future work.

To further demonstrate issues with agents forming a dependence on RTA, observe the drop in performance and success between evaluating with RTA (a & b) and without RTA (c & d) in Fig. 3. While the RTA helps all the agents reach success throughout the training process, the agents trained with RTA (*RTA no punishment*, *RTA punishment*, and *RTA Corrected Action*) do not maintain that same level of performance when evaluated without the RTA. In contrast, the *baseline punishment* agents learn successful behavior that works with and without the RTA.

[9]These values were provided by the authors of the environments during early development and do not match those published in [47]. These values were chosen with PPO as the target RL algorithm, which might explain why SAC struggled with learning to complete the task.

TABLE I
SPACECRAFT 2D SPACECRAFT DOCKING & 3D REWARD FUNCTION COMPONENTS

Terminal Rewards: All Configurations	
Successfully Docked ($d_{\mathrm{H}} \leq 20m$ and $\nu_{\mathrm{H}} \leq 0.2m/s$)	+1
Crashed ($d_{\mathrm{H}} \leq 20m$ with a velocity $\nu_{\mathrm{H}} > 0.2m/s$)	-1
Out of Bounds ($d_{\mathrm{H}} > 200m$)	-1
Over Max Time/Control	-1
Dense Reward: All Configurations	
Proximity	$0.0125(\Delta d_{\mathrm{H}})$
Safety Rewards: Punishment Configurations	
If RTA is Intervening	-0.001
Over Max Velocity	$-0.1 - 0.1(\nu_{\mathrm{H}} - v_{\max})$

TABLE II
THIS TABLE SHOWS FINAL POLICY EVALUATION RESULTS ACROSS ALL TEST ENVIRONMENTS TRAINED USING THE PPO ALGORITHM WITH THE IMPLICIT SIMPLEX RTA APPROACH. WE SHOW THE RECORDED PERFORMANCE MEASURED BY THE REWARD FUNCTION (RETURN) AND WHETHER THE AGENT WAS SUCCESSFUL AT COMPLETING THE TASK (SUCCESS). ROWS HIGHLIGHTED IN GRAY INDICATE A LEARNED DEPENDENCY.

Environment	Configuration	RTA	Return	Success
Pendulum	RTA no punishment	on	987.84 ± 10.86	1.00 ± 0.00
		off	987.59 ± 10.38	1.00 ± 0.00
Pendulum	RTA punishment	on	987.57 ± 11.20	1.00 ± 0.00
		off	987.85 ± 11.18	1.00 ± 0.00
2D Spacecraft Docking	RTA no punishment	on	2.02 ± 0.39	0.92 ± 0.27
		off	-22.39 ± 15.39	0.34 ± 0.47
2D Spacecraft Docking	RTA punishment	on	1.82 ± 0.60	0.73 ± 0.44
		off	-17.99 ± 5.16	0.46 ± 0.50
3D Spacecraft Docking	RTA no punishment	on	-33.29 ± 22.18	0.50 ± 0.50
		off	-23.51 ± 8.54	0.44 ± 0.50
3D Spacecraft Docking	RTA punishment	on	2.04 ± 0.39	0.89 ± 0.31
		off	2.07 ± 0.34	0.92 ± 0.27

B. Which RTA configuration is most effective?

Answer: *Baseline punishment* is the most effective. However, if safe exploration is necessary, *RTA punishment* is the most effective.

The most effective RTA configuration is the one that consistently trains the best performing agent evaluated without RTA. In the case of a tie and the final performance is comparable across multiple configurations, the best configuration is the one that learns the optimal performance quicker, i.e. requiring fewer samples. Across all our experiments, the *baseline punishment* configuration was consistently among the best performing agents. The next best performer was the *RTA punishment* configuration, which often outperformed the *baseline punishment* configuration in our PPO experiments, but did not do as well in all of our SAC experiments. We discuss why this might be the case in Section VI-D. To demonstrate this conclusion, we show the training curves in Fig. 4 from our experiments training agents across all configurations in both the 2D and 3D Spacecraft Docking environments using the PPO algorithm and the explicit simplex RTA approach.

In these particular examples, Fig. 4, both *baseline punishment* and *RTA punishment* have similar training curves that converge about the same return and success. This is similar across most of our experiments, except in some experiments when *RTA punishment* has a noticeably lower return because a dependence on the RTA formed.

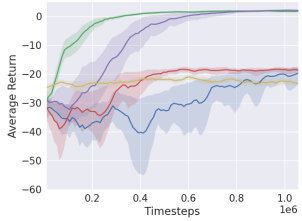
C. Which RTA approach is most effective?

Answer: The explicit simplex is the most effective RTA approach for training agents that consistently perform well and do not learn to depend on the RTA to maintain safety.

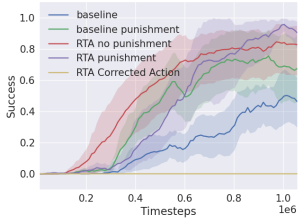
Fig. 5 shows the training curves for PPO agents trained in our 2D and 3D Spacecraft Docking environments with four different RTA approaches. All the training curves represent the PPO agents trained in the *RTA punishment* configuration and evaluated without the RTA. The curves broadly show ASIF RTA's guide the agent to success earlier on, but at the cost of increased sample complexity. The simplex approaches instead have a reduced sample complexity achieving a higher return sooner, which then leads to a greater chance of success.

We attribute these results to the differences between simplex and ASIF approaches. With simplex, the RTA does not intervene until the last moment, which allows for more agent-guided exploration, leading to more unique data samples. More unique data samples leads to a better approximation of the value- and/or Q-function, which reduces sample complexity. In contrast, ASIF approaches apply minimal corrections intended to guide the agent away from boundary conditions. This applies a greater restriction on agent-guided exploration, which can lead to more duplicated data samples.

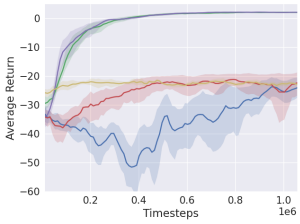
The implicit RTA approaches were less effective than the explicit approaches and were less consistent. In the 2D Spacecraft Docking environment, both ASIF training curves had similar return and success. However, in the 3D Spacecraft



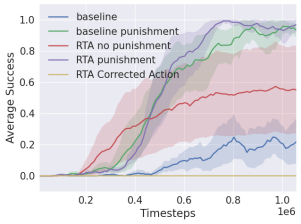
(a) 2D Docking, PPO evaluated without RTA Average Return



(b) 2D Docking, PPO evaluated without RTA Average Success

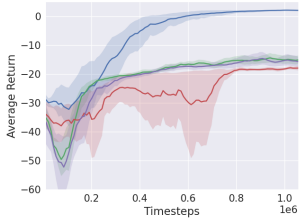


(c) 3D Docking, PPO evaluated without RTA Average Return

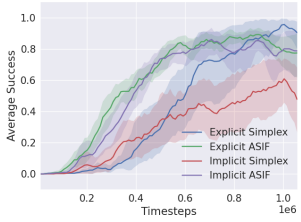


(d) 3D Docking, PPO evaluated without RTA Average Success

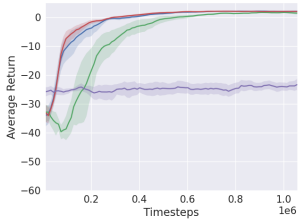
Fig. 4. Results collected from experiments run in the 2D (a & b) and 3D (c & d) Spacecraft Docking environment with an explicit simplex RTA. Each curve represents the average of 10 trials, and the shaded region is the 95% confidence interval about the mean. All plots show the *baseline punishment* and *RTA punishment* configurations learn at a similar rate and converge to similar levels of success and return.



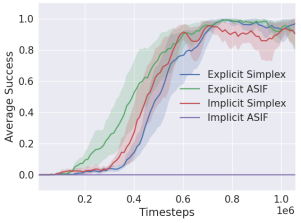
(a) PPO Average Return evaluated without RTA in the 2D Spacecraft Docking environment



(b) PPO Average Success evaluated without RTA in the 2D Spacecraft Docking environment



(c) PPO Average Return evaluated without RTA in the 3D Spacecraft Docking environment



(d) PPO Average Success evaluated without RTA in the 3D Spacecraft Docking environment

Fig. 5. Training curves collected from experiments run in the 2D and 3D Spacecraft Docking environments, training with PPO in the *RTA punishment* configuration across all four RTA approaches. Each curve represents the average of 10 trials, and the shaded region is the 95% confidence interval about the mean.

Docking environment, the explicit ASIF curve maintained the trend of earlier success with reduced return while the implicit ASIF curve failed to improve throughout the entire training process. Similarly, in the 3D Spacecraft Docking environment, the simplex curves had similar return and success, but in the 2D Spacecraft Docking environment the implicit simplex curve showed a large drop in both return and success.

Therefore, we reason explicit RTA approaches are better for training. Additionally, simplex approaches lead to a better performing agent in the long run.

D. Which works better with RTA, off-policy (SAC) or on-policy (PPO)?

Answer: On-policy methods see a greater benefit from training with RTA.

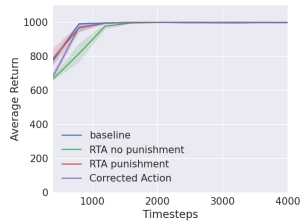
Our results showed PPO sees a greater benefit from training with RTA than SAC. This is likely a result of how the methods approach the *exploration versus exploitation* problem. Too much exploitation, using only known information (i.e. the current policy) too strictly, and the agent may never find the optimal policy. However, too much exploration and the agent may never learn what the goal is, particularly if the rewards are sparse. In general, on-policy methods leverage more exploitation than off-policy methods through their use of the learned policy.

On-policy methods exploit the learned policy. Therefore, guiding the agent to success and away from unsafe behavior helps the agent learn that behavior. As a result, the sample complexity is reduced. Fig. 6 (b & d) highlights this effect well. However, these benefits are hindered if the wrong configuration is chosen. Across almost all of our PPO experiments, the *RTA Corrected Action* configuration prevented the agents from improving the learned policy as shown in Fig. 4. This is likely a result of too much exploitation from the RTA intervening around boundary conditions, which prevented the agents from exploring other options to better define the optimal policy.

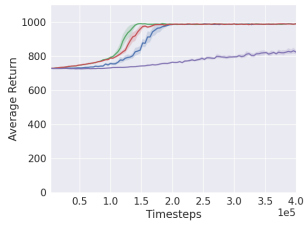
In contrast, **off-policy** methods have a larger focus on exploration. In particular, SAC maximizes entropy, assigning a higher value to unexplored state-action combinations. By restricting the actions taken near boundary conditions, the "unsafe" actions are never explored in actuality. Without making some distinction when the RTA intervenes makes the patterns harder to learn. This is shown in Fig. 7 where both *RTA no punishment* and *RTA Corrected Action* have a noticeably worse training curve than the *baseline* configuration, failing to improve at all through training. We see a similar trend in Fig. 6 (a & c) when SAC is used to learn the optimal policy for controlling our inverted pendulum. However, in this environment, the agent is still able to learn a successful policy.

E. Which is more important, Reward Shaping or Safe Exploration?

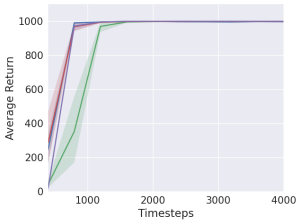
Answer: Reward shaping is generally more important for training. While safe exploration can improve sample complexity in some cases, a well-defined reward function is imperative for training successful RL agents.



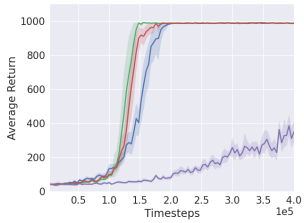
(a) SAC Average Return evaluated with RTA



(b) PPO Average Return evaluated with RTA

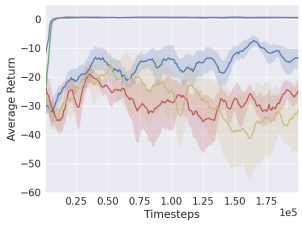


(c) SAC Average Return evaluated no RTA

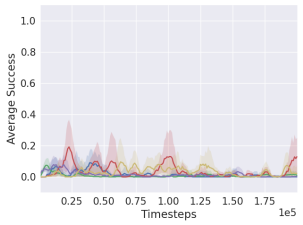


(d) PPO Average Return evaluated no RTA

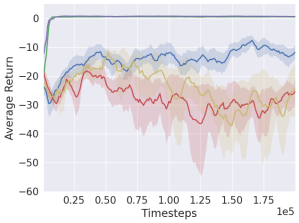
Fig. 6. Results collected from experiments run in the Pendulum environment. Each curve represents the average of 10 trials, and the shaded region is the 95% confidence interval about the mean. Note: maximum possible return in the environment is 1000.



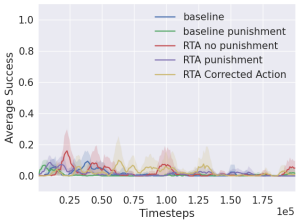
(a) SAC evaluated with RTA Average Return



(b) SAC evaluated with RTA Average Success



(c) SAC evaluated without RTA Average Return



(d) SAC evaluated without RTA Average Success

Fig. 7. Results collected from experiments run in the 2D Spacecraft Docking environment with explicit simplex RTA. Each curve represents the average of 10 trials, and the shaded region is the 95% confidence interval about the mean.

In every experiment where reward shaping was applied, we saw a more consistent and improved training curve. For example, note in Fig. 4 that the 95% confidence interval about *baseline punishment* and *RTA punishment* is smaller than *baseline* and *RTA no punishment* respectively. Additionally, the return and success tend to be much greater. The same trend shows in all of our experiments.

That said, safe exploration does improve sample complexity for on-policy RL, but the improvements are much greater when reward shaping is also applied in the *RTA punishment* configuration. Additionally, the punishment helps prevent the agent from becoming dependent on the RTA.

However, safe exploration on its own is no substitute for a well-defined/tuned reward function, as evidenced in our SAC experiments in the docking environments shown in Fig. 7. In these experiments, the agents with the *baseline punishment* and *RTA punishment* configurations quickly converged to an optimal performance, but the optimal performance did not result in success.

VII. Conclusions and Future Work

In conclusion, we trained 880 RL agents in 88 experimental configurations in order to answer some important questions regarding the use of RTA for training safe RL agents. Our results showed that **(1)** agents sometimes learn to become dependent on the RTA if trained with one, **(2)** *baseline punishment* and *RTA punishment* are the most effective configurations for training safe RL agents, **(3)** the explicit simplex RTA approach is most effective for consistent training results that do not depend on the RTA for safety, **(4)** PPO saw a greater benefit from training with RTA than SAC, suggesting that RTA may be more beneficial for on-policy than off-policy RL algorithms, and **(5)** effective reward shaping is generally more important than safe exploration for training safe RL agents.

In future work, and as more environments are released with RTA, we hope to expand this study to ensure our conclusions are representative of more complex training environments. Additionally, we would like to compare the effectiveness of correcting with RTA during training or retrain afterwards, evaluate *sim2real* transfer of the trained RL agents in representative robotic environments, and evaluate performance under environment and observation noise.

References

[1] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, Jan. 2016.

[2] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver, "Grandmaster Level in StarCraft II using Multi-Agent Reinforcement Learning," *Nature*, vol. 575, pp. 350–354, Oct. 2019.

[3] K. Jang, E. Vinitsky, B. Chalaki, B. Remer, L. Beaver, A. A. Malikopoulos, and A. Bayen, "Simulation to scaled city: zero-shot policy transfer for traffic control via autonomous vehicles," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, pp. 291–300, 2019.

[4] N. Hamilton, P. Musau, D. M. Lopez, and T. T. Johnson, "Zero-shot policy transfer in autonomous racing: reinforcement learning vs imitation learning," in *Proceedings of the 1st IEEE International Conference on Assured Autonomy*, pp. 11–20, 2022.

[5] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2018.

[6] N. Wagener, B. Boots, and C.-A. Cheng, "Safe reinforcement learning using advantage-based intervention," *arXiv preprint arXiv:2106.09110*, 2021.

[7] K. Jothimurugan, R. Alur, and O. Bastani, "A composable specification language for reinforcement learning tasks," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 13041–13051, Curran Associates, Inc., 2019.

[8] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, p. 3207–3214, 2018.

[9] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare, "Deep reinforcement learning at the edge of the statistical precipice," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[10] N. Hamilton, L. Schlemmer, C. Menart, C. Waddington, T. Jenkins, and T. T. Johnson, "Sonic to knuckles: evaluations on transfer reinforcement learning," in *Unmanned Systems Technology XXII*, vol. 11425, p. 114250J, International Society for Optics and Photonics, 2020.

[11] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, p. 2669–2678, 2018.

[12] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[13] H. Mania, A. Guy, and B. Recht, "Simple random search of static linear policies is competitive for reinforcement learning," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 1805–1814, 2018.

[14] N. Bernini, M. Bessa, R. Delmas, A. Gold, E. Goubault, R. Pennec, S. Putot, and F. Sillion, "A few lessons learned in reinforcement learning for quadcopter attitude control," in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, (New York, NY, USA), pp. 1–11, Association for Computing Machinery, 2021.

[15] D. Silver, "Lectures on reinforcement learning." URL: https://www.davidsilver.uk/teaching/, 2015.

[16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, PMLR, 2016.

[17] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, PMLR, 2015.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[19] 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.

[20] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, pp. 1587–1596, PMLR, 2018.

[21] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[22] E. Altman, *Constrained Markov decision processes*, vol. 7. CRC Press, 1999.

[23] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *International Conference on Machine Learning*, pp. 22–31, PMLR, 2017.

[24] A. Wachi and Y. Sui, "Safe reinforcement learning in constrained markov decision processes," in *International Conference on Machine Learning*, pp. 9797–9806, PMLR, 2020.

[25] D. Ding, X. Wei, Z. Yang, Z. Wang, and M. Jovanovic, "Provably efficient safe exploration via primal-dual policy optimization," in *International Conference on Artificial Intelligence and Statistics*, pp. 3304–3312, PMLR, 2021.

[26] A. HasanzadeZonuzy, D. M. Kalathil, and S. Shakkottai, "Learning with safety constraints: Sample complexity of reinforcement learning for constrained mdps," *arXiv preprint arXiv:2008.00311*, 2020.

[27] H. Satija, P. Amortila, and J. Pineau, "Constrained markov decision processes via backward value functions," in *International Conference on Machine Learning*, pp. 8502–8511, PMLR, 2020.

[28] M. Wu, J. Wang, J. Deshmukh, and C. Wang, "Shield synthesis for real: Enforcing safety in cyber-physical systems," in *2019 Formal Methods in Computer Aided Design (FMCAD)*, pp. 129–137, IEEE, 2019.

[29] C. Neary, C. Verginis, M. Cubuktepe, and U. Topcu, "Verifiable and compositional reinforcement learning systems," *arXiv preprint arXiv:2106.05864*, 2021.

[30] L. Zhang, R. Zhang, T. Wu, R. Weng, M. Han, and Y. Zhao, "Safe reinforcement learning with stability guarantee for motion planning of autonomous vehicles," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–10, 2021.

[31] Y. Wang, C. Huang, Z. Wang, Z. Wang, and Q. Zhu, "Verification in the loop: Correct-by-construction control learning with reach-avoid guarantees," *arXiv preprint arXiv:2106.03245*, 2021.

[32] N. Hunt, N. Fulton, S. Magliacane, T. N. Hoang, S. Das, and A. Solar-Lezama, "Verifiably safe exploration for end-to-end reinforcement learning," in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, (New York, NY, USA), p. 1–11, Association for Computing Machinery, 2021.

[33] Z. Xiong and S. Jagannathan, "Scalable synthesis of verified controllers in deep reinforcement learning," *arXiv preprint arXiv:2104.10219*, 2021.

[34] Y. Li, N. Li, H. E. Tseng, A. Girard, D. Filev, and I. Kolmanovsky, "Safe reinforcement learning using robust action governor," in *Learning for Dynamics and Control*, pp. 1093–1104, PMLR, 2021.

[35] N. Fulton and A. Platzer, "Safe reinforcement learning via formal methods," in *AAAI Conference on Artificial Intelligence*, p. 6485–6492, 2018.

[36] A. Desai, S. Ghosh, S. A. Seshia, N. Shankar, and A. Tiwari, "Soter: a runtime assurance framework for programming safe robotics systems," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 138–150, IEEE, 2019.

[37] A. Murugesan, M. Moghadamfalahi, and A. Chattopadhyay, "Formal methods assisted training of safe reinforcement learning agents," in *NASA Formal Methods Symposium*, pp. 333–340, Springer, 2019.

[38] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3387–3395, 2019.

[39] H. Zhao, X. Zeng, T. Chen, Z. Liu, and J. Woodcock, "Learning safe neural network controllers with barrier certificates," in *International Symposium on Dependable Software Engineering: Theories, Tools, and Applications*, pp. 177–185, Springer, 2020.

[40] D. T. Phan, R. Grosu, N. Jansen, N. Paoletti, S. A. Smolka, and S. D. Stoller, "Neural simplex architecture," in *NASA Formal Methods Symposium*, pp. 97–114, Springer, 2020.

[41] X. Wang, S. Nair, and M. Althoff, "Falsification-based robust adversarial reinforcement learning," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 205–212, IEEE, 2020.

[42] X. Yang, T. Yamaguchi, H.-D. Tran, B. Hoxha, T. T. Johnson, and D. Prokhorov, "Neural network repair with reachability analysis," *arXiv preprint arXiv:2108.04214*, 2021.

[43] J. G. Rivera, A. A. Danylyszyn, C. B. Weinstock, L. R. Sha, and M. J. Gagliardi, "An architectural description of the simplex architecture.," tech. rep., CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 1996.

[44] T. Gurriet, *Applied Safety Critical Control*. PhD thesis, California Institute of Technology, 2020.

[45] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, pp. 3420–3431, IEEE, 2019.

[46] M. Nagumo, "Über die lage der integralkurven gewöhnlicher differentialgleichungen," *Proceedings of the Physico-Mathematical Society of Japan. 3rd Series*, vol. 24, pp. 551–559, 1942.

[47] U. J. Ravaioli, J. Cunningham, J. McCarroll, V. Gangal, K. Dunlap, and K. L. Hobbs, "Safe reinforcement learning benchmark environments for aerospace control systems," in *2022 IEEE Aerospace Conference (50100)*, pp. 1–18, IEEE, 2022.

[48] K. Dunlap, M. Mote, K. Delsing, and K. L. Hobbs, "Run time assured reinforcement learning for safe satellite docking," in *2022 AIAA SciTech Forum*, pp. 1–20, 2022.

[49] K. Dunlap, M. Hibbard, M. Mote, and K. Hobbs, "Comparing run time assurance approaches for safe spacecraft docking," *IEEE Control Systems Letters*, 2021.