Safe Policy Optimization for Reinforcement Learning in Robotics

Hao Wang and Zhen Kan, Department of Automation, University of Science and Technology of China, Hefei, Anhui, China

 doi:10.1016/B978-0-443-14081-5.00041-6

Abstract

Learning-based policy optimization methods have shown great potential for building general-purpose control systems in robotics. However, existing methods still struggle to achieve complex task objectives while ensuring policy safety during learning and deployment phases for robot systems. Motivated to address these challenges, this chapter introduce the authors' three safe policy optimization methods based on data-driven control theory, projection, and shielded planning, respectively, for reinforcement leaning in robotics. In particular, this chapter includes Data-driven Safe Policy Optimization (D^2SPO), a novel reinforcement learning (RL) based policy improvement method that jointly learns a control barrier function (CBF) for system safety and a linear temporal logic (LTL) guided RL algorithm for complex task objectives; Fast and Safe Policy Optimization (FSPO) algorithm, which consists of three steps: the first step involves reward improvement update, the second step projects the policy to the neighborhood of the baseline policy to accelerate the optimization process, and the third step addresses the constraint violation by projecting the policy back onto the constraint set. Such a projection-based optimization can improve the convergence and learning performance; and Shielded Planning guided Policy Optimization (SPPO), a new model-based safe reinforcement learning method that augments policy optimization algorithms with path planning and shielding mechanism for improving the exploration efficiency and ensuring the safety during the training process. The experimental results demonstrate the methods in this chapter outperform existing algorithms in terms of policy performance, learning efficiency, safety guarantees and task completion rate.

Key Points

The main key points of this work are summarized as follows:

- Data-driven Safe Policy Optimization (D^2SPO), a novel reinforcement learning (RL) that jointly learns a control barrier function (CBF) and a linear temporal logic (LTL) guided RL policy is developed. D^2SPO can significantly improve the system safety as RL agents interact with the environment, while efficiently accomplish various tasks with LTL objectives.
- Fast and Safe Policy Optimization (FSPO), a new projection-based optimization method is proposed to improve the convergence and learning performance. FSPO employs a first-order method to avoid expensive computations, enabling fast and safe policy optimization, especially for challenging tasks.
- Shielded Planning guided Policy Optimization (SPPO) that augments reinforcement learning with MPPI and shielding rules is introduced to improve exploration efficiency and ensure policy safety even during training. SPPO can transforms a constrained optimization problem into an unconstrained one, making the optimization process simpler, more stable, and easier to solve.

Chapter Outline

In this chapter, we present three safe policy optimization methods for reinforcement learning in robotics. The rest of this chapter is organized as follows. In Section "Key Points", we provide a short bulleted list of key points for this chapter. A sufficient introduction to the previous works is given in Section "Introduction". After that, we first introduce the preliminaries in Section "Preliminaries", then, the Data-driven Safe Policy Optimization (D^2SPO), the Fast and Safe Policy Optimization (FSPO), and the Shielded Planning guided Policy Optimization (SPPO) are introduced in Section "Data-Driven Safe Policy Optimization for Reinforcement Learning with Temporal Logic Specifications", Section "Projection-Based Fast and Safe Policy Optimization for Reinforcement Learning", and Section "Shielded Planning Guided Safe Policy Optimization for Reinforcement Learning", respectively. Finally, we summarize the contents of this chapter and our vision for future work in Section "Outlook".

Introduction

Reinforcement learning (RL), as a promising paradigm, enables an agent to interact with the environment iteratively to achieve desired behaviors with high returns. However, most RL agents interact with the environment relying on random exploration, which leads to much invalid data sampling, making agent exploration and learning inefficient, and generates undesired behavior, i.e., visiting undesired states to get high returns, even during training. For instance, when training a UAV for cruise missions, it is desired that the UAV can avoid collisions with obstacles that could damage itself not only in deployment but also in the training process. Although much recent effort has focused on lowering exploration burden (Co-Reyes *et al.*, 2018; Dong *et al.*, 2021; Eysenbach *et al.*, 2019; Wang *et al.*, 2023), or developing safe RL (Garca and Fernández, 2015; Modares *et al.*, 2023; Zhang *et al.*, 2022) to synthesize safe policies, how to improve exploration efficiency or ensure safety during training and deployment remains a challenging open problem, let alone ensuring them simultaneously.

Planning is a powerful tool to improve the exploration efficiency. Such methods, unlike model-free algorithms that learn policies purely by trial-and-error, can plan action trajectories for guided exploration through an internal model of the environment. Specifically, prior model-based methods can be classified as: (i) planning (Hafner *et al.*, 2019; Janner *et al.*,

2019; Lin *et al.*, 2022; Nguyen *et al.*, 2021; Schrittwieser *et al.*, 2020; Williams *et al.*, 2017; Zhang *et al.*, 2019) and (ii) improving sample efficiency of model-free methods via the learned models (Dong *et al.*, 2021; Hansen *et al.*, 2022; Sekar *et al.*, 2020). They both fully exploit the key advantages of model-based learning and have been extremely successful in application fields such as game-playing (Kaiser *et al.*, 2019; Schrittwieser *et al.*, 2020) and continuous control (Chua *et al.*, 2018; Hansen *et al.*, 2022; Janner *et al.*, 2019; Tassa *et al.*, 2012). However, the former is limited by the high cost of long-horizon planning, while the latter leads to poor performance due to the potential propagation of learned model biases into the policy. Another disadvantage of these methods is that they require the network to simulate everything in the environment to learn accurately predicting the future, which is extremely difficult. In order to overcome this challenge, Hansen *et al.* (2022), Schrittwieser *et al.* (2020) and Ye *et al.* (2021) learn the latent dynamics models directly through the reward prediction, completely avoiding the dependence of model learning on environmental observations. Although these methods work well, none of them consider the safety issue of the learned policy during training and deployment.

Safe reinforcement learning (Garca and Fernández, 2015) is an effective solution to adress the challenges of policy safety. In general, the main technical route for safe RL is either to abstract the problem as a constrained Markov decision process (CMDP) (Altman, 1999) or to utilize control-theoretic tools to constrain the original policy to meet safety requirements. Either way, it is necessary to trade-off the natural conflict between maximizing long-term return and satisfying the safety constraints. Concretely, CMDP-based safe RL methods (Achiam *et al.*, 2017; Borkar, 2005; Le *et al.*, 2019) are popular frameworks inspired by constrained optimization algorithms. There have been related works focusing on developing such frameworks. The works of Chow *et al.* (2017), Peng *et al.* (2022) and Tessler *et al.* (2018) solve the stochastic nonconvex saddle-point problem based on first-order primal-dual optimization. Despite the fact that such methods can eventually synthesize a safe policy, they struggle to guarantee the safety in the training process. To solve this problem, Achiam *et al.* (2017), Bharadhwaj *et al.* (2021), Jayant and Bhatnagar (2022), Le *et al.* (2019) and Liu *et al.* (2022) perform safety constraints in each iteration by solving restricted optimization problems to ensure safety during training. However, these methods are less robust compared to the unconstrained RL methods, due to the numerical instability associated with solving stochastic nonconvex problems (Lin *et al.*, 2020). While Yang *et al.* (2021, 2022) can ensure safety by mapping optimized policies to trust regions at each step, they are limited by high computational complexity, leading to poor scalability.

Meanwhile, recent advances have applied control-theoretic approaches to RL to learn policies with safety constraints. These methods enforce safe interactions between agent and environment through shielding, projection, planning or succinctly employing backup safe policies (Carr *et al.*, 2023; Li and Bastani, 2020; Thananjeyan *et al.*, 2021; Zhang *et al.*, 2023; Zhou *et al.*, 2023). The most prominent methods are those based on control barrier function (CBF) (Cohen and Belta, 2023; Dawson *et al.*, 2023; Modares *et al.*, 2023; Wabersich and Zeilinger, 2022; Zhang *et al.*, 2022) and reachability (Hsu *et al.*, 2023; Selim *et al.*, 2022; Yu *et al.*, 2022). However, since they usually rely on domain-specific heuristics to determine whether an action sampled from a policy can be safely performed, the performance of these methods cannot be evaluated explicitly. Alternatively, the restrictions on exploration due to forced interventions may lead to poor policy performance or even failure to find feasible solutions. Moreover, the control-theoretic techniques always require strong assumptions (e.g., smoothness or ergodicity) that are typically not satisfied.

Besides safety concerns, another challenge is the synthesis of control policies for complex task constraints. Due to the rich expressivity in specifying complex logic and temporal constraints, linear temporal logics (LTL) are used with RL to enable efficient learning (Bozkurt *et al.*, 2020; Cai *et al.*, 2021; Toro Icarte *et al.*, 2018; Vaezipoor *et al.*, 2021). However, most of these methods struggle to guarantee the safety during exploration and execution. In contrast, the works of Li *et al.* (2019) and Cai and Vasile (2021) ensure exploration safety and policy safety while satisfying complex task constraints by integrating CBF with temporal logic guided RL. However, the CBF in Li *et al.* (2019) and Cai and Vasile (2021) requires accurate dynamics model and has to be manually designed, which imposes considerable challenges in real-world applications.

Motivated to address the above challenges, in this chapter, we introduce three safe policy optimization methods based on data-driven control theory, projection, and shielded planning, respectively, for reinforcement learning in robotics. In particular, we first introduce the Data-driven Safe Policy Optimization (D^2SPO), a novel reinforcement learning (RL) method. D^2SPO can significantly improve the system safety as RL agents interact with the environment, while efficiently accomplish various tasks with LTL objectives by jointly learning a control barrier function (CBF) and a linear temporal logic (LTL) guided RL policy. Then, a new projection-based optimization method, named Fast and Safe Policy Optimization (FSPO), is proposed to improve the convergence and learning performance. FSPO employs a first-order method to avoid expensive computations, enabling fast and safe policy optimization, especially for challenging tasks. In addition, Shielded Planning guided Policy Optimization (SPPO) that augments reinforcement learning with model predictive control and shielding mechanism is developed to improve exploration efficiency and ensure policy safety during training. Based on the shielding mechanism, SPPO can transform the CMDP problem into an unconstrained MDP problem, which is easier to implement and more stable than typical constraint-based optimization methods.

Preliminaries

In this section, we introduce the main preliminaries used in this chapter.

Reinforcement Learning

As a sequential decision-making process, RL is usually modeled as a Markov decision process (MDP) $\mathcal{M} = (S, A, \mathcal{T}, r, \gamma)$, where S is the state space, A is the action space, $\mathcal{T} : S \times A \mapsto S$ is the state-action transition function, $r : S \times A \times S \mapsto \mathbb{R}$ is the reward function, and γ is the discount factor. Given a policy π that maps a state to an action, the reward r is received after applying the action. The expected cumulative reward is defined as $Q_\pi = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$. A common goal is to learn an optimal policy $\pi^* = \underset{\pi}{argmax} Q_\pi$.

Safe Reinforcement Learning

RL agents learn safe policies by interacting with a constrained environment, where safety means that the agents have a very low probability of violating constraints (i.e., entering unsafe subsets) when operating in the environment. Such a process is typically formulated as a Constrained Markov Decision Process (CMDP) (Altman, 1999; Chow *et al.*, 2017). A CMDP is a tuple $(S, A, \mathcal{T}, R, C)$, where S is the set of states, A is the set of actions, $\mathcal{T} : S \times A \times S \mapsto [0, 1]$ is the transition probability, $R : S \times A \mapsto \mathbb{R}$ is the reward function, and $C : S \times A \mapsto \mathbb{R}$ is the cost function. Let $\pi(a|s)$ with $s \in S$ and $a \in A$ denote a policy. That is, given a state s, an action a is selected according to $\pi(a|s)$ and the state transits from s to s' according to the state transition model $\mathcal{T}(s'|s, a)$. A reward $R(s, a)$ and cost $C(s, a)$ can then be received, respectively.

The goal is to find a policy which maximizes the cumulative discounted reward $J^R(\pi) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$ while keeping the cumulative discounted cost below h_C to satisfy the constraints, i.e., $J^C(\pi) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t)\right] \leq h_C$ where $\gamma \in (0, 1)$ denotes the discount factor, $\tau = (s_0, a_0, s_1, \cdots)$ denotes the trajectory, and $\tau\sim\pi$ is a trajectory induced by π satisfying $\pi : s_0\sim\mu, a_t\sim\pi(a_t|s_t), s_{t+1}\sim T(s_{t+1}|s_t, a_t)$, where μ is the initial state distribution.

Model Predictive Path Integral

Model Predictive Path Integration (MPPI) is a model-based policy optimization algorithm that iteratively updates system parameters through importance sampling (Williams *et al.*, 2015). Specifically, it iteratively updates the parameter distribution by estimating the expected return of sampled trajectories and taking the top k for importance-weighted averaging. Inspired by Hansen *et al.* (2022), a time-dependent multivariate Gaussian with diagonal covariance is used to fit the parameters in this work, i.e., $(\mu^0, \sigma^0)_{t:t+H}$ denotes a control input trajectory of length H with an initial mean $\mu^0 \in \mathbb{R}^{m\times H}$ and standard deviation $\sigma^0 \in \mathbb{R}^{m\times H}$, where m denotes the dimension of the control input. Using the rollouts generated by the dynamics model $s_{t+1} = d(s_t, u_t)$ with s_t, u_t denoting the state and control input at time t, respectively, one can sample N control input trajectories independently and estimate the total return Φ_Γ of a sampled trajectory $\Gamma = (u_0, u_1, \ldots, u_H)$ as

$$\Phi_\Gamma = \mathbb{E}_\Gamma\left[\varphi(s_H) + \sum_{t=0}^{H-1}\left(q(s_t) + \frac{1}{2}u_t^T \Sigma u_t\right)\right], \tag{1}$$

where $\varphi(s_H)$ is the terminal return, $q(s_t)$ is the state-dependent step return, $\frac{1}{2}u_t^T\Sigma u_t$ is the immediate control cost with Σ denoting a positive definite matrix, and $u_t\sim\mathcal{N}\left(\mu_t^{j-1}, \left(\sigma_t^{j-1}\right)^2 \mathbf{I}\right)$ at iteration $j-1$. By selecting and utilizing the top-k returns $\left\{\Phi^\star_{\Gamma,i}\right\}_{i=1}^k$, the mean and standard deviation at the jth iteration are updated as

$$\mu^j = \frac{\sum_{i=1}^k \Omega_i \Gamma_i^\star}{\sum_{i=1}^k \Omega_i}, \tag{2}$$

$$\sigma^j = \sqrt{\frac{\sum_{i=1}^k \Omega_i \left(\Gamma_i^\star - \mu^j\right)^2}{\sum_{i=1}^k \Omega_i}}, \tag{3}$$

where $\Omega_i = e^{\tau\left(\Phi^\star_{\Gamma,i} - \max_g\left(\Phi^\star_{\Gamma,g}\right)\right)}$, τ is a temperature parameter that controls the weighted 'sharpness', and $\Gamma_i^\star$ indicates the ith top-k trajectory corresponding to return estimate $\Phi_\Gamma^\star$. In MPPI, the above planning procedure is terminated after a fixed number of iterations J and a control input trajectory is sampled from the final distribution over action sequences. Note that planning is performed at each decision step t, but only the first action is executed, i.e., the receding-horizon MPC is adopted to generate a feedback policy.

Linear Temporal Logic

LTL is a formal language capable of expressing rich task specifications. Given the atomic propositions AP, the syntax of the LTL formula is defined as

$$\phi ::= true | ap | \neg\phi | \phi_1 \wedge \phi_2 | \bigcirc\phi | \phi_1 \mathrm{U} \phi_2,$$

where *true* is the Boolean value, $ap \in AP$ is an atomic proposition, ϕ, ϕ_1, ϕ_2 are LTL formulas, $\neg$ (negation) and $\wedge$ (conjunction) are standard Boolean operators, and $\bigcirc$ (next) and U (until) are temporal operators. The semantics of an LTL formulae is defined over an infinite sequence $\sigma = \sigma_0\sigma_1\ldots$ with $\sigma_i \in 2^{AP}$, $i \in \mathbb{Z}_{\geq 0}$, where 2^{AP} represents the power set of AP. Denote by $\sigma\phi$ if the word σ satisfies the LTL formula ϕ. Detailed descriptions of the syntax and semantics of LTL can be found in Baier and Katoen (2008).

An LTL formula can be translated to a nondeterministic Büchi automaton (NBA) $B = \{\mathcal{Q}, q_0, \Sigma, \Delta, F\}$, where $\mathcal{Q}$ is a finite set of automaton states, and $q_0 \in \mathcal{Q}$ is the initial state, $\Sigma = 2^{AP}$ is the input alphabet, $\Delta : S \times S \to 2^{\Sigma}$ is the transition function, and $F \in \mathcal{Q}$ is the set of accepting states. For any LTL formula ϕ, one can construct an NBA with input alphabet Σ accepting all and only words that satisfy ϕ (Baier and Katoen, 2008). NBA will be used in this work to indicate the task progress. To convert an LTL formula to an NBA, readers are referred to Gastin and Oddoux (2001) for algorithms and implementations.

Data-Driven Safe Policy Optimization for Reinforcement Learning With Temporal Logic Specifications

In this section, we develop Data-driven Safe Policy Optimization (D^2SPO), a novel reinforcement learning (RL) based policy improvement method that jointly learns a control barrier function (CBF) for system safety and a linear temporal logic (LTL) guided RL algorithm for complex task objectives. Unlike many existing works that assume known system dynamics, by carefully constructing the date sets and redesigning the loss functions of D^2SPO, a provably safe CBF is learned for black-box dynamical systems, which continuously evolves for improved system safety as RL interacts with the environment. To deal with complex task objectives, we take advantage of the capability of LTL in representing the task progress and develop LTL-guided RL policy for efficient completion of various tasks with LTL objectives. Extensive numerical and experimental studies demonstrate that D^2SPO outperforms most state-of-the-art baselines and can achieve over 95% safety rate and nearly 100% task completion rates. The experiment video is available at https://youtu.be/2RgaH-zcmkY.

Problem Formulation

Consider a black-box dynamical system

$$\dot{s}(t) = f(s(t), u(t)), \quad s(0) \in \mathbb{R}^n \tag{4}$$

where $s(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$ represent the system state and control input, respectively, and the system dynamics f is unknown. The system in (4) is considered as safe if the states are restricted within the set

$$\mathcal{C} := \{s \in \mathbb{R}^n | h(s) \geq 0\}, \tag{5}$$

where $h : \mathbb{R}^n \to \mathbb{R}$ is a twice continuously differentiable function. That is, the set $\mathcal{C}$ models system safety specifications.

Definition 1 (Ames *et al.*, 2017) Let $\mathcal{D}$ be an open set such that $\mathcal{D} \supset \mathcal{C}$. The function $h(s)$ is called a CBF on $\mathcal{D}$ if there exists a locally Lipschitz continuous extended class $\mathcal{K}$ function $\alpha : \mathbb{R} \to \mathbb{R}$ such that for all $s \in \mathcal{D}$

$$\sup_{u \in \mathcal{U}} \{\langle \nabla h(s), f(s,u) \rangle + \alpha(h(s))\} \geq 0, \tag{6}$$

where $\mathcal{U} \in \mathbb{R}^m$ is the control space. The set of inputs induced by the CBF $h(s)$ is defined as

$$\mathcal{J}(s) := \{u \in \mathcal{U} | \langle \nabla h(s), f(s,u) \rangle + \alpha(h(s)) \geq 0\}. \tag{7}$$

The following lemma indicates that for all $s \in \mathcal{D}$, the safety of system (4) can be guaranteed with appropriate control input $u(s) \in \mathcal{J}(s)$.

Lemma 1 (Ames *et al.*, 2017) *Assume that $h(s)$ is a CBF on $\mathcal{D}$* and $u(s) \in \mathcal{J}(s)$ is locally Lipschitz continuous. Then, it holds that $s(t) \in \mathcal{C}$ for all $t \geq 0$ if $s(0) \in \mathcal{C}$.

Lemma 1 indicates, given a valid CBF, the system safety is guaranteed. However, the constructed CBF in (6) builds upon a key assumption that the system dynamics $f(s,u)$ is known. If the system dynamics is unknown, as in this work, significant challenges are imposed. Hence, given a black-box dynamical system and an LTL task ϕ, the goal of this work is to 1) learn the CBF using data collected during run time without knowing system dynamics, and 2) develop safe RL algorithms that synthesize the learned CBF to accomplish the LTL task ϕ.

Example 1 To elaborate this idea, we use a navigation task as a running example throughout this work. Consider a mobile robot navigating in an environment scattered with a set of M obstacles $\{obs_i\}_{i=1}^{M}$. The robot dynamics is unknown as in (4) and it can only detect the obstacles within its sensing range, e.g., a disk area with radius H. The agent is required to navigate among a set of destinations while avoiding collision with obstacles. Such a task is represented as an LTL formula ϕ. Let (x, y) and $\boldsymbol{\Theta}$ denote the position and orientation of robot and let $\mathcal{G}$ denote the geometric safe set.[1] The control inputs are the angular velocity ω and linear velocity v.

[1]The geometric safe set $\mathcal{G}$ is a superset of $\mathcal{C}$, i.e., $\mathcal{C} \subset \mathcal{G}$, that can be directly specified on the state-space of the system (i.e., the free motion region without collisions with obstacles).

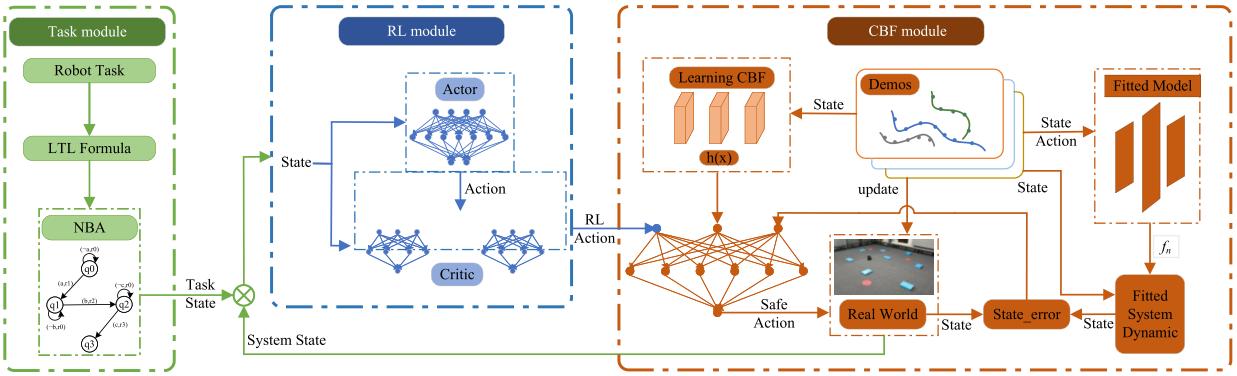


Fig. 1 The framework of D^{2}SPO consists of three main components: the robot task expressed by LTL specification, the learning of CBF for system safety, and the learning of policies for LTL tasks. Using online collected demonstrations of safe and desirable behaviors, the CBF is continuously learned and improved, which is then integrated with the LTL-guided RL policy to achieve safe exploration and robot control.

Data-Driven Safe Policy Optimization

This section presents the Data-driven Safe Policy Optimization (D^{2}SPO) for black-box dynamical systems with safety constraints and LTL task constraints. We first present how the data-set of expert demonstration is constructed in Section "Construction of the data-sets of safe demonstration", based on which an optimization-based approach is developed to facilitate the learning of CBF in Section "Learning of control barrier function". Together with the LTL-guided RL policy developed in Section "Temporal logic guided RL", the D^{2}SPO is finally presented in Section "Safe policy optimization" which generates safe policies for the completion of LTL tasks. The architecture of D^{2}SPO is illustrated in **Fig. 1**.

Construction of the data-sets of safe demonstration

Despite the unknown dynamics, suppose that the states $s(t)$ and $s(t+\Delta t)$ of the system (4) can be sampled. Unlike many existing works with known differentiable dynamics f, due to the considered black-box dynamics, the policy gradient cannot be back-propagated to update the controller in this work. To address this challenge, we consider a differentiable nominal model f_n, which can be established using many existing methods, e.g., fitting a neural network using sampled system states (Forgione and Piga, 2021). Note that f_n does not have to perfectly match the system (4).

Given the current control input $u(t)$ and the sampled system state $s(t)$, the estimated state at $t+\Delta t$ using the nominal model f_n is denoted by

$$s_n(t+\Delta t) = s(t) + f_n(s(t), u(t))\Delta t. \tag{8}$$

Based on (8) and the sampled system state $s(t+\Delta t)$, inspired by Qin *et al.* (2022), we construct

$$s_g(t+\Delta t) = s_n(t+\Delta t) + g(s(t+\Delta t) - s_n(t+\Delta t)) \tag{9}$$

where $g(s) = s$ is an identity function but without gradient. That is, $s_g(t+\Delta t)$ is an error-free estimation of $s(t+\Delta t)$. The requirement of no gradient is to avoid the back-propagation of the its gradient to the argument s.

Based on (8) and (9), a set of N discretized data-point-pairs is constructed as $T := \{(z_i, z_i', O_s)\}_{i=1}^{N}$. To facilitate the following development, we use z_i and z_i' to represent the current state $s(t)$ and the state $s_g(t+\Delta t)$, respectively, and use $O_s = \{obs_i\}_{i=1}^{M'}$ to represent a set of obstacles within the sensing range of the agent at $s(t)$. To learn CBF via collected data, a crucial step is to identify a set of safe states that can be exploited to learn the CBF. Hence, after obtaining the data-point-pairs T, the next is to construct the safe demonstration P_{exp} from T. Specifically, we propose $b(z_i, O_s)$, based on which we construct the set of data-point-pairs $P_{exp} := \{(x_i, x_i') | x_i \geq d\}_{i=1}^{N_1}$., where $x_i = b(z_i, O_s)$, $x_i' = b(z_i', O_s)$, $\forall (z_i, z_i', O_s) \in T$ and $d \in \mathbb{R}^+$ is a desired safety threshold. Note that P_{exp} is defined based on the relative distance, which improves the response to the obstacles. In the following sections, we will show how P_{exp} can be exploited to learn the CBF. It is worth pointing out that P_{exp} is online updated during the runtime, which enables continuous improvement of CBF using the newly generated date by RL.

Learning of control barrier function

In this section, we present how the CBF is learned from P_{exp} without knowing the system dynamics. We first define regions $D' := \cup_{i=1}^{N_1} B_{\varepsilon,p}(x_i), \forall x_i \in P_{exp}$, and $D := D' \backslash bd(D')$, where $\mathcal{B}_{\varepsilon,p}(x_i) := \{x \in \mathbb{R}^n | \|x - x_i\|_p \leq \varepsilon\}$ denotes the closed p-norm ball centered at $x_i \in \mathbb{R}^n$ with radius ε and $bd(D')$ denotes the boundary of D'. By adjusting ε or ignoring the sampled data near the boundary of D, $D \subseteq \mathcal{G}$ can be ensured. Then, we construct the set[2]

$$\mathcal{L} \triangleq \{bd(D) \oplus B_{\sigma,p}(0)\} \backslash D,$$

[2]For two sets $\mathcal{D}_1$ and $\mathcal{D}_2$, the Minkowski sum is defined as $\mathcal{D}_1 \oplus \mathcal{D}_2 := \{x_1 + x_2 \in \mathbb{R}^n | x_1 \in \mathcal{D}_1, x_2 \in \mathcal{D}_2\}$.

which can be considered as a ring with width σ around D. The idea behind is to define regions such that $h(x_i) \geq 0$ for $x_i \in D$ and $h(x_i) < 0$ for $x_i \in \mathcal{L}$ so that CBF can be learned. Based on D, the safe data-set is denoted as $X_s = \{x_i | (x_i, x_i') \in P_{exp}\}$. Similarly, to get unsafe data-set, $X_{\mathcal{L}} = \{x_i\}_{i=0}^{N_2}$ is constructed by sampling from $\mathcal{L}$. The set X_s and $X_{\mathcal{L}}$ form the ε-net and $\bar{\varepsilon}$-nets on D and $\mathcal{L}$, respectively, where the ε-net indicates that for any $x \in D$ there exists $x_i \in X_s$ such that $\|x - x_i\|_p \leq \varepsilon$.

Consider a twice continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ with the local Lipschitz bound

$$L_h(x) \triangleq \sup_{x_1, x_2 \in \mathcal{B}_{\varepsilon,p}(x)} \frac{|h(x_1) - h(x_2)|}{\|x_1 - x_2\|_p}.$$

To ensure that the learned CBF is valid with respect to Def. 1, we have the following constraints.

Proposition 1 *Suppose $h(x)$ is Lipchitz continuous with constant $L_h(x)$.* It holds that $h(x) < 0$ for all $x \in \mathcal{L}$, if

$$h(x_i) \leq -\gamma_u, \quad \forall x_i \in X_{\mathcal{L}},$$

where $X_{\mathcal{L}}$ is an $\bar{\varepsilon}$-net of $\mathcal{L}$ with $\bar{\varepsilon} < \gamma_u / L_h(x_i)$.

The proof is given in appendix. After finding the constraint on $\bar{\varepsilon}$, we consider a function that finds the condition for ε.

Proposition 2 *Denote by $e(x) := \dot{h}(x) + \alpha(h(x))$ and suppose $e(x)$* is Lipchitz continuous with constant $L_e(x)$. It holds that $e(x) \geq 0$ for all $x \in D$, if

$$e(x_i) \geq \gamma_{exp}, \quad \forall x_i \in X_s,$$

where X_s is an ε-net of D with $\varepsilon \leq \gamma_{exp} / L_e(x_i)$.

The proof is given in appendix. Based on the Propositions 1 and 2, the CBF can be learned from data by solving the following optimization problem:

$$\begin{aligned}
&\min \|h\| \\
&\text{s.t.} \, h(x_i) \geq \gamma_s, \quad \forall x_i \in X_s, \\
&h(x_i) \leq -\gamma_u, \quad \forall x_i \in X_{\mathcal{L}}, \\
&Lip(h(x_i), \bar{\varepsilon}) \leq L_h, \quad \forall x_i \in X_{\mathcal{L}}, \\
&e(x_i) \geq \gamma_{exp}, \quad \forall x_i \in X_s, \\
&Lip(e(x_i), \varepsilon) \leq L_e, \quad \forall (x_i, x_i') \in \mathcal{P}_{exp}
\end{aligned} \tag{10}$$

In (10), $Lip(\cdot, \varepsilon)$ indicates the upper bound of the Lipschitz constant given its argument is in an ε-neighborhood, and the hyper-parameters γ_s, γ_u, γ_{exp}, L_h and L_e are positive constants determined by the ε and $\bar{\varepsilon}$-nets using the data from the data-sets X_s and $X_{\mathcal{L}}$. The first and second constraints ensure that $h(x) > 0$ if x is in the safe set X_s and $h(x) < 0$ if x is in the unsafe set $X_{\mathcal{L}}$. The fourth constraint ensures the derivative condition in (6). Although the learning of CBF can be well formulated as an optimization problem as in (10), it cannot be solved as $e(x)$ contains $\dot{h}(x)$, which is not available due to black-box dynamics. We will show in Section "Safe policy optimization" how this issued can be addressed.

Temporal logic guided RL

Due to the consideration of black-box dynamical systems, model-free RL is employed to search for a policy π_{rl} to complete the LTL task ϕ. Temporal logic guided RL (TLGRL) is developed in this work, which takes advantage of the capability of LTL in representing the task progress to design the reward function of RL so that π_{rl} can be efficiently learned. Specifically, given the state z_{t-1} at time $t-1$ and the state z_t after applying the action a_t, let $q = L(z_{t-1})$ and $q' = L(z_t)$ denote the automaton states of B_ϕ as defined in Section "Linear Temporal Logic", where L maps the state z to an automaton state. Since B_ϕ is a graph, for any given $q \in Q$, graph search methods can be employed to determine a node path that starts from q and ends at the accepting set. We denote by $Prog(q)$ by such a node sequence that excludes q. The reward function is designed as

$$r(z_{t-1}, a_1, z_t) = \begin{cases} -c_n, & \text{if } q' = q \\ c_p, & \text{if } q' \in Prog(q) \\ r_{goal}, & \text{if } q' \in F \end{cases} \tag{11}$$

where $c_n, r_{goal}, c_p \in \mathbb{R}^+$. If $q = q'$, i.e., no mission progress occurs, a negative reward $-c_n$ will be given to motivate transitions to the next state in B_ϕ, as less reward is obtained if more steps are spent in the current state. If the state transits towards the accepting set F, i.e. $q' \in Prog(q)$, a positive reward c_p will be given. Once the task completes, i.e., $q' \in F$, a large reward r_{goal} will be received.

During training, the completion of a sub-task (e.g., visiting a destination) is evaluated in the form of $k_1 d_s \cdot e^{k_2 I_{ep}}$. Given a predetermined threshold $\bar{d}$, the subtask is considered as completed if $k_1 d_s \cdot e^{k_2 I_{ep}} \leq \bar{d}$ (i.e., the robot reaches the proximity of the destination). In the early stage of training where I_{ep} is small, a large d_s is allowed to relax the mission completion requirement. As training continues where I_{ep} becomes larger, smaller d_s is enforced to meet the mission completion requirement (i.e., the robot has to be closer to the destination to be considered as mission completion). Such a design, together with the progressive rewards, can improve training efficiency, as the robot can complete the task quickly in the early training to reducing unnecessary exploration.

With the designed reward function r, RL algorithm is used to solve the proposed problem. Due to the consideration of continuous state and action spaces, the Twin Delayed Deep Deterministic Policy Gradient algorithm (TD3) from Fujimoto *et al.* (2018) is adapted in this work. The TD3 algorithm can mitigate the issue of overestimation of Q values by double networks and reduce the correlation of preceding and following actions using the experience buffer. In TD3, there are two critic networks (Twin) representing Q_1 and Q_2 values, which are parameterized by θ_1 and θ_2, respectively. The target networks are parameterized by θ_1' and θ_2', respectively, with the relatively smaller one serving as the target Q network, i.e.,

$$y = r + \gamma \min_{i=1,2} Q_{\theta_i'}(z', \widetilde{a}),$$

where r is the reward designed in (11). The actor network is parameterized by ϕ and the control action a is selected according to the policy network π_ϕ. After applying a, the reward r and new state z' are obtained. Then, at time step t, the agent observes the current state z_t and takes action a_t. The transition tuple (z, a, r, z') is stored in the replay buffer. To enable adequate exploration and smoothing regularization, clipped Gaussian noise is added to the action as

$$\widetilde{a} = \pi_{\phi'}(z') + \xi, \qquad \xi \sim clip(\mathcal{N}(0, \widetilde{\sigma}), -c, c).$$

Randomly sampling mini-batch of T transitions from the replay buffer, the critics are updated by following

$$\theta_i = \text{argmin}_{\theta_i} \frac{1}{T} \sum (y - Q_{\theta_i}(z, a))^2, i = 1, 2.$$

The policy is optimized by following the gradient

$$\nabla_a Q_{\theta_1}(z, a)|_{a=\pi_\phi(z)} \nabla_\phi \pi_\phi(z).$$

Safe policy optimization

An unconstrained relaxation of problem (10) is proposed in this section which can be solved efficiently in practice by first order gradient based methods. The idea is outlined in Alg. 1, which consists of an outer loop and an inner loop. Since each outer loop has at most 200 inner loop steps, N outer loops has at most $200N$ loop steps, and thus the time complexity of Alg. 1 is $O(N)$.

Specifically, we model joint control policy π and CBF h as neural networks with parameters Ω and $\boldsymbol{\Theta}$, respectively. The unconstrained optimization is then formulated as

$$T_L = \min T(\boldsymbol{\Theta}, \Omega), \tag{12}$$

where

$$\begin{aligned}
T_s^{\boldsymbol{\Theta}} &= \sum_{x_i \in X_s} \max\{\gamma_s - h_{\boldsymbol{\Theta}}(x_i), 0\}, \\
T_u^{\boldsymbol{\Theta}} &= \sum_{x_i \in X_{\mathcal{L}}} \max\{\gamma_u + h_{\boldsymbol{\Theta}}(x_i), 0\}, \\
T_e^{\boldsymbol{\Theta},\Omega} &= \sum_{(x_i, x_i') \in P_{exp}} \max\left\{\gamma_{exp} - \frac{h_{\boldsymbol{\Theta}}(x_i') - h_{\boldsymbol{\Theta}}(x_i)}{\Delta t} - \alpha(h_{\boldsymbol{\Theta}}(x_i)), 0\right\}, \\
T_g^{\Omega} &= \sum_{x_i \in X_s, z_i \in Z_{exp}} \|\pi_\Omega(x_i) - \pi_{rl}(z_i)\|_2^2,
\end{aligned}$$

where $T(\boldsymbol{\Theta}, \Omega) \triangleq \|\boldsymbol{\Theta}\|^2 + \lambda_s T_s^{\boldsymbol{\Theta}} + \lambda_u T_u^{\boldsymbol{\Theta}} + \lambda_e T_e^{\boldsymbol{\Theta},\Omega} + \lambda_g T_g^{\Omega}$. and $\lambda_s, \lambda_u, \lambda_e, \lambda_g \in \mathbb{R}^+$ are tuning parameters indicating the relative importance to balance the goal-reaching and safety objectives. In (12), $T_s^{\boldsymbol{\Theta}}$ and $T_u^{\boldsymbol{\Theta}}$ are consistent with the first two constraints in (10). T_g indicates that the joint control policy π_Ω should be close to the RL policy π_{rl} generated in Section "Temporal logic guided RL" to maximally complete the LTL task ϕ. Since f is unknown in (4), the gradient cannot be back-propagated to Ω. Instead, we approximate $\dot{h}(x)$ by $(h(x') - h(x))/\Delta t$, where x and x' are defined in Section "Construction of the data-sets of safe demonstration". It is worth pointing out that the data-sets are not fixed during training and will be periodically updated with new samples during the runtime by the current controller.

The following theorem shows that $T_e^{\boldsymbol{\Theta},\Omega}$ is a valid candidate to learn the CBF.

Theorem 1 *$T_e^{\boldsymbol{\Theta},\Omega}$ is differentiable w.r.t. the parameter Ω, and when $\Delta t \to 0$, this is an error-free approximation.*

Proof When the fitted model is differentiable, s_n and s_g in (9) are differentiable w.s.t. π, T_e is also differentiable w.s.t. π and its parameter Ω. Thus, $\nabla_\Omega T_e$ exists. Since $s_g(t + \Delta t)$ is an error-free estimation of $s(t + \Delta t)$, $\dot{h}(x)$ is an error-free approximation when $\Delta t \to 0$, thus T_e is an error-free approximation when $\Delta t \to 0$. □

Algorithm 1 Safe Policy Optimization

for Black-Box Systems

1. **Input:** State sets X_s, $X_{\mathcal{L}}$ and set P_{exp}, loss function $T_L(\Omega, \boldsymbol{\Theta})$, training iterations N and episode steps E.

2. **Initial:** controller π with initial random parameters Ω, CBF h with parameters $\boldsymbol{\Theta}$. Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network with random parameters θ_1, θ_2, ϕ. Initialize target networks $\theta_1' \leftarrow \theta_1$, $\theta_2' \leftarrow \theta_2$, $\phi' \leftarrow \phi$. Initialize replay buffer.
3. **for** i=0,... *N***do**
4. Initialize Trajectory data-set $\mathcal{D} \leftarrow \varnothing$
5. *Getz*(0)
6. 6: **for** j=0,... *E***do**
7. 7: $\pi_{rl}(z) \leftarrow \pi(z;\phi)$
8. $\mathcal{D}_j \leftarrow TRAJECTORY(\pi(\cdot;\boldsymbol{\Omega}))$
9. Store transition tuple $(z, \pi(\cdot;\boldsymbol{\Omega}), r, z')$
10. $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_j$
11. $X_{\mathcal{L}}, X_s, P_{exp} \leftarrow UPDATE(\mathcal{D})$
12. **if** i mod d **then**
13. Update θ_i
14. Update ϕ, θ_i', ϕ'
15. **end if**
16. **end for**
17. $\boldsymbol{\Omega}, \boldsymbol{\Theta} \leftarrow UPDATE(\boldsymbol{\Omega}, \boldsymbol{\Theta}, X_s, X_{\mathcal{L}}, P_{exp}, T_L)$
18. **end for**
19. **return** parameter Ω of π and parameter $\boldsymbol{\Theta}$ of h

Simulations and Experiment

Numerical simulation and physical experiments are carried out to evaluate the performance of D^{2}SPO in terms of 1) the effectiveness of the learned CBF, 2) the learning efficiency of TLGRL in completing desired tasks, and 3) the performance in physical experiment.

Experiment setup

Consider a workspace consisting of 4 areas of interest and 8 obstacles scattered in the environment. The agent is required to monitor the workspace by sequentially visiting the areas of interest while avoiding obstacles. We consider two tasks of different complexities:

$$\begin{aligned}\phi_1 &= ((\neg A_3)\mathrm{U}A_2) \wedge ((\neg A_2)\mathrm{U}A_1) \wedge (\Diamond A_1),\\ \phi_2 &= ((A_1 \wedge (\bigcirc A_3)) \vee (A_3 \wedge (\bigcirc A_1)))\\ &\quad \wedge (\bigcirc(\Diamond(A_4 \wedge (\bigcirc(\Diamond A_2))))),\end{aligned}$$

where A_i, $i = 1,\ldots 4$, represents the ith area of interest. The task ϕ_1 requires the agent to visit the areas of interest in a given order (i.e., $A_1A_2A_3$). Such an order is relaxed in task ϕ_2 to give the agent more freedom to complete the task (i.e., $A_1A_3A_4A_2$ or $A_3A_1A_4A_2$). In the following analysis, ϕ_1 will be used to evaluate the performance of our approach against the state-of-the-art (SOTA) methods, while both ϕ_1 and ϕ_2 will be used to validate the effectiveness of TLGRL. For comparisons, we consider the following SOTA methods: 1) constrained policy optimization (CPO) (Achiam *et al.*, 2017); 2) reaching constrained reinforcement learning (RCRL) (Yu *et al.*, 2022), and 3) a set of variants of Lagrangian methods (Tessler *et al.*, 2018) with mainstream RL algorithm, i.e., LPPO (PPO-Lagrangian), LTD3 (TD3-Lagrangian), and LSAC (SAC-Lagrangian).

Indicator definitions

These methods, as well as our approach, are trained for 20,000 episodes with a maximum of 200 steps per episode under the same task and environment. The safety rate is defined as the ratio of the number of episodes that the agent does not hit the obstacle to the total episode number. The task completion rate is defined similarly as the ratio of the number of episodes that the agent completes the task to the total episode number.

Main results

Table 1 Performance of different methods for ϕ_1

Method	*Safety rate*	*Task completion rate*
CPO (Achiam *et al.*, 2017)	90.71%	19.44%
LPPO (Tessler *et al.*, 2018)	96.84%	0%
LTD3	83.08%	0%
LSAC	77.76%	96.74%
RCRL (Yu *et al.*, 2022)	74.47%	77.52%
D^{2}SPO (Ours)	**95.23%**	**99.31%**

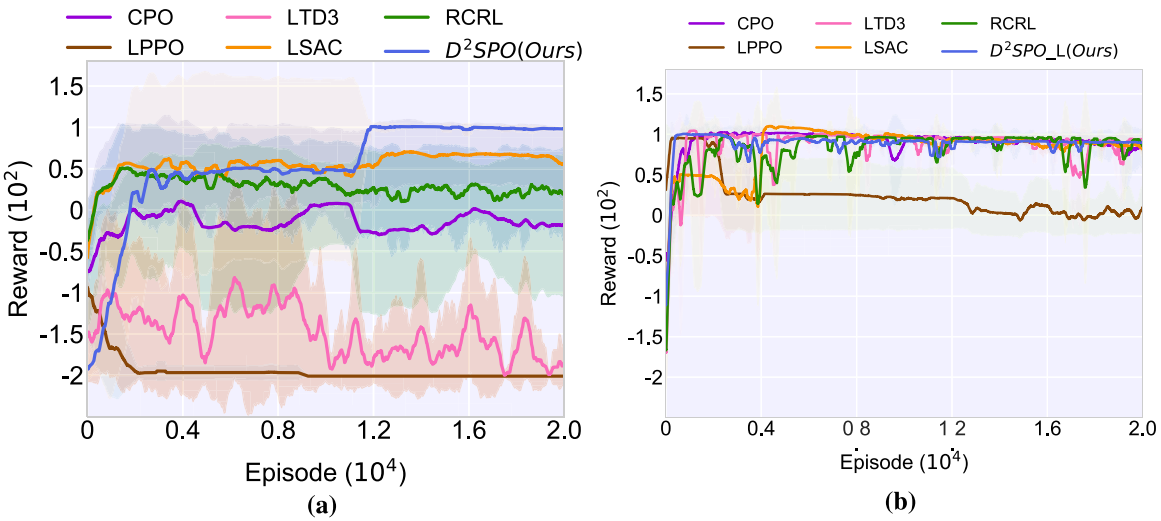


Fig. 2 (a) The evolution of reward function for ϕ_1. (b) The evolution of reward for the relaxed ϕ_1 with fewer obstacles and destinations.

Table 2 Performance of different methods for relaxed ϕ_1

Method	*Safety rate*	*Task completion rate*
CPO (Achiam *et al.*, 2017)	94.49%	98.02%
LPPO (Tessler *et al.*, 2018)	91.59%	63.50%
LTD3	95.10%	97.11%
LSAC	93.94%	98.75%
RCRL (Yu *et al.*, 2022)	94.19%	94.82%
D²SPO (Ours)	**97.94%**	**99.48%**

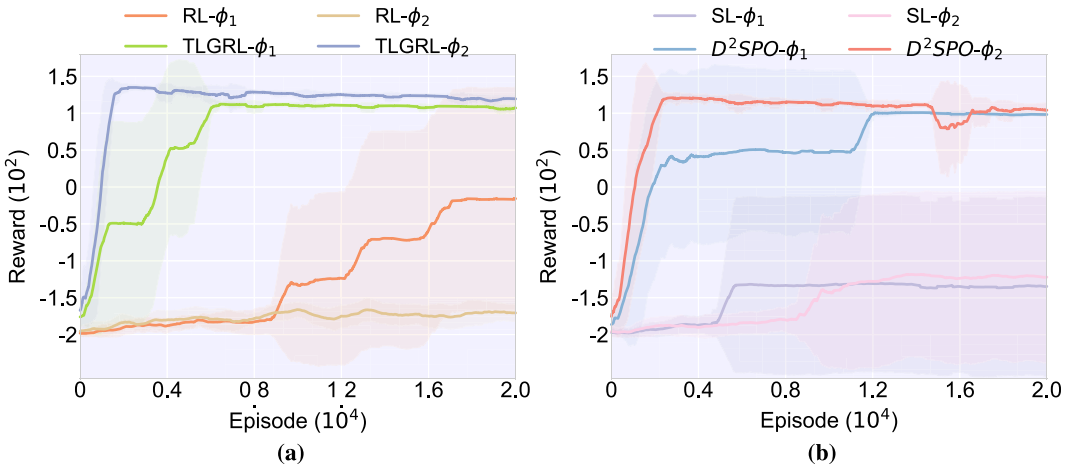


Fig. 3 The performance of TLGRL when completing different level tasks. (a) is the reward curves for each method on complex cruising missions, and (b) display the performance of high-freedom cruising mission.

The simulation results are shown in **Table 1** and **Fig. 2**a, where the safety rate and the task completion rate for the last 2000 episodes are reported. The results indicate that the SOTA methods either fail to accomplish the specified task or violate the safety constraints. This is due to the difficulty of CPO in guaranteeing safety during the training process. The Lagrangian-based safe RL methods (LPPO, LTD3 and LSAC) aim to find a balance between accomplishing tasks and satisfying safety constraints, and it is found that the Lagrangian-based methods are very sensitive to the initial values of the Lagrange multipliers. Since the parameters in

the multipliers and RL are updated alternately and can converge to a saddle point, its performance varies considerably with different RL algorithms. In our experiments, PPO and TD3 are more inclined to satisfy the safety constraints under the same parameter conditions, but this also greatly limits their exploration performance, making it difficult to complete the task. It can be seen that SAC has a high task success rate, which is due to the fact that SAC obtains superior exploration performance by maximizing the entropy of the action distribution in each state compared to PPO and TD3. In contrast, our method guarantees training and deployment safety through the learned CBF, while guaranteeing task completion through temporal logic guided RL algorithm, and thus outperforms these baselines in terms of task completion and safety guarantees.

To show the performance on simple tasks, we consider a relaxed task that simplifies ϕ_1 by considering the same navigation task but only with 1 obstacle and 2 areas of interest. As indicated in **Table 2** and **Fig. 2**(b), for relatively simple tasks, these SOTA methods can yield satisfying performance in terms of mission completion and safety guarantees, although our approach still outperforms them. However, if more complex tasks are considered as in **Table 1**, their performance degrades significantly while our approach still shows promising performance. In summary, the SOTA methods might work for relative simple tasks, but in general can not yield satisfactory performance for complex tasks. In contrast, our approach works for both simple and complex tasks and outperforms SOTA methods in terms of mission completion and safety guarantees.

To show the learning efficiency of TLGRL, we first consider the environment without obstacles. The ϕ_1 and ϕ_2 are both present to show the benefit of our method. RL-ϕ_1 and RL-ϕ_2 mean that the agent performs ϕ_1 and ϕ_2 using the method in Fujimoto *et al.* (2018), respectively. The performance of RL-ϕ_1, RL-ϕ_2, TLGRL-ϕ_1, and TLGRL-ϕ_2 are shown in **Fig. 3** (a). Clearly, TLGRL significantly improves the speed of convergence and the value of the reward compared to basic RL, which means temporal logic actually improves the learning efficiency in RL. Then, the safety-concerned environment is incorporated to show whether the approach in the learning process is still effective or not. Let SL denote the variant of D^{2}SPO without LTL guided progressive reward. The performance of SL-ϕ_1, SL-ϕ_2, D^{2}SPO-ϕ_1, and D^{2}SPO-ϕ_2 are presented in **Fig. 3**(b). D^{2}SPO obtains far more return than SL, which is mainly due to the progression reward in (11) that guides the learning towards mission completion.

Physical Experiments

Physical experiments are carried out in this section, where the turtlebot with limited sensing capability, e.g., only detect obstacles within a certain radius, is used. To demonstrate the capability of learning a valid CBF via interactions with the environment, we consider 4 cases with different deployment of obstacles and destinations. TD3 is employed for the turtlebot to explore the environment, which can also be replaced by other RL algorithms. **Fig. 4**a–d show the learned CBF respectively while **Fig. 4**(e)–h show the corresponding contours. Note that the areas below 0 may be connected if the randomly generated obstacles are too close.

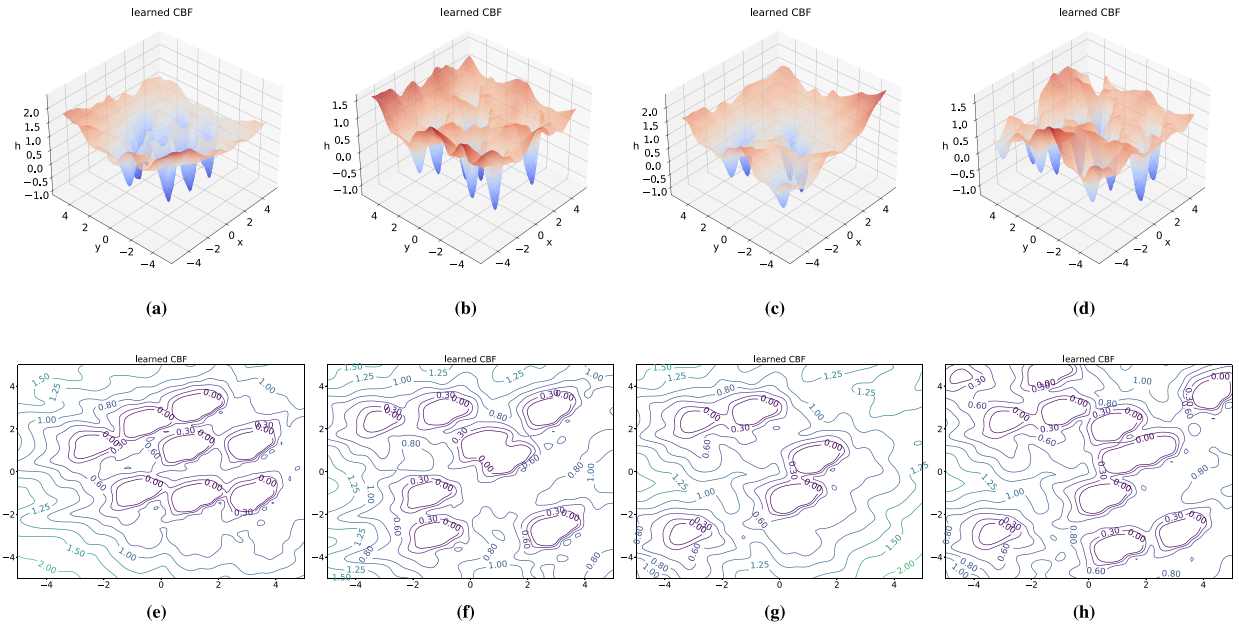


Fig. 4 (a)–(d) are the surface plots of the learned CBF under different quantities and location and (e)–(h) are the corresponding level set plots of the learned CBF.

[3] https://youtu.be/2RgaH-zcmkY

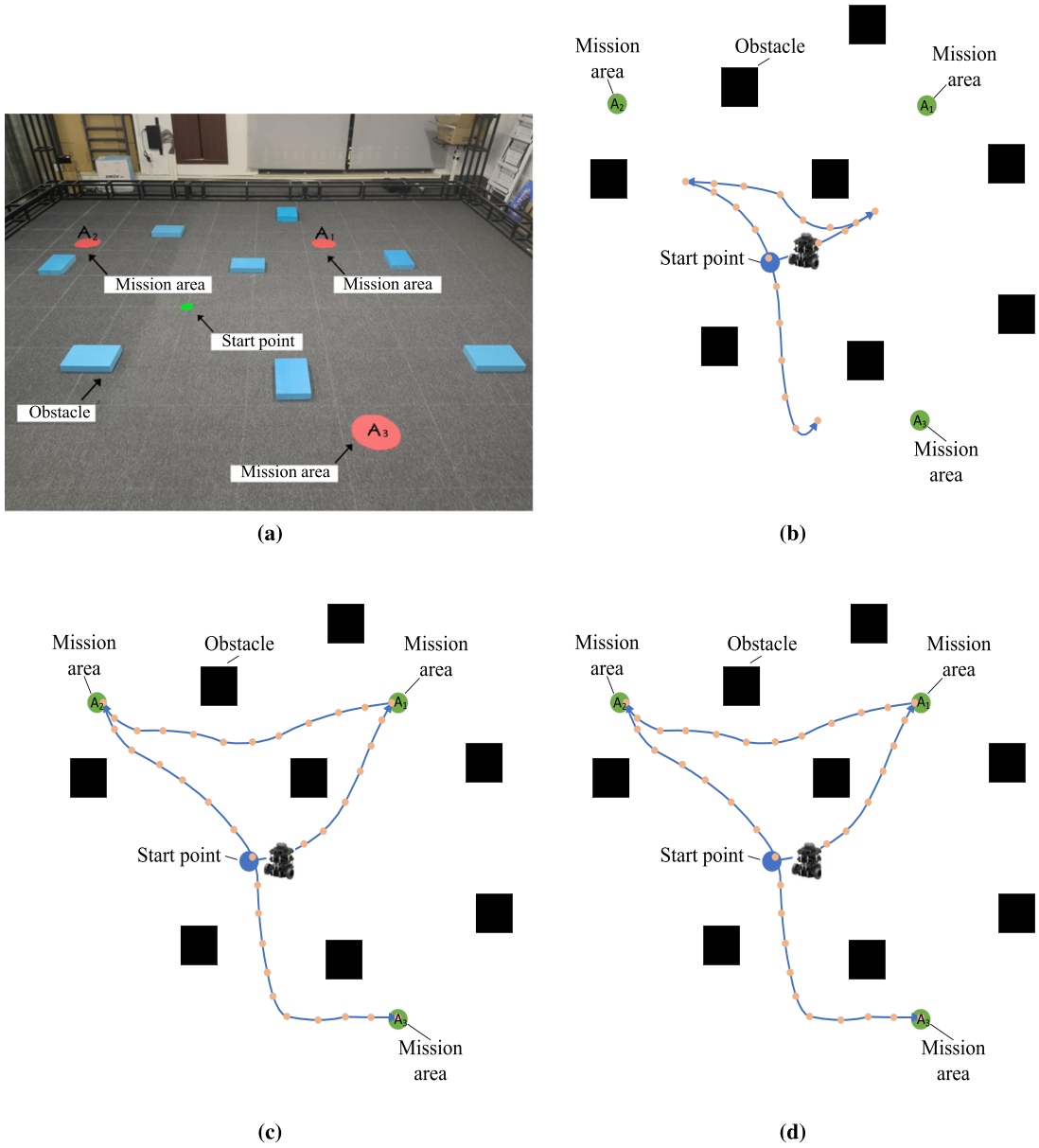


Fig. 5 (a) The physical environment. (b)–(d) The corresponding environment in simulation, where black squares indicate the obstacles, the blue and green dots indicate the initial position of the robot and the areas of interest. The performance after 2000 episodes, 10,000 episodes, and 20,000 episodes of training are shown in (b), (c), and (d), respectively.

One of the 4 physical environments is shown in Fig. 5a, where the blue squares indicate the obstacles, the green square is the initial position of the robot, and red disk are the areas of interest. The turtlebot is tasked to access the given area in the order of $A_1A_2A_3$. In the early training phase, although the mission of visiting areas of interest is not completed yet, the robot can successfully avoid obstacles, which demonstrates the safety ensurance during exploring. As more training is carried out, the robot finally completes the task with collision avoidance. The experiment video with more explanations is provided.[3]

Conclusion

The Data-driven Safe Policy Optimization for black-box dynamical systems is developed in this section, which jointly learns a control barrier function for system safety and a linear temporal logic guided RL algorithm for complex task objectives. Extensive numerical simulation shows that D^{2}SPO outperforms state-of-the-art baselines. The effectiveness of D^{2}SPO is also verified in physical environments.

Appendix

The proof of proposition 3

Proof For any $x \in \mathcal{L}$, there exists $x_i \in X_{\mathcal{L}}$ satisfying $\|x - x_i\| \leq \bar{\varepsilon} < \gamma_u / L_h(x_i)$, we have

$$\begin{aligned} h(x) &= h(x) - h(x_i) + h(x_i) \leq |h(x) - h(x_i)| - \gamma_u \\ &\leq L_h(x_i)\|x - x_i\|_p - \gamma_u \leq L_h(x_i)\bar{\varepsilon} - \gamma_u < 0 \end{aligned}$$

The proof of proposition 4

Proof For any $x \in D$, there exists $x_i \in X_s$ satisfying $\|x - x_i\| \leq \varepsilon \leq \gamma_{exp} / L_e(x_i)$, we have

$$\begin{aligned} e(x) &= e(x) - e(x_i) + e(x_i) \geq \gamma_{exp} - |e(x) - e(x_i)| \\ &\geq \gamma_{exp} - L_e(x_i)\|x - x_i\|_p \geq \gamma_{exp} - L_e(x_i)\varepsilon \geq 0 \end{aligned}$$

Projection-Based Fast and Safe Policy Optimization for Reinforcement Learning

This section proposes a new Fast and Safe Policy Optimization (FSPO) algorithm, which consists of three steps: the first step involves reward improvement update, the second step projects the policy to the neighborhood of the baseline policy to accelerate the optimization process, and the third step addresses the constraint violation by projecting the policy back onto the constraint set. Such a projection-based optimization can improve the convergence and learning performance. Unlike many existing works that require convex approximations for the objectives and constraints, this work exploits a first-order method to avoid expensive computations and high dimensional issues, enabling fast and safe policy optimization, especially for challenging tasks. Numerical simulation and physical experiments demonstrate that FSPO outperforms existing methods in terms of safety guarantees and task completion rate.

Problem Formulation

We define the reward advantage function as $A_\pi^R(s,a) \doteq Q_\pi^R(s,a) - V_\pi^R(s)$, where $Q_\pi^R(s,a) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t,a_t) | s_0 = s, a_0 = a\right]$ is the expected reward following policy π starting from state s and action a, and $V_\pi^R(s) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t,a_t) | s_0 = s\right]$ is the expected reward from state s under policy π. Similarly, we can define the cost advantage function:

$$A_\pi^C(s,a) \doteq Q_\pi^C(s,a) - V_\pi^C(s),$$

where $Q_\pi^C(s,a) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t C(s_t,a_t) | s_0 = s, a_0 = a\right]$ and $V_\pi^C(s) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t C(s_t,a_t) | s_0 = s\right]$.

Let $P(s = s_t|\pi)$ denote the state distribution at time t under policy π. The discounted state distribution induced by π is $d^\pi(s) \doteq (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \mu_t(s|\pi)$. Then the performance difference between two policies π and π' can be compactly expressed as (Kakade and Langford, 2002)

$$J^R\left(\pi'\right) - J^R(\pi) = \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s\sim d^{\pi'} \\ a\sim\pi'}} \left[A_\pi^R(s,a)\right]. \tag{13}$$

Similarly, we have

$$J^C\left(\pi'\right) - J^C(\pi) = \frac{1}{1-\gamma} \mathop{\mathbb{E}}_{\substack{s\sim d^{\pi'} \\ a\sim\pi'}} \left[A_\pi^C(s,a)\right]. \tag{14}$$

The policy can be safely learned by (Achiam *et al.*, 2017):

$$\pi_{k+1} = \arg\max_{\pi\varepsilon\Pi_\theta} \mathop{\mathrm{E}}_{\substack{s\sim d^{\pi_k} \\ a\sim\pi}} \left[A_{\pi_k}^R(s,a)\right]$$

$$\text{s.t. } J^C(\pi_k) + \frac{1}{1-\gamma} \mathop{\mathrm{E}}_{\substack{s\sim d^{\pi_k} \\ a\sim\pi}} \left[A_{\pi_k}^C(s,a)\right] \leq d_i \quad \forall i$$

$$\overline{D}_{KL}(\pi\|\pi_k) \leq \delta. \tag{15}$$

where $\overline{D}_{KL}$ is the KL-divergence between two policies.

Given a safe reinforcement learning problem formulated in (15), we denote by π_B a baseline policy (i.e., a pre-training policy that can be obtained by any reinforcement algorithm). The distance between $\pi(s)$ and $\pi_B(s)$ can be measured by KL-divergence as $D(s) \doteq D_{\mathrm{KL}}(\pi(s)\|\pi_B(s))$. Similarly, we can define $J^D(\pi) \doteq \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t D(s_t)\right]$ and

$$J^D\left(\pi^{'}\right) - J^D(\pi) = \frac{1}{1-\gamma}\mathop{\mathbb{E}}_{\substack{s\sim d^{\pi^{'}}\\ a\sim\pi^{'}}}\left[A^D_{\pi}(s,a)\right]. \tag{16}$$

Using the reward machine and KL-divergence between the learning policy and baseline policy, the training process can be improved since more information is utilized. Hence, the goal is to solve the following optimization problem:

$$\begin{aligned}
\pi_{k+1} = \arg\max_{\pi\in\Pi_\theta} & \mathop{\mathrm{E}}_{\substack{s\sim d^{\pi_k}\\ a\sim\pi}}\left[A^{R^{'}}_{\pi_k}(s,a)\right] \\
\text{s.t.}\ \ J^C(\pi_k) + \frac{1}{1-\gamma}&\mathop{\mathrm{E}}_{\substack{s\sim d^{\pi_k}\\ a\sim\pi}}\left[A^C_{\pi_k}(s,a)\right] \le d_i \quad \forall i \\
J^D(\pi_k) + \frac{1}{1-\gamma}&\mathop{\mathrm{E}}_{\substack{s\sim d^{\pi_k}\\ a\sim\pi}}\left[A^D_{\pi_k}(s,a)\right] \le d_i \quad \forall i \\
\overline{D}_{KL}(\pi\|\pi_k) &\le \delta.
\end{aligned} \tag{17}$$

where $R^{'}$ is the reward function generated by the reward machine.

Fast and Safe Policy Optimization

Approach overview

Inspired by Yang *et al.* (2021), we use three steps to facilitate the training of the policy, i.e., the reward improvement step, the divergence projection step, and the constraints projection step to solve the optimization problem in (17). To avoid residual errors due to the convex approximation, our approach is based on the dual methods. Unlike existing methods such as RCPO (Tessler *et al.*, 2018) that uses dual methods to mix the constrain and the reward, in this work the dual method is used after policy optimization.

As outlined in Alg. 2, π_0 is an initial policy to be updated and π_B is a pre-trained baseline policy that can be obtained by any reinforcement algorithm such as PPO (Schulman *et al.*, 2017), SAC (Haarnoja *et al.*, 2018). For each $k = 0, 1, \ldots$, after sampling trajectories, we first maximize the reward in the trust region by (18) or (19) to get $\pi_{\theta_{k+\frac{1}{3}}}$ in line 5. Then, to accelerate the learning process, we project $\pi_{\theta_{k+\frac{1}{3}}}$ onto the region around π_B using (22), (23) and (24) to obtain $\pi_{\theta_{k+\frac{2}{3}}}$. The Lagrangian coefficients ν_k can be update by (30) or (31). Finally, to satisfy the safety constraints, we project $\pi_{\theta_{k+\frac{2}{3}}}$ to the constrain set by (27), (28) and (29), update μ_k by (31) or (33), and output $\pi_{\theta_{k+1}}$ at last.

Algorithm 2 FSPO

1. Input: (An initial policy $\pi_0 = \pi(\cdot|\theta_0)$, the initial Lagrangian coefficient μ_0 and ν_0, a baseline policy π_B and a trajectory buffer $\mathcal{B}$)
2. Output: the optimal policy θ^*
3. **while** Reward A^R and cost J^C not converged **do**
4. **for** each $k = 0, 1, 2\ldots$ **do**
5. Sample a set of trajectories $D = \tau\sim\pi_k = \pi(\theta_k)$
6. Run Reward Improvement Step by (18) or (19) to obtain $\pi_{\theta_{k+\frac{1}{3}}}$
7. Project the $\pi_{\theta_{k+\frac{1}{3}}}$ to baseline policy π_B neighbor set by (22), (23) and (24) to obtain $\pi_{\theta_{k+\frac{2}{3}}}$
8. update ν_k by (30) or (32)
9. Project the $\pi_{\theta_{k+\frac{2}{3}}}$ to constrain set by (27), (28) and (29) to obtain $\pi_{\theta_{k+1}}$
10. update μ_k by (31) or (33)
11. **end for**
12. obtain θ_{k+1}, ν_k and μ_k, Use $\pi_{\theta_{k+1}}$ in the environment to get new samples
13. **end while**
14. **end procedure**

Three optimization steps

As illustrated in **Fig. 6**, our approach consists of three optimization steps.

Reward Improvement Step: inspired by the reward improvement method in TRPO (Schulman *et al.*, 2015) and PPO (Schulman *et al.*, 2017), we first maximize the reward advantage function $A^R_\pi(s,a)$ with the constraint of Kullback_Leibler (KL)

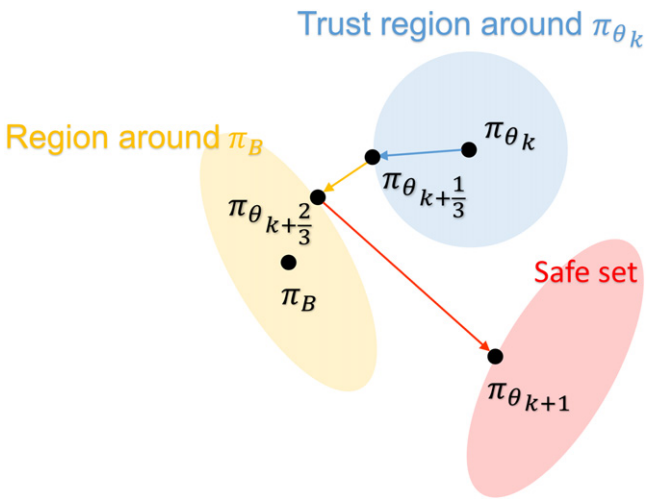


Fig. 6 Three optimization steps of FSPO.

divergence to perform a stable policy update. There are two ways in this step:

$$\pi_{\theta_{k+\frac{1}{3}}} = \arg\max_{\pi_\theta \in \Pi_\theta} \left\{ \mathbb{E}_{s\sim d_{\pi_{\theta_k}}(\cdot), a\sim\pi_\theta(\cdot|s)} \left[A^R_{\pi_{\theta_k}}(s,a) \right] - \alpha_k \sqrt{\mathbb{E}_{s\sim d_{\pi_{\theta_k}}} [KL(\pi_{\theta_k}, \pi_\theta)[s]]} \right\}. \tag{18}$$

or update like PPO with clip as

$$\pi_{\theta_{k+\frac{1}{3}}} \approx \arg\max_{\pi_\theta \in \Pi_\theta} \left\{ \mathbb{E}_{s\sim d_{\pi_{\theta_k}}(\cdot), a\sim\pi_\theta(\cdot|s)} \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^R_{\pi_{\theta_k}}(s_t,a_t), clip\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}, 1-\varepsilon, 1+\varepsilon \right) A^R_{\pi_{\theta_k}}(s_t,a_t) \right) \right\}. \tag{19}$$

Both ways can yield the reward improvement over the neighborhood of the origin policy.

Divergence Projection Step: to accelerate the training process, we leverage the baseline policy π_B and project $\pi_{\theta_{k+\frac{1}{3}}}$ onto the region around π_B. The distance between $\pi_{\theta_{k+\frac{1}{3}}}$ and π_θ is minimized using a distance measure D as

$$\pi_{\theta_{k+\frac{2}{3}}} = \arg\min_{\pi_\theta \in \Pi_\theta} D(\pi_\theta, \pi_{\theta_{k+\frac{1}{3}}}) \tag{20}$$

$$\begin{aligned} &s.t.\ J^D(\pi_{\theta_k}) + \frac{1}{1-\widetilde{\gamma}} \mathbb{E}_{s\sim d_{\pi_{\theta_k}}(\cdot), a\sim\pi_\theta(\cdot|s)} \left[A^D_{\pi_{\theta_k}}(s,a) \right] \\ &+\beta_k \sqrt{\mathbb{E}_{s\sim d_{\pi_{\theta_k}}} [D_{KL}(\pi_{\theta_k}, \pi_\theta)[s]]} \le h^D. \end{aligned} \tag{21}$$

To avoid the convex approximation like CPO and PCPO, the primal-dual approach is developed as

$$(\pi_{\theta_{k+\frac{2}{3}}}, \nu_{k+1}) = \arg\min_{\pi_\theta \in \Pi_\theta} \max_{\nu \ge 0} \hat{L}_D(\pi_\theta, \pi_{\theta_k}, \theta_{k+\frac{1}{3}}), \tag{22}$$

$$\hat{L}_D(\pi_\theta, \pi_{\theta_k}, \theta_{k+\frac{1}{3}}) = D_{KL}(\pi_{\theta_{k+\frac{1}{3}}}, \pi_\theta) + \nu_k \hat{D}(\pi_\theta, \pi_{\theta_k}). \tag{23}$$

The divergence constraint function is

$$\hat{D}(\pi_\theta, \pi_{\theta_k}) = J^D + \frac{1}{1-\widetilde{\gamma}} \mathbb{E}_{s\sim d_{\pi_{\theta_k}}(\cdot), a\sim\pi_\theta(\cdot|s)} A^D_{\pi_{\theta_k}}(s_t,a_t) + \beta_k \sqrt{\mathbb{E}_{s\sim d_{\pi_{\theta_k}}} [D_{KL}(\pi_{\theta_k}, \pi_\theta)[s]]} - h^D. \tag{24}$$

Constrain projection Step: To ensure policy safety, we project $\pi_{\theta_{k+\frac{2}{3}}}$ onto the safe constrain set in this step. Similar to last step, the distance between $\pi_{\theta_{k+\frac{2}{3}}}$ and π_θ is minimized as

$$\pi_{\theta_{k+1}} = \arg\min_{\pi_\theta \in \Pi_\theta} D(\pi_\theta, \pi_{\theta_{k+\frac{2}{3}}}) \tag{25}$$

$$\begin{aligned} &s.t.\ J^C(\pi_{\theta_k}) + \frac{1}{1-\widetilde{\gamma}} \mathbb{E}_{s\sim d_{\pi_{\theta_k}}(\cdot), a\sim\pi_\theta(\cdot|s)} \left[A^C_{\pi_{\theta_k}}(s,a) \right] \\ &+\eta_k \sqrt{\mathbb{E}_{s\sim d_{\pi_{\theta_k}}} [D_{KL}(\pi_{\theta_k}, \pi_\theta)[s]]} \le h^C. \end{aligned} \tag{26}$$

Then we solve the following problem

$$(\pi_{\theta_{k+1}}, \mu_{k+1}) = \arg\min_{\pi_\theta \in \Pi_\theta} \max_{\mu \ge 0} \hat{L}_C(\pi_\theta, \pi_{\theta_k}, \theta_{k+\frac{2}{3}}), \tag{27}$$

$$\hat{L}_C(\pi_\theta, \pi_{\theta_k}, \theta_{k+\frac{2}{3}}) = D_{KL}(\pi_{\theta_{k+\frac{2}{3}}}, \pi_\theta) + \mu_k \hat{C}(\pi_\theta, \pi_{\theta_k}). \tag{28}$$

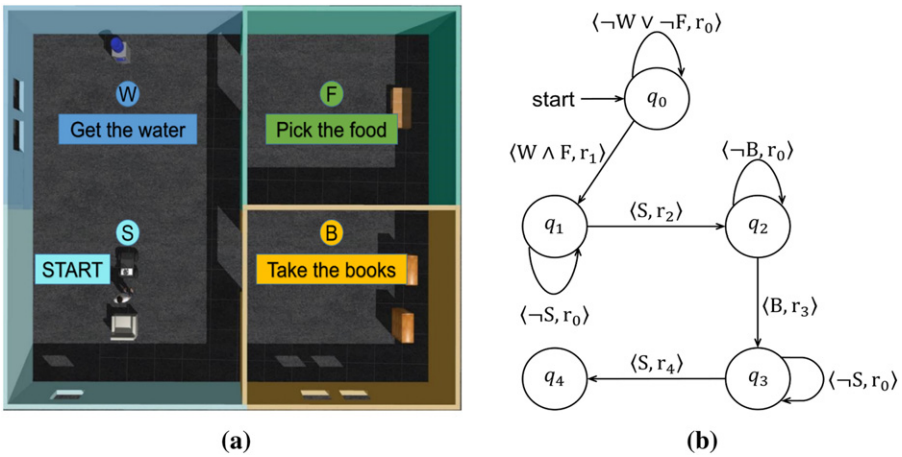


Fig. 7 (a) The simulation environment. (b) The reward machine.

The safety constraint function is

$$\hat{C}(\pi_\theta, \pi_{\theta_k}) = J^C + \frac{1}{1-\widetilde{\gamma}} \cdot \mathbb{E}_{s\sim d_{\pi_{\theta_k}}(\cdot), a\sim\pi_\theta(\cdot|s)} A^C_{\pi_{\theta_k}}(s_t, a_t) + \eta_k \sqrt{\mathbb{E}_{s\sim d_{\pi_{\theta_k}}}[D_{KL}(\pi_{\theta_k}, \pi_\theta)[s]]} - h^C. \tag{29}$$

As for the coefficient v and μ, there are two ways to update:

$$v_{k+1} = v_k + \eta_v (J^D - h_D)_+, \tag{30}$$

$$\mu_{k+1} = \mu_k + \eta_\mu (J^C - h_C)_+, \tag{31}$$

or treat them as parameters of the neural network with the loss functions

$$L_v = -v * \eta_v (J^D - h_D), \tag{32}$$

$$L_\mu = -\mu * \eta_\mu (J^C - h_C). \tag{33}$$

Reward machine

When encountering more challenging tasks (e.g., long-horizon tasks with complex temporal and logic constraints), we encode the task as a linear temporal logic (LTL) formula. Then, we build a reward machine (Icarte *et al.*, 2018) based on the LTL formula to facilitate agent learning.

Definition 3.1 (Reward Machine). Given a tuple $\langle S, A, \mathcal{P}, L\rangle$ where S is a finite set of environment states, A is a finite set of environment states, $\mathcal{P}$ is a finite set of propositional symbols, and L is a labeling function: $L : S \times A \times S \to 2^{\mathcal{P}}$. A reward machine (RM) is a tuple $\langle Q, q_0, \Sigma, \mathcal{R}, \delta, \rho\rangle$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, $\Sigma = 2^{\mathcal{P}}$ is the input alphabet, and $\mathcal{R}$ is a finite set where each $R : S \times A \times S \to \mathbb{R}$ in $\mathcal{R}$ is a reward function.

Example 2 Consider a robot tasked to deliver the water and food to the people, and then pick up the books and send them to the people, as shown in Fig. 7. Such a task can be written in an LTL formula as $((\neg start)\mathrm{U}books) \wedge ((\neg books)\mathrm{U}start) \wedge ((\neg start)\mathrm{U}(water \wedge food)) \wedge (\diamond(water \wedge food))$. A reward machine can be defined over a set of propositional symbols $\mathcal{P} = \{water, food, books, start\}$, where $o \in \mathcal{P}$ occurs when the agent is at location o. The states of reward machine is $Q = \{q_0, q_1, q_2, q_3, q_4\}$ and the reward function set is $\mathcal{R} = \{r_0, r_1, r_2, r_3, r_4\}$ as shown in Fig. 7. The q_0 means the agent haven't got the water and food, and thus the reward is a negative value r_0. If the agent has obtained the water and food, the state of reward machine comes to q_1 and the reward of this transition is r_1. Now the robot needs to send the water and food back to the people. If the agent hasn't go back to S, no matter where it goes, it always stays in state q_1 and gets the negative reward r_0. Once it reaches S, the positive reward r_2 is received, and the RM state comes to q_2. Similarly, only if the agent gets the books, the state can be transited to q_3. The agent then gets the reward r_3 and goes back to the people with state transiting to q_4 to complete the whole task. Otherwise it can only get a negative reward r_0 and stay in q_3 or q_4.

Experiment

In this section, our method is evaluated against recent representative methods. Particularly, we investigate 1) the policy performance: whether our method improves the performance of learned policy in terms of reward collection and convergence; 2) the

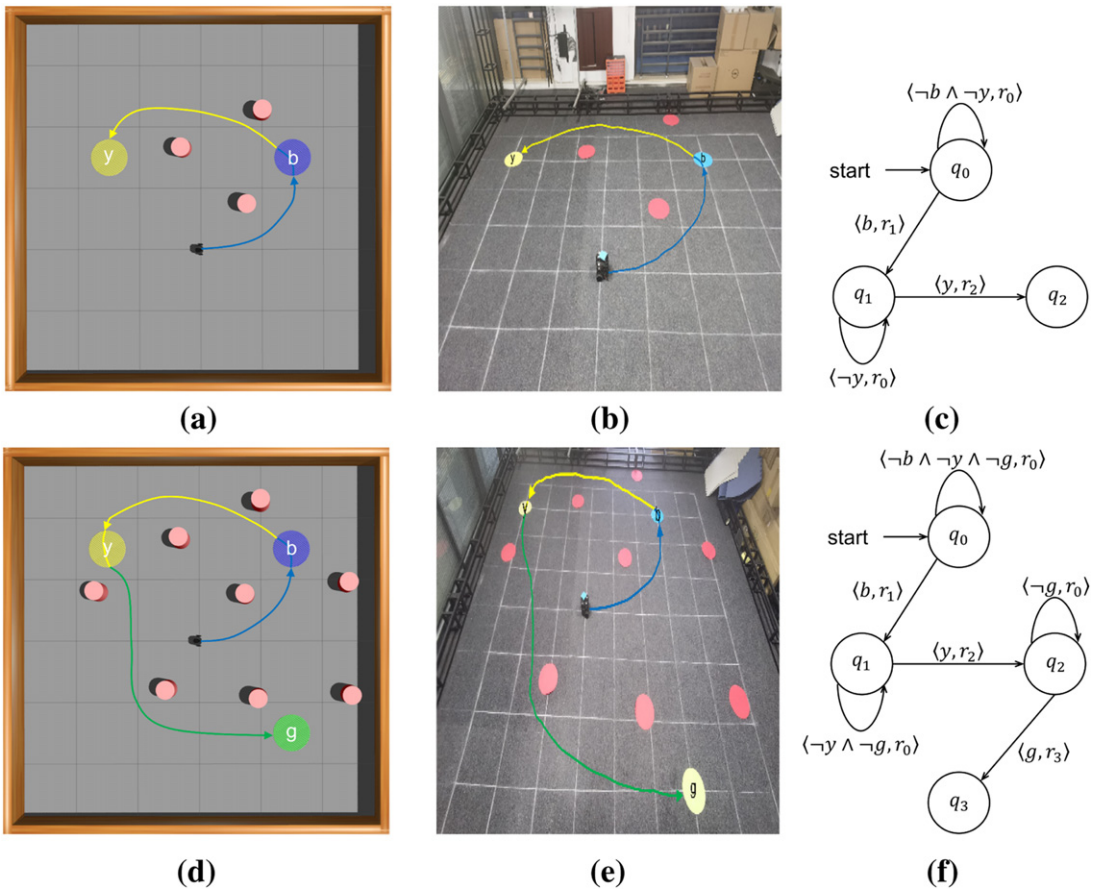


Fig. 8 The top row indicates Case 1 and the bottom row indicates Case 2. (a) The simple environment in simulation. (b) The corresponding physical environment. (c) The reward machine. (d) The challenging environment with more obstacles. (e) The corresponding physical environment. (f) The reward machine.

safety guarantee: whether our method outperforms previous methods in terms of safety guarantee in both training and deployment stages; 3) the task completion: whether our method can complete the required tasks.

Experiment setup

To verify the advantages of our method in different environments, we consider two cases. Case 1 is a relatively simple environment with three obstacles as shown in Fig. 8(a)–(b). The LTL task is $((\neg area_y) \mathrm{U} area_b) \wedge (\diamond area_b)$, which requires the robot to first reach the blue area and then the yellow area. Case 2 is a more challenging environment as shown in Fig. 8(d)–(e), which requires the agent to reach area b, y, and g sequentially, while avoiding the obstacles. Such a task can be written in LTL as $((\neg area_g) \mathrm{U} area_y) \wedge ((\neg area_y) \mathrm{U} area_b) \wedge (\diamond area_b)$.

The reward machines for the above two cases are shown in Fig. 8(c) and (f), respectively. In the experiment, we set $r_0 = -1$, $r_1 = r_2 = 20$, and $r_3 = 100$. The agent is rewarded based on the accomplishment of subtasks, and punished if no progress is made. The cost is designed as

$$c = \begin{cases} \dfrac{50}{1 + e^{10(d - d_{safe})}}, & \text{if } d > d_{safe}, \\ 50, & \text{if } d \leq d_{safe}, \end{cases}$$

which indicates that, the closer the agent to the obstacle, the greater the cost it will get. If it collides with an obstacle, a large penalty will be received.

We consider five baselines: constrained policy optimization (CPO) (Achiam *et al.*, 2017), projection-based constrained policy optimization (PCPO) (Yang *et al.*, 2019), safe policy adaptation with constrained exploration (SPACE) (Yang *et al.*, 2021), reward constrained policy optimization (RCPO) (Tessler *et al.*, 2018), reachability constrained reinforcement learning (RCRL) (Yu *et al.*, 2022). As for RCPO, we use PPO and SAC as the RL base algorithms, denoted as PPO_Lagrangian (PPO_L) and SAC_Lagrangian (SAC_L), respectively. To show the effect of divergence projection step, besides FSPO, we also consider SPO, a simplified version of FSPO in the ablation experiment that only includes the reward improvement step and constrain projection step without the divergence projection step. To compare with the above different update methods, we use FSPO with neural network updated

Lagrangian coefficient (FSPO_N) and FSPO with manually updated Lagrangian coefficient (FSPO_M). Similarly, SPO_M and SPO_N are considered.

Main result

This section presents the comparison results of the policy performance, the safety rate, and the task completion rate between our method and the baseline methods. The role of the divergence projection step and the training process of Lagrangian coefficients are also discussed.

For Case 1, **Fig. 9**(a) and (b) show that FSPO converges slower than the baseline methods, but performs similarly to the baseline methods in terms of the reward collection and the cost performance. This is mainly due to the fact that Case 1 is a relatively simple case, for which the baseline methods can generally perform well. However, when encountering more challenging environments and tasks, as in Case 2 with more obstacles and more complex task requirements, the strength of our method emerges. As shown in **Fig. 9**(c) and (d), the reward of FSPO is higher than baseline algorithms while the cost is significantly lower the others. Note that the SPACE cannot even complete the task. A possible explanation is that, since we do not use the convex approximation method, it results in higher reward and lower cost with high safety guarantees.

We compare the safety rate and task completion rate during and after training of our algorithm with the baseline algorithms for Case 2. We choose the last 2000 episodes in the training to show the results after training. Both SPO and FSPO have high safety rate and high task completion rate. Our algorithm also shows high safe guarantees. It is worth mentioning that, although PPO_L shows the highest safety rate, it cannot complete the task.

As mentioned in (30)–(33), there are two ways to update the Lagrangian coefficients. To verify the effect of the divergence projection step, we compare the performance between the following four algorithms: FSPO_M, FSPO_N, SPO_M and SPO_A. As shown in **Fig. 10**, FSPO and SPO have similar performance for Case 1. When encountering more challenging Case 2, SPO converges slowly even if it has high reward and low cost at last. However, the convergence of FSPO is greatly improved.

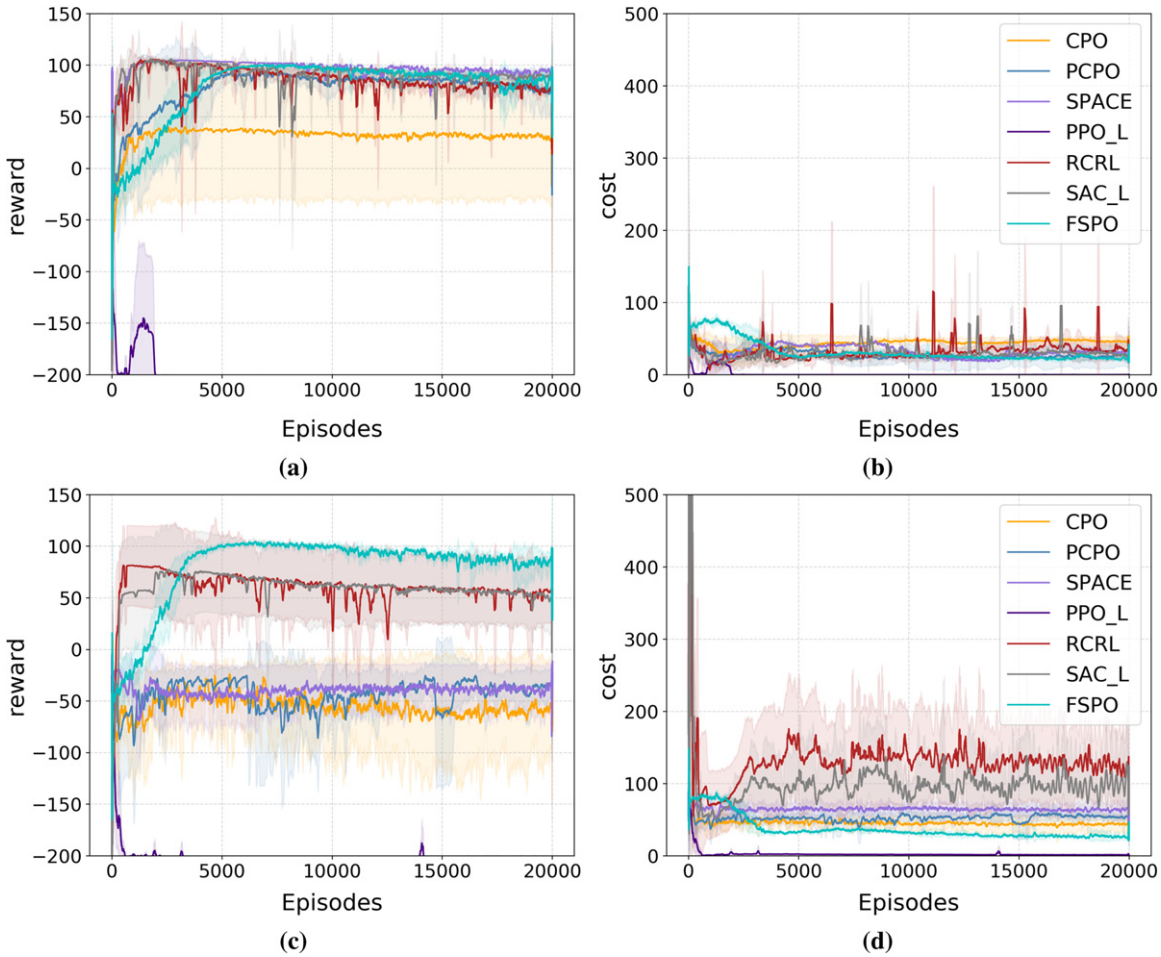


Fig. 9 The reward and cost curves of different algorithm. (a) and (b) show the reward and cost curves for Case 1, while (c) and (d) show the performance for Case 2.

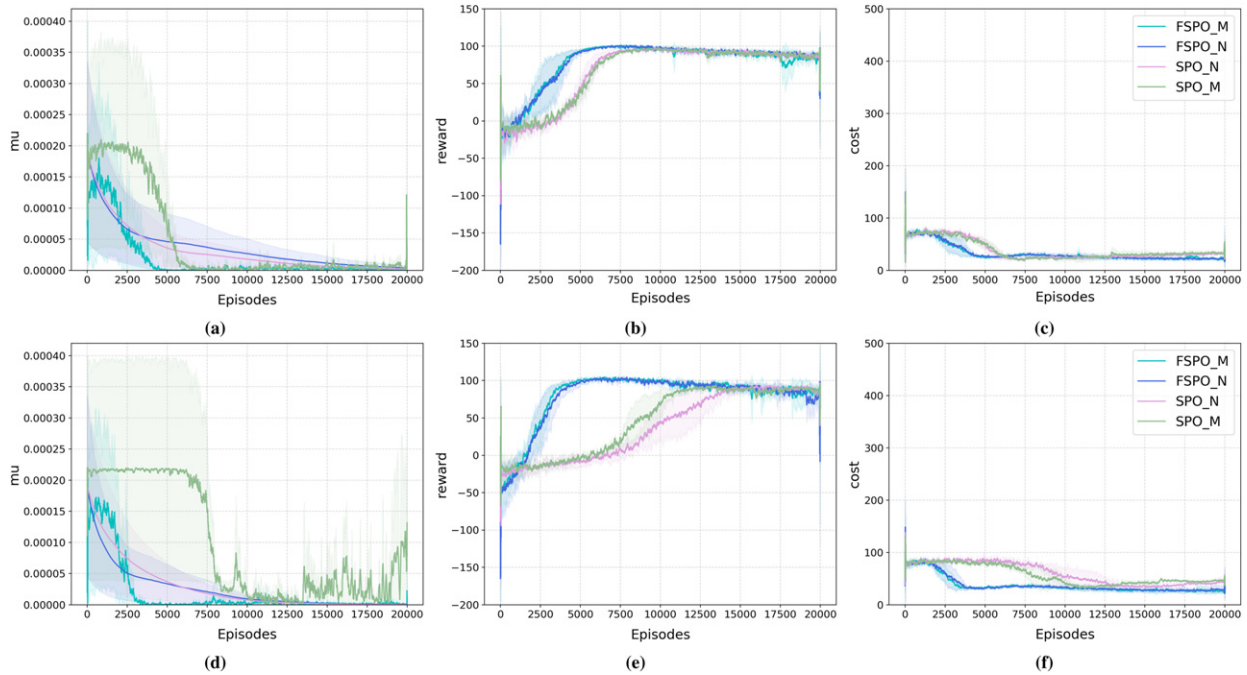


Fig. 10 The evolution of the Lagrangian coefficient, the reward, and the cost using FSPO_M, FSPO_N, SPO_M, and SPO_N. The top row shows the result for Case 1 while the bottom row shows the result for Case 2.

Table 3 Safety rate and task completion rate

Method	*Final safety rate*	*Task completion rate*	*Safety rate during training*
CPO (Achiam *et al.*, 2017)	43.15%	0%	39.50%
PCPO (Yang *et al.*, 2019)	52.75%	0%	49.89%
SPACE (Yang *et al.*, 2021)	11.1%	0%	25.16%
PPO_L (Tessler *et al.*, 2018)	99.46%	0%	98.69%
SAC_L (Tessler *et al.*, 2018)	84.70%	84.31%	57.47%
RCRL (Yu *et al.*, 2022)	87.27%	85.38%	63.42%
SPO_M(Ours)	94.97%	94.97%	58.46%
SPO_A(Ours)	93.25%	93.25%	51.81%
FSPO_M(Ours)	**95.10%**	**94.93%**	**86.53%**
FSPO_N(Ours)	**92.70%**	**91.97%**	**86.25%**

Since Lagrangian coefficient determines the trade off between the safety and performance, we first show the training process of Lagrangian coefficients during the learning process. As shown in **Fig. 10**(a) and (d), the Lagrangian coefficient updated by neural network slowly decreases as the training progresses. But when updating it manually, it stays at a high value when the agent is not safe enough. Once the agent has been trained to be sufficiently safe, the value decreases fast. The performance of FSPO_M is slightly better than FSPO_N (**Table 3**).

The divergence cost means the divergence between the policy and the baseline policy π_B. We show the divergence cost and the Lagrangian coefficients of step 2 in **Fig. 11**a and (b), respectively. It is observed that the value is high at the beginning, and then quickly decreases after some episodes. It is speculated that such a process can help accelerate the training at the beginning and increase the upper limit of the algorithm in the end.

Physical Experiment

The effectiveness of our approach is also verified in physical environment for Case 1 and 2 using a turtlebot. As shown in **Fig. 8**(b) and (e), the turtlebot can safely complete the tasks. The experiment video with more explanations is provided.[4]

[4]https://youtu.be/2RgaH-zcmk

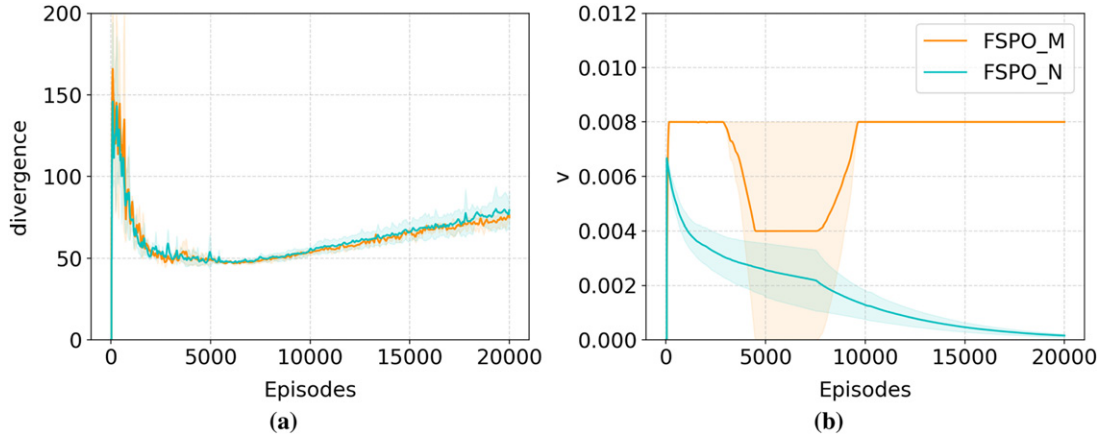


Fig. 11 The evolution of the divergence cost and the Lagrangian coefficient ν for Case 2.

Conclusion

In this section, a Fast and Safe Policy Optimization (FSPO) algorithm is developed to improve the policy learning efficiency and performance while ensuring policy safety. Numerical simulation and physical experiments demonstrate the effectiveness of FSPO. Future research will consider extending FSPO to further ensure safety in both training and deployment.

Shielded Planning Guided Safe Policy Optimization for Reinforcement Learning

In this section, we develop Shielded Planning guided Policy Optimization (SPPO), a new model-based safe reinforcement learning method that augments policy optimization algorithms with path planning and shielding mechanism. In particular, SPPO is equipped with shielded planning for guided exploration and efficient data collection via model predictive path integral (MPPI), along with an advantage-based shielding rule to keep the above processes safe. Based on the collected safe data, a task-oriented parameter optimization (TOPO) method is used for policy improvement, as well as the observation-independent latent dynamics enhancement. In addition, SPPO provides explicit theoretical guarantees, i.e., clear theoretical bounds for training safety, deployment safety, and the learned policy performance. Experiments demonstrate that SPPO outperforms baselines in terms of policy performance, learning efficiency, and safety performance during training.

Problem Formulation

In this section, we first redefine the CMDP and then formally formulate the problem to be addressed. In particular, given a state space S, let $S_{safe}, S_{unsafe} \subset S$ be the safe and unsafe states, respectively, such that $S_{safe} = S \backslash S_{unsafe}$, the unsafe states can be rewritten as $S_{unsafe} := \{s_\triangleright, s_\circ\}$, where $s_\triangleright$ is the set of unsafe regions in the environment and $s_\circ$ is a virtual state that captures the absorbing property of S_{unsafe}. The state space is then extended to $S \cup \{s_\circ\}$. The idea behind is that, when an agent leaves S_{safe} and enters S_{unsafe}, it enters $s_\triangleright$ first, and then enters $s_\circ$ and never leaves no matter what actions the agent takes in $s_\triangleright$. Based on the above definitions, the CMDP can be formally redefined as follows.

Definition 2 A CMDP is a tuple $\mathcal{M}_c = (\mathcal{S}, \mathcal{A}, P, r, c, \gamma)$, which is defined using a reward-based MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$ and a cost-based MDP $\overline{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, P, c, \gamma)$, where $S = S_{safe} \cup S_{unsafe}$ with $S_{unsafe} = \{s_\triangleright, s_\circ\}$ is a state space, $\mathcal{A}$ is an action apace, $P(s'|s,a)$ is the transition dynamics, $r(s,a) \in [0,1]$ is a reward function where the reward on S_{unsafe} is 0, $c(s,a) = 1\{s = s_\triangleright\}$ is the cost function where 1 denotes the indicator function, and $\gamma \in [0,1]$ is a discount factor.

The agent can interact with environment to learn the optimal behavior π^* by solving the following CMDP problem,

$$\begin{aligned} \pi^* &= \underset{\pi}{\operatorname{argmax}} V^\pi, \\ \text{s.t. } & \overline{V}^\pi \le \delta, \end{aligned} \tag{34}$$

to maximize the expected discounted return $V^\pi = \mathbb{E}_\pi\left[\sum_{i=o}^{\infty} \gamma^t r(s_t, a_t)\right]$ over the reward-based MDP $\mathcal{M}$ while satisfying the discounted cost $\overline{V}^\pi = \mathbb{E}_\pi\left[\sum_{i=0}^{\infty} \gamma^t c(s_t, a_t)\right] \le \delta$ over the cost-based MDP $\overline{\mathcal{M}}$. As discussed in Wagener *et al.* (2021), requiring the agent to find a policy with high cumulative reward while entering the unsafe set S_{unsafe} with a low probability (i.e., identifying a safe policy) is equivalent to solving the CMDP problem in (34).

There are two main challenges to solve the CMDP problem in (34). The first challenge is to ensure the agent safety during the training process. Although the CMDP problem in (34) has been extensively studied (Achiam *et al.*, 2017; Bharadhwaj *et al.*, 2021; Chow *et al.*, 2017; Tessler *et al.*, 2018), these methods cannot guarantee the safety of the exploration process and generally suffer from the high computational cost of the stochastic optimizer and poor numerical stability. The control-theoretic techniques (Li and Bastani, 2020; Thananjeyan *et al.*, 2021; Zhang *et al.*, 2023; Zhou *et al.*, 2023), on the other hand, require strong assumptions (e.g., smoothness or ergodicity) that are often not satisfied, and the optimality of the policy is difficult to be guaranteed. Another challenge is that existing model-free RL methods suffer from low sampling efficiency, i.e., the agent perform a large number of invalid interactions with the environment, and may lead to dangerous behavior of the agent due to extensive random exploration.

In this work, our aim is to find a data-efficient and safe algorithm to overcome the above two challenges, i.e., to guarantee sampling efficiency while ensuring the safety of the agent during training.

Problem 1 Given a CMDP $\mathcal{M}_c = (\mathcal{S}, \mathcal{A}, P, r, c, \gamma)$, the goal is to learn a parameterized mapping $\Pi_\theta : \mathcal{S} \to \mathcal{A}$ to solve (34) while ensuring exploration safety along a state-action sequence $\mathcal{L} = (s_0, a_0, s_1, a_1, ..., s_\infty)$, where, at each decision step t, $s_{t+1} \sim P(\cdot|s_t, a_t)$ with $a_t \sim \Pi_\theta(\cdot|s_t)$.

Shielded Planning Guided Policy Optimization

In this section, we provide a new data-efficient and safe reinforcement learning method, namely Shielded Planning guided Policy Optimization (SPPO), to address Problem 1 enabling agents to explore safely and efficiently. We first describe the overview of SPPO in Section "Overview of SPPO" and then explain the technical details in Sections "Advantage-based shielding rules" to "Task-oriented parameter optimization".

Overview of SPPO

The overall idea behind SPPO is to use a shielded planning to ensure the efficiency and safety of exploration during the learning process. As illustrated in **Fig. 12**, SPPO finds an approximate solution to Problem 1 using MPPI (for guided exploration) and advantage-based shielding rules (for safety guarantees) for safe and efficient data collection. A task-oriented parameter optimization method is then used for policy improvement. SPPO has several advantages. First, in contrast to the random exploration in classical RL, we use MPPI for more efficient exploration via a task-oriented latent dynamic model, which completely avoids the dependence of model learning on environmental observations. Second, using the shielding rules, an absorbing MDP $\widetilde{\mathcal{M}}$ is constructed from $\mathcal{M}$, based on which SPPO can find nearly optimal policies that satisfy the properties of the shielding rule and thus ensure exploration safety. In addition, since SPPO transforms the CMDP problem into an unconstrained MDP problem, it is more efficient and stable than typical constraint-based optimization methods.

The pseudo-code of SPPO is outlined in Alg. 3, including the Exploration Phase (lines 3–9) and the Training Phase (lines 10–13). In the exploration phase, a shielded planning algorithm (Alg. 4) is firstly used to determine a safe action a^{safe} and an absorbing action $\widetilde{a}$ to interact with the environment in each step (line 4). The safe action a^{safe} running in $\mathcal{M}$ collects the data set $\mathcal{B}$, while the absorbing action $\widetilde{a}$ running in $\widetilde{\mathcal{M}}$ collects the safety data set $\widetilde{\mathcal{B}}$ (line 5–8). In the training phase, $\widetilde{\mathcal{B}}$ is fed into a task-

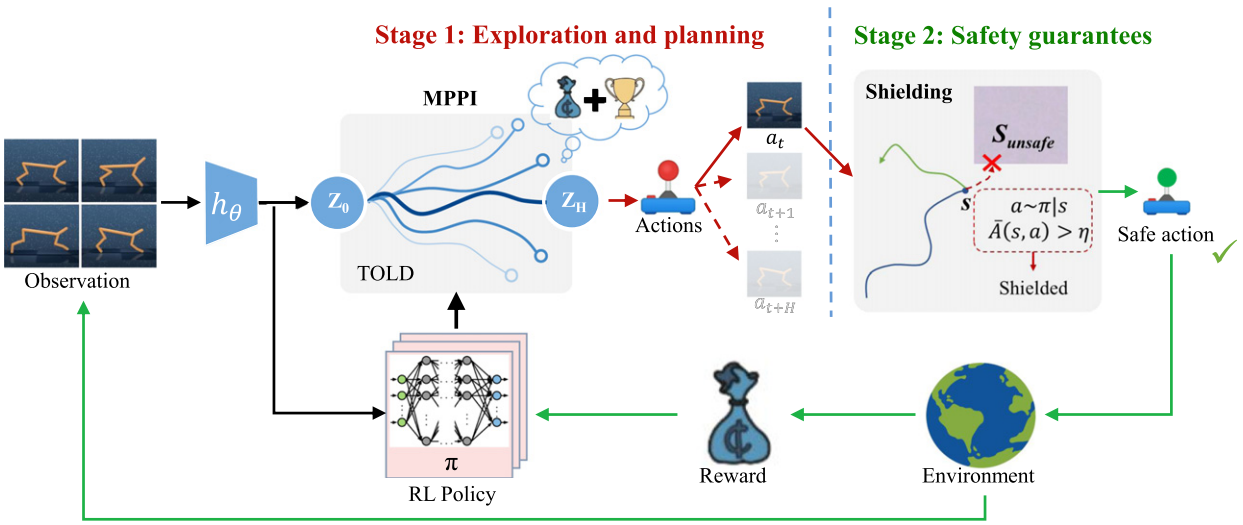


Fig. 12 The framework of SPPO that utilizes a shielded planning to improve exploration efficiency and ensure policy safety during training. SPPO consists of two stages. Stage 1: model predictive path integral (MPPI) is employed for guided exploration and efficient data collection; Stage 2: an advantage-based shielding rule is applied to keep the exploration process safe.

oriented parameter optimization algorithm (Alg. 5) for policy improvement (line 11), and optionally using $\mathcal{B}$ to refine the shielding rule $\mathcal{G}$ in every iteration (line 12). The above process continues until the training budget is exhausted. Then, SPPO terminates and returns the learned optimal network parameters $\mathbf{\Theta}^*$ and policy $\hat{\Pi}^*$ (line 15–16).

In the following, we first present the advantage-based shielding rules and the absorbing MDP in Section "Advantage-based shielding rules" and Section "Absorbing MDP", and detail the shielded planning algorithm based on the shielding mechanism and MPPI in Section "Shielded planning". The task-oriented parameter optimization is introduced in Section "Task-oriented parameter optimization".

Algorithm 3 SPPO

1. **procedure** Input: $\mathcal{B}$, $\widetilde{\mathcal{B}}$, $\mathcal{M}$, and $\widetilde{\mathcal{M}}$
2. Output: $\mathbf{\Theta}^* = (\theta^*, \phi^*, \psi^*, \chi^*, \omega^*), \Pi^*$: Optimal network parameters and policy
3. Initialization: $\mathbf{\Theta}_0 = (\theta_0, \phi_0, \psi_0, \chi_0, \omega_0)$: initial network parameters
4. **while** episode not terminated **do**
5. **for** step $t = 0, 1, ..., T$ Exploration Phase
6. $a_t^{safe}, \widetilde{a} \sim ShieldedPlanning()$ Algorithm 7.2.4
7. $(s_{t+1}, r_t) \sim \mathcal{T}\left(\cdot|s_t, a_t^{safe}\right), r\left(\cdot|s_t, a_t^{safe}\right)$
8. $(\widetilde{s}_{t+1}, \widetilde{r}_t) \sim \widetilde{\mathcal{T}}(\cdot|s_t, \widetilde{a}_t), \widetilde{r}(\cdot|s_t, \widetilde{a}_t)$
9. $\mathcal{B} \leftarrow \mathcal{B} \cup (s_t, a_t, s_{t+1}, r_t)$
10. $\widetilde{\mathcal{B}} \leftarrow \widetilde{\mathcal{B}} \cup (s_t, a_t, \widetilde{s}_{t+1}, \widetilde{r}_t)$
11. **end for**
12. training step $t = 0, 1, ..., K$ Training Phase
13. $\mathbf{\Theta} \leftarrow TOPO\left(\widetilde{B}\right)$ Algorithm 13
14. $\mathcal{G} \leftarrow UpdateShieldingRule(\mathcal{B})$ (Optional)
15. **end for**
16. **end while**
17. $\mathbf{\Theta}^* \leftarrow GetOptimizeParameters()$
18. $\hat{\Pi}^* \leftarrow GetOptimizePolicy\left(\mathbf{\Theta}^*\right)$
19. **end procedure**

Advantage-based shielding rules

This section presents the developed advantage-based shielding rules that map an original policy to a safe policy to enable safe exploration. Specifically, inspired by Wagener *et al.* (2021), given a CMDP $\mathcal{M}_c$ in Def. 2, the corresponding shielding rule is defined as a 5-tuple $\mathcal{G} = (\overline{Q}, \zeta, \overline{A}, \eta, \mathcal{F})$, where $\overline{Q}: S_{safe} \times \mathcal{A} \to [0, 1]$ indicates the state-action cost value estimator, $\zeta \in \Pi$ is a backup policy, $\overline{A}(s, a) := \overline{Q}(s, a) - \overline{Q}(s, a_\zeta)$ represents the advantage-like function between the original policy $\pi(\cdot|s)$ and the backup policy $\zeta(\cdot|s)$, $\eta \in [0, 1]$ is an advantage threshold, and $\mathcal{F} = \left\{(s, a) \in S_{safe} \times \mathcal{A}: \overline{A}(s, a) > \eta\right\}$ is a shielding set. In this work, when $\overline{A}(s, a) > \eta \geq 0$, i.e., $\overline{Q}(s, a) > \overline{Q}(s, a_\zeta)$, the (s, a) is considered unsafe, and thus the shielding set $\mathcal{F}$ is regarded as an unsafe set.

Proposition 3 If $\eta \geq 0$, $\forall (s, a) \in \mathcal{F}$, there always exists some $a' \in \mathcal{A}$ satisfying $(s, a') \notin \mathcal{F}$.

Proof Given a pair $(s, a) \in \mathcal{F}$ and define $a' = \arg\min_{a'' \in \mathcal{A}} \overline{Q}(s, a'')$, one has

$$\overline{A}(s, a') = \left(\overline{Q}\right)(s, a') - \left(\overline{Q}\right)(s, \zeta) \leq 0 \leq \eta,$$

and hence $(s, a') \notin \mathcal{F}$.

Proposition 3 guarantees the existence of safe actions, which is essential for the unconstrained MDP simplification in Section "Shielded planning". Given a policy π, the shielded policy $\pi' = \mathcal{G}(\pi)$ can be synthesized as

$$\pi'(a|s) = \begin{cases} \pi(a|s), & if\, (s, a) \notin \mathcal{F} \\ \zeta(a|s) \cdot (1 - w(s)), & otherwise \end{cases} \tag{35}$$

where $w(s)$ denotes the probability that, when $(s, a) \notin \mathcal{F}$, π' still take actions in $\mathcal{F}$. As indicated in (35), when sampling an action a from $\pi'(a|s)$, an action a' is first sampled from $\pi(\cdot|s)$. If $(s, a') \notin \mathcal{F}$, $a = a'$, otherwise, a samples from $\zeta(\cdot|s)$. Since the shielding set $\mathcal{F}$ only allows actions that are no less safe than the backup policy ζ, the shielded policy π' is at least as safe as ζ. In this way, the shielding rule is effective in ensuring safety. It is worth noting that the shielded policy π' synthesized by (35) may still take actions in $\mathcal{F}$, depending on the probability assigned to these actions by ζ. This design avoids the problem that direct intervention rules based on Q-based functions (Bharadhwaj *et al.*, 2021; Thananjeyan *et al.*, 2021) may be overly conservative, providing the agent with a trade-off between exploratory and safety. Hence, we formally define a class of adoptable shielding rules.

Definition 3 (β-Adoptable Shielding Rule). A shielding rule $\mathcal{G} = \left(\overline{Q}, \zeta, \overline{A}, \eta, \mathcal{F}\right)$ is β-adoptable if $\overline{Q}(s, a) \in [0, \gamma]$ and for some $\beta \geq 0$ the following holds for all $s \in S_{safe}$ and $a \in \mathcal{A}$:

$$\overline{Q}(s,a)+\sigma \geq c(s,a)+\gamma \mathbb{E}_{s'|s,a}\left[\overline{Q}\left(s',\zeta\right)\right], \tag{36}$$

where $c(s,a)=1\{s=s_{\triangleright}\}$. If the above still holds for $\beta=0$, then $\mathcal{G}$ is said *adoptable*.

Proposition 4 *The β-adoptable shielding rule has the following properties.*

1. Baseline policy: If ζ is a baseline policy of $\overline{\mathcal{M}}$ and ζ^{+} is a greedy policy of $\mathcal{M}_c$ with $\overline{Q}^{\zeta}$ denoting the state-action cost of ζ on $\overline{\mathcal{M}}$, then $\mathcal{G}=\left(\overline{Q}^{\zeta},\zeta,\overline{A},\eta,\mathcal{F}\right)$ or $\mathcal{G}=\left(\overline{Q}^{\zeta},\zeta^{+},\overline{A},\eta,\mathcal{F}\right)$ is adoptable.

2. Optimal shielding: If $\overline{\pi}^{*}$ is the optimal policy for $\overline{\mathcal{M}}$ with $\overline{Q}^{*}$ denoting the corresponding state-action value function, then $\mathcal{G}^{*}=\left(\overline{Q}^{*},\overline{\pi}^{*},\overline{A}^{*},\eta,\mathcal{F}^{*}\right)$ is adoptable.

3. Approximation: Consider a β-adoptable $\mathcal{G}=\left(\overline{Q},\zeta,\overline{A},\eta,\mathcal{F}\right)$ and $\hat{Q}\in[0,\gamma]$ for all $s\in S_{safe}$ and $a\in\mathcal{A}$, if $\|\hat{Q}-\overline{Q}\|_{\infty}\leq\delta$, then $\hat{\mathcal{G}}=\left(\hat{Q},\zeta,\hat{A},\eta,\hat{\mathcal{F}}\right)$ is $\beta+(1+\gamma)\delta$-adoptable.

The proof of Proposition 4 is omitted, since it is a trivial extension of Proposition 3 in Wagener *et al.* (2021). Based on Proposition 4, a β-adoptable shielding rule can be constructed by: 1) using an existing baseline policy, 2) performing short-term planning, e.g., model-predictive control. 3) searching for the optimal state-action value function $\overline{Q}^{*}$ in $\overline{\mathcal{M}}$, and 4) learning from data or inferring from inaccurate models. Note that among all the shielding rules that provide optimal safety, $\mathcal{G}^{*}=\left(\overline{Q}^{*},\overline{\pi}^{*},\overline{A}^{*},0,\mathcal{F}^{*}\right)$ provides the most free space for safe exploration. In this way, the newly collected data from $\overline{\mathcal{M}}$ during SPPO learning process can be used to improve the estimation of the ideal $\overline{Q}$, i.e., performing additional policy improvement to find $\overline{Q}^{*}$ in $\overline{\mathcal{M}}$. Next, we will demonstrate how to use shielding rules for safe planning.

Absorbing MDP

One advantage of SPPO is that it transforms the CMDP problem into an unconstrained MDP problem by constructing an absorbing MDP. Given an MDP $\mathcal{M}=(\mathcal{S},\mathcal{A},P,r,\gamma)$ and an absorbing state $\{s_{\dagger}\}$, based on the shielding set $\mathcal{F}$ introduced in Section "Advantage-based shielding rules", the absorbing MDP corresponding to $\mathcal{M}$ is constructed as $\widetilde{\mathcal{M}}=\left(\widetilde{\mathcal{S}},\mathcal{A},\widetilde{P},\widetilde{r},\gamma\right)$, where. The transition reward is modified as

$$\widetilde{r}(s,a)=\begin{cases}\widetilde{R}, & (s,a)\in\mathcal{F},\\ 0, & s=s_{\dagger},\\ r(s,a), & \text{otherwise},\end{cases} \tag{37}$$

where $\widetilde{R}\leq 0$ is the penalty. The transition probability is defined as

$$\widetilde{P}\left(s'|s,a\right)=\begin{cases}\mathbb{1}\{s'=s_{\dagger}\}, & (s,a)\in\mathcal{F}\ \text{ or }\ s=s_{\dagger}\\ P\left(s'|s,a\right), & \text{otherwise}.\end{cases} \tag{38}$$

Without loss of generality, when the agent is in an absorbing state $s_{\dagger}$, the policy $\pi(a|s_{\dagger})$ is defined as the uniform distribution over $\mathcal{A}$.

The absorbing MDP $\widetilde{\mathcal{M}}$ constructed from $\mathcal{M}$ has more absorbing state-action pairs with lower rewards compared to the original one. When an agent performs some $(s,a)\in\mathcal{F}$, it enters an absorbing state $s_{\dagger}$ and receives a penalty $\widetilde{R}$. This means that performing shielding state-actions in $\widetilde{\mathcal{M}}$ have larger potential penalties than entering unsafe set S_{unsafe} with $r=0$. This design guarantees high reward and low probability access to shielded state-action pairs when any near-optimal policy of $\widetilde{\mathcal{M}}$ runs on $\mathcal{M}$. Since we simulate the experience of policy Π_{θ} in $\widetilde{\mathcal{M}}$ by running policy $\Pi_{\theta}^{'}$ in $\mathcal{M}$ and collecting samples for SPPO as shown in Section "Shielded planning". and Section "Task-oriented parameter optimization", solving $\widetilde{\mathcal{M}}$ yields a safe policy with potentially good performance in the original MDP $\mathcal{M}$ as long as $\mathcal{G}$ provides a safe shielded policy, as will be detailed in Section "Theoretical Analysis of SPPO".

Shielded planning

Classical RL algorithms often rely on random exploration to interact with the environment, which is neither efficient nor safe. In this section we propose a shielded planning method based on MPPI and shielding rules to solve the above problem. The procedure of shielded planning is outlined in Alg. 3, including policy guided planning based on MPPI (line 3–14) as well as safe shielding based on advantage-based shielding mechanisms (line 15–17).

Algorithm 4 Shielded Planning

1. **procedure** Input: ($\theta,\phi,\psi,\chi,\omega$: learned network parameters of policy, representation, latent dynamics, reward and value; N/N_{π}: number of sample/policy trajectories; s_t, H: current state, rollout horizon)

   Output: a^{safe},$\widetilde{a}$: shielded safe action, action of $\widetilde{\mathcal{M}}$

   Initialization: μ^{0}, σ^{0}: initial parameters for $\mathcal{N}$

2. Encode state $z_t\leftarrow h_{\phi}(s_t)$

3. **for** iteration $j = 1, 2, ..., J$ **do**
4. Sample N traj. of len. H from $\mathcal{N}\left(\mu^{j-1}, (\sigma^{j-1})^2\mathbf{I}\right)$
5. Sample N_π traj. of len. H using π_θ, d_ψ
6. **for** all $N + N_\pi$ trajectories $(a_t, a_{t+1}, ..., a_{t+H})$ **do**
7. **for** step $t = 0, 1, ..., H-1$ **do**
8. $V_\Gamma = V_\Gamma + \gamma^t R_\chi(z_t, a_t)$
9. $z_{t+1} = d_\psi(z_t, a_t)$
10. **end for**
11. $V_\Gamma = V_\Gamma + \gamma^H Q_\omega(z_H, a_H)$
12. **end for**
13. $\mu^j = \frac{\sum_{i=1}^k \Omega_i \Gamma_i^\star}{\sum_{i=1}^k \Omega_i}, \sigma^j = \max\left(\sqrt{\frac{\sum_{i=1}^k \Omega_i \left(\Gamma_i^\star - \mu^j\right)^2}{\sum_{i=1}^k \Omega_i}}, \varepsilon\right)$
14. **end for**
15. $a \sim \mathcal{N}\left(\mu^J, (\sigma^J)^2\mathbf{I}\right)$
16. $a^{safe} = \begin{cases} a, \ if (s,a) \notin \mathcal{F} \\ a_\zeta \sim [\zeta(\cdot|s) \cdot (1 - w(s))], \ otherwise \end{cases}$
17. $\widetilde{a} = \begin{cases} a, if \ \ a^{safe} = a \\ a_T, if \ \ a_T^{safe} = a_\zeta \\ rand(A), if \ \ a_T^{safe} = a_\zeta \ \ and \ \ t \geq T+1 \end{cases}$

return $a^{safe}, \widetilde{a}$

18. **end procedure**

Before presenting the shielded planning algorithm, we first introduce the Task-Oriented Latent Dynamics (TOLD) (Hansen et al., 2022)

$$\begin{aligned} \text{Representation}: \quad & z_t = h_\phi(s_t) \\ \text{Latentdynamics}: \quad & z_{t+1} = d_\psi(z_t, a_t) \\ \text{Reward}: \quad & \hat{r}_t = R_\chi(z_t, a_t) \\ \text{Value}: \quad & \hat{q}_t = Q_\omega(z_t, a_t) \\ \text{Policy}: \quad & \hat{a}_t = \pi_\theta(z_t) \end{aligned} \tag{39}$$

where $\theta, \phi, \psi, \chi, \omega$ denote the network parameters of policy, representation, latent dynamics, reward and value, respectively. Given the current action a_t and the latent state z_t encoded by the representation network h_ϕ using the observed state s_t as inputs, TOLD can predict the next moment latent state z_{t+1}, the single-step reward $\hat{r}_t$, the state action value $\hat{q}_t$, and the action $\hat{a}_t$ that (approximately) maximizes the Q-function. The update and optimization of TOLD parameters will be introduced in Section "Task-oriented parameter optimization". The use of TOLD has two advantages. On one hand, TOLD learns to model only the elements of the environment that predict rewards, avoiding the need to model the environment itself, thus making the problem simpler. On the other hand, all components of TOLD are implemented using only deterministic MLPs, i.e., neither an RNN gating mechanism nor a probabilistic model.

Based on TOLD, we redefine the return Φ_Γ in MPPI as

$$\Phi_\Gamma = \mathbb{E}_\Gamma\left[\gamma^H Q_\omega(z_H, a_H) + \sum_{t=0}^{H-1} \gamma^t R_\chi(z_t, a_t)\right] \tag{40}$$

where $\gamma^H Q_\omega(z_H, a_H)$ is the terminal return and $\sum_{t=0}^{H-1} \gamma^t R_\chi(z_t, a_t)$ is the state-dependent step return.

Although MPPI is inherently stochastic in planning due to sampling, it tends to fall into local optima due to insufficient exploration. To encourage MPPI to perform consistent exploration across tasks like model-free RL, the standard deviation σ of the sampling distribution is restricted. That is, for μ^j obtained from (2), σ^j is rewritten as

$$\sigma^j = \max\left(\sqrt{\frac{\sum_{i=1}^k \Omega_i \left(\Gamma_i^\star - \mu^j\right)^2}{\sum_{i=1}^k \Omega_i}}, \varepsilon\right) \tag{41}$$

where $\varepsilon \in \mathbb{R}_+$ is a linearly decayed constant.

Another trick in the shielded planning is to use the learned policy π_θ to guide trajectory optimization. In particular, the sampled trajectory using MPPI consists of N trajectories from $\mathcal{N}\left(\mu^{j-1}, (\sigma^{j-1})^2\mathbf{I}\right)$ and N_π trajectories from π_θ. The top-k trajectories are selected from all $N + N_\pi$ sampled trajectories. This sampling method leads to one of two situations: the policy trajectories will be excluded from the top-k if they are poorly estimated, otherwise, the policy trajectories may be included with an impact proportional to their estimated return Φ_Γ.

After J iterations, the current action a can be sampled from $\mathcal{N}\left(\mu^J, (\sigma^J)^2\mathbf{I}\right)$ and filtered based on the shielding rules to ensure its safety. Mathematically,

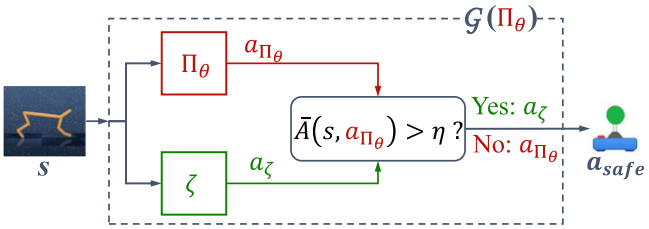


Fig. 13 Construction of shielded policy. The agent executes the action a_{Π_θ} sampled from Π_θ, if $(s, a_{\Pi_\theta}) \notin \mathcal{F}$, i.e., the agent is safe to execute a_{Π_θ} in state s. Otherwise, it executes the safe a_ζ sampled from the backup policy ζ.

$$a^{safe} = \begin{cases} a, \ if (s,a) \notin \mathcal{F} \\ a_\zeta \sim [\zeta(\cdot|s) \cdot (1 - w(s))], \ \text{otherwise} \end{cases} \tag{42}$$

where $(s,a) \in S_{safe} \times \mathcal{A}$, a_ζ is sampled according to the updated backup policy, i.e., $\zeta(\cdot|s) \cdot (1 - w(s))$ where $w(s)$ denotes the probability that a^{safe} still takes actions in $\mathcal{F}$ when $(s,a) \notin \mathcal{F}$. As indicated in (42), the agent executes the action a sampled from $\mathcal{N}\left(\mu^l, (\sigma^l)^2 \mathbf{I}\right)$ if $(s,a) \notin \mathcal{F}$, i.e., it is safe to execute a in state s. Otherwise, it executes the safe a_ζ sampled from the updated backup policy. The shielded safe policy $\Pi_\theta^{'} = \mathcal{G}(\Pi_\theta)$ can be constructed from (42), as illustrated in **Fig. 13**.

Given an MDP $\mathcal{M}$, the corresponding absorbing MDP $\widetilde{\mathcal{M}}$ can be constructed as discussed in Section "Absorbing MDP". Specifically, $\widetilde{r}$ and $\widetilde{P}$ are shown in (37) and (38), respectively. The policy Π_θ running in $\widetilde{\mathcal{M}}$ can be constructed by $\Pi_\theta^{'}$ in $\mathcal{M}$ according to

$$\widetilde{a} = \begin{cases} a, if \ \ a^{safe} = a \\ a_T, if \ \ a_T^{safe} = a_\zeta \\ rand(A), if \ \ a_T^{safe} = a_\zeta \ \ and \ \ t \geq T+1 \end{cases}$$

where $\widetilde{a} \sim \Pi_\theta(\cdot|\widetilde{s})$, $a^{safe} \sim \Pi_\theta^{'}(\cdot|s)$ and T is the first time of shielding intervention. Thus, we can simulate the experience of Π_θ in $\widetilde{\mathcal{M}}$ by running $\Pi_\theta^{'}$ in $\mathcal{M}$ and collecting samples $\mathcal{B}$ and $\widetilde{\mathcal{B}}$ until the shielding intervention is triggered. For instance, if $\mathcal{L} = (s_0, a_0, s_1, a_1, ..., s_T, a_T^{'}, ...)$ is a state-action sequence generated by running $\Pi_\theta^{'}$ in $\mathcal{M}$, where $a_T^{'}$ is the first action sampled from the backup policy $\zeta(\cdot|s_T)$, and a_T denotes the overridden action sampled from $\Pi_\theta(\cdot|s_T)$. Then, the state-action sequence $\widetilde{\mathcal{L}} = (s_0, a_0, s_1, a_1, ..., s_T, a_T, \widetilde{s}_{T+1}, \widetilde{a}_{T+1}, ...)$ can be constructed to simulate running Π_θ in $\widetilde{\mathcal{M}}$, where $\widetilde{s}_\tau = s_\dagger$ and $\widetilde{a}_\tau \in \mathcal{A}$ is arbitrary for any $\tau \geq T+1$. Since $\mathcal{M}$ and $\widetilde{\mathcal{M}}$ share the same dynamics until the shielding intervention occurs, the above process is reasonable. We will present the theoretical guarantees of this process in Section "Theoretical Analysis of SPPO"..

Task-oriented parameter optimization

As introduced in Section "Shielded planning"., our shielded planning is based on TOLD. Since TOLD learns to model only the elements of the environment that predict rewards, our parameter optimization approach is oriented towards task goal completion, which we refer to as Task-Oriented Parameter Optimization (TOPO).

In the training process, the overall goal of TOPO is to minimize the local weighted objective function

$$J(\mathbf{\Theta}, \Gamma) = \sum_{i=t}^{t+H} \lambda^{i-t} L(\mathbf{\Theta}, \Gamma_i), \tag{43}$$

where $\Gamma \sim \widetilde{\mathcal{B}}$ is a trajectory $(s_t, a_t, r_t, s_{t+1})_{t:t+H}$ sampled from the replay buffer $\widetilde{\mathcal{B}}$, $\lambda \in \mathbb{R}_+$ is a constant that has a higher weight for the near-term predictions. The single-step loss function is

$$L(\mathbf{\Theta}, \Gamma_i) = c_1 L_r(\chi, \Gamma_i) + c_2 L_v(\theta, \omega, \Gamma_i) + c_3 L_c(\phi, \psi, \Gamma_i), \tag{44}$$

which includes the reward prediction loss

$$L_r(\chi, \Gamma_i) = \| R_\chi(z_i, a_i) - r_i \|_2^2, \tag{45}$$

the value prediction loss

$$L_v(\theta, \omega, \Gamma_i) = \| Q_\omega(z_i, a_i) - (r_i + \gamma Q_{\omega^-}(z_{i+1}, \pi_\theta(z_{i+1}))) \|_2^2, \tag{46}$$

and the latent state consistency loss

$$L_c(\phi, \psi, \Gamma_i) = \| d_\psi(z_i, a_i) - h_{\phi^-}(s_{i+1}) \|_2^2, \tag{47}$$

where $c_{1:3}$ in (44) are tuning weights balancing the losses.

Apparently, (43) is convex and can be solved by the gradient descent method. Alg. 5 outlines the process of TOPO. For each update episode, a trajectory $(s_t, a_t, r_t, s_{t+1})_{t:t+H}$ sampled from the replay buffer $\widetilde{\mathcal{B}}$ (line 3), and the state s_t is mapped to an latent representation $z_t = h_\phi(s_t)$ (line 4). In the H-step prediction loop, rewards, values, and policies are predicted and used to compute the total cost J according to (43)-(47) (line 5–11). Note that the recurrent prediction is completely in the latent space, i.e., $z_t = h_\phi(s_t)$, $z_{t+1} = d_\psi(s_t, a_t)$,..., $z_{t+H} = d_\psi(s_{t+H-1}, a_{t+H-1})$. Only the first observation s_t is encoded with h_ϕ and the others are predicted by d_ψ. Based on J, the parameters $\mathbf{\Theta}$ are updated by back-propagation through the gradient descent method (line 12), and the

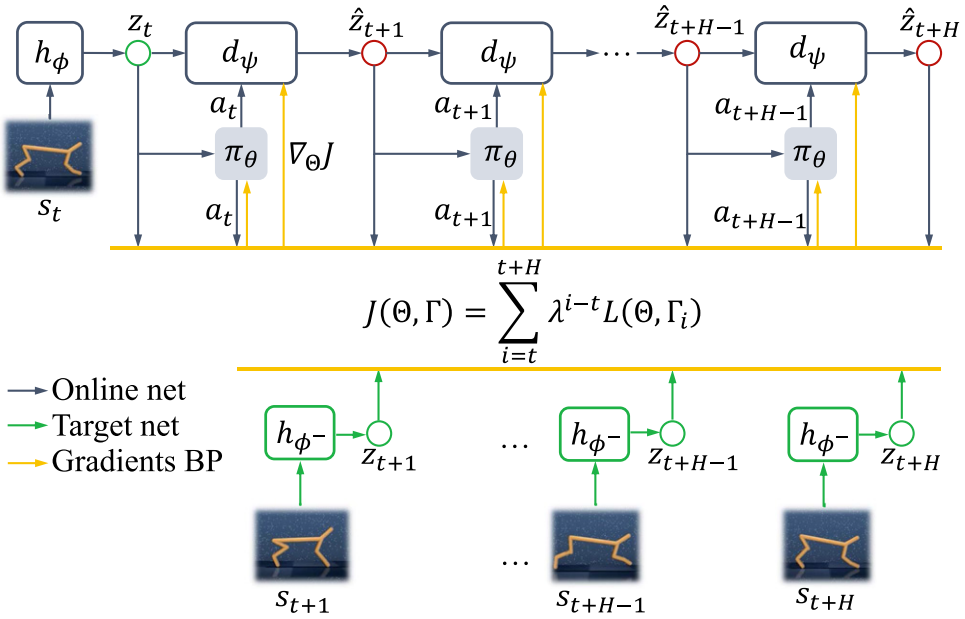


Fig. 14 Update of TOPO. In each update episode, a H-step trajectory $\Gamma_{t:t+H}$ is sampled from the replay buffer. The first state s_t is mapped to an latent representation z_t, and then the following latent states $\hat{z}_{t+1}, \hat{z}_{t+2}, \ldots, \hat{z}_{t+H}$, as well as the rewards, values, and policies are predicted and used to compute the total cost J. The parameters Θ are updated by back-propagation of J through the gradient descent method, and the target network parameters Θ^- are updated by an exponential moving average of Θ.

target network parameters $\boldsymbol{\Theta}^-$ are updated by an exponential moving average of $\boldsymbol{\Theta}$ (line 13). The above process is repeated until the update budget is exhausted, and then the optimized $\boldsymbol{\Theta}$ and $\boldsymbol{\Theta}^-$ are returned. The update process of TOPO is graphically shown in Fig. (14).

Most of the existing model-based RL methods usually predict future observations directly (Janner *et al.*, 2019; Kaiser *et al.*, 2019; Schrittwieser *et al.*, 2020). However, since these methods require the network to simulate everything in the environment for learning to accurately predict the future, this is extremely difficult. In this paper, TOPO uses a latent state consistency loss to optimize the latent dynamics in TOLD, completely avoiding the dependence of model learning on environmental observations. In addition, as TOPO uses trajectories sampled from $\widetilde{\mathcal{B}}$, the optimized policy can ensure safety during the training process, this property will be formally discussed Section "Theoretical Analysis of SPPO".

Algorithm 5 Task-Oriented Parameter Optimization (TOPO)

1. **procedure** Input: $\eta, \tau, \lambda, \widetilde{\mathcal{B}}$: learning rate, coefficients, replay buffer of $\widetilde{\mathcal{M}}$

   Output: $\boldsymbol{\Theta} = (\theta, \phi, \psi, \chi, \omega), \boldsymbol{\Theta}^-$: optimized network parameters, target network parameters

   Initialization: $J = 0$: initialize J for loss accumulation

2. **for** num updates per episode **do**
3. $\{s_t, a_t, r_t, s_{t+1}\}_{t:t+H} \sim \widetilde{\mathcal{B}}$
4. $z_t = h_\phi(s_t)$
5. **for** $i = t, t+1, \ldots, t+H$ **do**
6. $\hat{r}_i = R_\chi(z_i, a_i)$
7. $\hat{q}_i = Q_\omega(z_i, a_i)$
8. $z_{i+1} = d_\psi(z_i, a_i)$
9. $\hat{a}_i = \pi_\theta(z_i)$
10. $J \leftarrow J + \lambda^{i-1} L(z_{i+1}, \hat{r}_i, \hat{q}_i, \hat{a}_i)$
11. **end for**
12. $\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - \frac{1}{H}\eta \nabla_{\boldsymbol{\Theta}} J$
13. $\boldsymbol{\Theta}^- \leftarrow (1-\tau)\boldsymbol{\Theta}^- + \tau\boldsymbol{\Theta}$
14. **end for**

    return $\boldsymbol{\Theta}, \boldsymbol{\Theta}^-$
15. **end procedure**

Theoretical Analysis of SPPO

In this section, we present the theoretical guarantees of SPPO, including the safety of shielded planning policy (i.e., the safety guarantee in exploration process), the suboptimality in $\widetilde{\mathcal{M}}$ to suboptimality and safety in $\mathcal{M}$, and the performance and safety of the learned policy. The advantages of SPPO are also summarized and discussed.

Theoretical guarantees

To facilitate the following analysis, we use $\overline{V}^{\square}$ and $\overline{Q}^{\square}$ to denote the corresponding $\overline{V}$ and $\overline{Q}$ using the policy $\square$ (e.g., $\square$ can take π', ζ, etc.),

Theorem 2 *(Safety of shielded planning policy) If* $\mathcal{G} = (\overline{Q}, \zeta, \overline{A}, \eta, \mathcal{F})$ *is a* β-adoptable shielding rule and the shielded planning policy is $\pi' = \mathcal{G}(\Pi)$ where Π is the planning policy without shielding for $\widetilde{\mathcal{M}}$, then,

$$\overline{V}^{\Pi} \leq \overline{Q}^{\zeta} + \frac{\min\{\beta + \eta, 2\gamma\}}{1-\gamma}.$$

where γ is the discount factor of $\widetilde{\mathcal{M}}$.

Theorem 3 *(Optimality of shielding planning policy) Assume that* $\widetilde{\Pi}$ *is a* ε-suboptimal policy for $\widetilde{\mathcal{M}}$ learned using the shielded planning policy π' on $\mathcal{M}$. If $\widetilde{R}$ is negative, then, given any suboptimal policy π^*, it holds for $\widetilde{\Pi}$ in $\mathcal{M}$ with the following performance and safety guarantees:

$$V^{\pi^*} - V^{\widetilde{\Pi}} \leq \left(|\widetilde{R}| + \frac{1}{1-\gamma}\right) P_{\mathcal{G}}(\pi^*) + \varepsilon$$
$$\overline{V}^{\widetilde{\Pi}} \leq \overline{V}^{\pi'} + \frac{\varepsilon}{|\widetilde{R}|}$$

where $P_{\mathcal{G}}(\pi^*) = (1-\gamma)\sum_{h=0}^{\infty} \gamma^h \Pr\left(\xi^h \cap \mathcal{F} \neq \varnothing | \pi^*, \mathcal{M}\right)$ denotes the probability that π^* visits $\mathcal{F}$ in $\mathcal{M}$.

Since Theorem 2 and 3 are trivial extensions of Theorem 2 and Proposition 7 of Wagener *et al.* (2021), their proofs are omitted here. Theorem 2 provides an upper bound for the value function under the shielded planning policy π' over the cost-based MDP $\overline{\mathcal{M}}$, which indicates that π' generated by the shielding rule $\mathcal{G} = (\overline{Q}, \zeta, \overline{A}, \eta, \mathcal{F})$ has a small unsafe cost if the backup policy ζ has a small cost. Hence, the safety of the agent in the exploration process can be ensured by selecting appropriate shielding rules. Theorem 3 gives performance and security bounds for the policy $\widetilde{\Pi}$ in $\widetilde{\mathcal{M}}$ learned using the shielded planning policy π' on $\mathcal{M}$, which justifies the collection of data from $\widetilde{\mathcal{M}}$ and its use for the training of TOPO. Based on Theorem 2 and 3, the following corollary is developed.

Corollary 1 *(Performance and safety of the learned policy) If* $\widetilde{R} = -1$ *and* $\mathcal{G} = (\overline{Q}, \zeta, \overline{A}, \eta, \mathcal{F})$ is a β-adoptable shielding rule, suppose $\hat{\Pi}$ is a ε-suboptimal policy for $\widetilde{\mathcal{M}}$ learned by SPPO, then, given any suboptimal policy π^*, it holds for $\hat{\Pi}$ in $\mathcal{M}$ with the following performance and safety guarantees:

$$V^{\pi^*} - V^{\hat{\Pi}} \leq \frac{2}{1-\gamma} P_{\mathcal{G}}(\pi^*) + \varepsilon$$
$$\overline{V}^{\hat{\Pi}} \leq \overline{Q}^{\zeta} + \frac{min\{\beta + \eta, 2\gamma\}}{1-\gamma} + \varepsilon.$$

Proof From Theorem 3, one has

$$V^{\pi^*} - V^{\hat{\Pi}} \leq \left(|\widetilde{R}| + \frac{1}{1-\gamma}\right) P_{\mathcal{G}}(\pi^*) + \varepsilon = \left(1 + \frac{1}{1-\gamma}\right) P_{\mathcal{G}}(\pi^*) + \varepsilon = \left(\frac{2-\gamma}{1-\gamma}\right) P_{\mathcal{G}}(\pi^*) + \varepsilon \leq \frac{2}{1-\gamma} P_{\mathcal{G}}(\pi^*) + \varepsilon$$

For the safety bound, one has

$$\overline{V}^{\hat{\Pi}} \leq \overline{Q}^{\mathcal{G}(\hat{\Pi})} + \varepsilon \quad \leq \overline{Q}^{\zeta} + \frac{\min\{\beta + \eta, 2\gamma\}}{1-\gamma} + \varepsilon$$

where the second inequality is derived from Theorem 2.

Corollary 1 indicates that if SPPO finds a ε-suboptimal policy $\hat{\Pi}$ in $\widetilde{\mathcal{M}}$, then this $\hat{\Pi}$ is also roughly ε-suboptimal in the original MDP $\mathcal{M}$ as long as $P_{\mathcal{G}}(\pi^*)$ is sufficiently small and the additional error is proportional to $P_{\mathcal{G}}(\pi^*)$. Furthermore, the learned policy $\hat{\Pi}$ is nearly as safe as the backup policy ζ. The deterioration in safety depends on the suboptimality ε, the shielding threshold η, and the parameter β, i.e., small values lead to small deterioration. It is worth noting that the above results only require a mild assumption that the unsafe subset S_{unsafe} is absorbing and the reward on S_{unsafe} is 0, without any other assumptions.

Theoretical advantages

The theoretical advantages of SPPO are summarized as follows. In contrast to classical RL algorithms that typically rely on stochastic exploration to interact with the environment, we apply planning-oriented exploration in reinforcement learning to improve learning efficiency. Meanwhile, since existing model-based RL methods usually predict future observations, they require the network to simulate everything in the environment to accurately predict the future, which is extremely difficult. In this paper, we address this challenge by using a task-oriented latent dynamics to simulate the environment model, and then optimizing this model by TOPO with potential latent state consistency loss, completely avoiding the dependence of model learning on environmental observations. On the other hand, the advantage-based shielding rules are applied to constrain the planning exploration to ensure safety during training. Although safe reinforcement learning has been widely studied, existing CMDP-based safe RL methods (Achiam *et al.*, 2017; Bharadhwaj *et al.*, 2021; Chow *et al.*, 2017; Tessler *et al.*, 2018) hardly guarantee the safety of the exploration process and suffer from the computational cost and poor numerical stability, while control-theoretic techniques (Li and Bastani, 2020; Thananjeyan *et al.*, 2021; Zhang *et al.*, 2023; Zhou *et al.*, 2023) usually require intractable strong assumptions and hardly guarantee the optimality of the policy. In contrast, our SPPO is able to guarantee the training safety by constraining the exploration with shielding rules. Moreover, SPPO is ingeniously designed to transform a constrained optimization problem into an unconstrained one, which makes the optimization process simpler, more stable, and easier to solve. Another advantage of SPPO is that it provides explicit theoretical guarantees, i.e., clear theoretical bounds for training and deployment safety as well as for the performance of learned policy. Note that the above results make only a mild assumption, i.e., the unsafe subset S_{unsafe} is absorbing and the reward on S_{unsafe} is 0.

Experiments

In this section, SPPO is evaluated against previous works. Particularly, we investigate **1) policy performance:** whether the policy learned by SPPO can obtain higher returns than the baselines; **2) learning efficiency:** whether SPPO outperforms previous methods in terms of exploration efficiency; **3) training safety:** whether SPPO can guarantee the safety of agents during training.

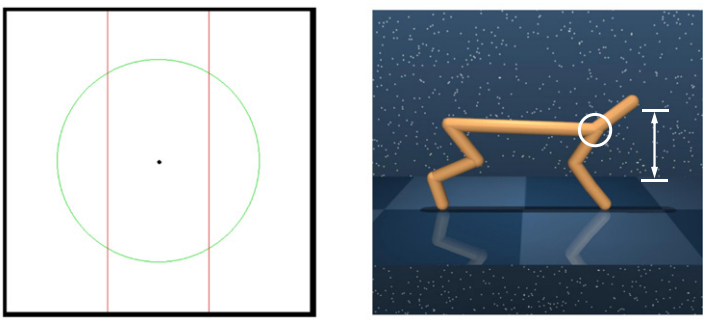
Experimental setup

We consider two different environments in **Fig. 15**. One is the Point Robot (Achiam *et al.*, 2017), which is rewarded for tracking a circular route with high speed, but restricted to areas smaller than the desired circle. The other is the Half-Cheetah (Tunyasuvunakool *et al.*, 2020), which is rewarded for high forward speed, but restricted to limit the height of one of its links to a given range, beyond which the agent is considered as unsafe. In both experiments, a shaped cost function $\hat{c}$ is applied to compute $\overline{Q}$, making our shielding mechanism more conservative, and thus the training process is much safer. The $\hat{c}$ is defined as

$$\hat{c} = \begin{cases} \mathbb{1}\{\mathrm{dist}(s, S_{unsafe}) = 0\}, & \alpha = 0, \\ max\left\{0, 1 - \frac{1}{\alpha}\mathrm{dist}(s, S_{unsafe})\right\}, & otherwise, \end{cases} \tag{48}$$

where $\mathrm{dist}(s, S_{unsafe})$ denotes the distance from the current state s to the unsafe region, α is a non-negative constant. Note that, if $\alpha > 0$, $\hat{c}$ is upper bounded by the original sparse cost c, and $\hat{c} = c$ if $\alpha = 0$.

To ensure the safety of the exploration process, we construct the shielding rule $\mathcal{G} = (\overline{Q}, \zeta, \overline{A}, \eta, \mathscr{F})$. For Point Robot, the backup policy ζ applies a deceleration force to slow down the velocity of the agent to zero. The advantage threshold is set as $\eta = 0$. The function $\overline{Q}$ is defined through the shaped cost function $\hat{c}$ and the agent dynamics $\hat{\mathcal{P}}$ as



Point Robot Half-Cheetah

Fig. 15 Point Robot: The black point correspond to agent, green circle indicates the desired path, and red lines denote the constraint at horizontal position. Half-Cheetah: the center of the white circle is the link of interest, whose allowed height range is indicated by the white double-headed arrow.

$$\overline{Q} = \mathbb{E}_{\hat{\rho}^{\zeta}}\left[\sum_{t=0}^{\infty} \gamma^t \hat{c}(s_t, a_t)\right],$$

where, $\hat{\rho}^{\zeta}$ is the trajectory distribution based on ζ and $\hat{\mathcal{P}}$.

For Half-Cheetah, the MPC algorithm (Williams *et al.*, 2017) is employed as a backup policy ζ. This MPC algorithm is optimized over a range of H_m time steps, minimizing a cost function that corresponds to a metric of link heights in the range $[h_{min}, h_{max}]$. The function $\overline{Q}$ is defined as

$$\overline{Q} = \mathbb{E}_{\hat{\rho}}\left[\sum_{t=0}^{H_m} \gamma^t \hat{c}(\hat{s}_t, \hat{a}_t) | \hat{a}_{1:H} = \mathrm{MPC}(\hat{s}_1)\right].$$

Moreover, the advantage threshold is set to $\eta = 0.2$.

To demonstrate the performance of the proposed SPPO, several baselines are evaluated for comparison. The first classical algorithm is CPO (Achiam *et al.*, 2017), which is a CMDP-based method that enforces safety constraints by solving restricted optimization problems in each iteration. Another CMDP-based algorithm is PDO (Chow *et al.*, 2017), in which PPO is used as a policy optimization subroutine, while double gradient ascent is performed as a Lagrange multiplier update. In addition, the variant of PDO known as CSC (Bharadhwaj *et al.*, 2021) is also considered as a baesline, in which well-learned conservative critics are applied to filter unsafe behaviors. Furthermore, as an important baseline, SAILR (Wagener *et al.*, 2021), a safe RL method based on advantage-based intervention rules, is thoroughly compared with our approach.

The above baselines and SPPO are all performed on the Ubuntu 18.04 desktop with Intel Core i9 CPU and NVIDIA 3090 GPU. The evaluation results are averaged over 8 (Point Robot) or 5 (Half-Cheetah) random seeds.

Main experimental results

In this section, experiment results demonstrate that SPPO shows improved learning efficiency and well-learned policy performance compared with all baselines, and performs well in terms of safety guarantee during training.

SPPO and four baseline methods are employed to perform the two constrained tasks as shown in **Fig. 15**. The episode returns over the course of training for SPPO and baselines are shown in **Fig. 16** (left) and **Fig. 17** (left), from which it can be observed that (1) the shielding-based safe RL algorithms (SAILR and SPPO) perform better policy performance and higher learning efficiency

Fig. 16 Experimental results for Point Robot. Left: Episode return learning curves that show the mean and standard deviation of the episode return throughout training; Right: Histogram of the cumulative constraint violations upon training completion.

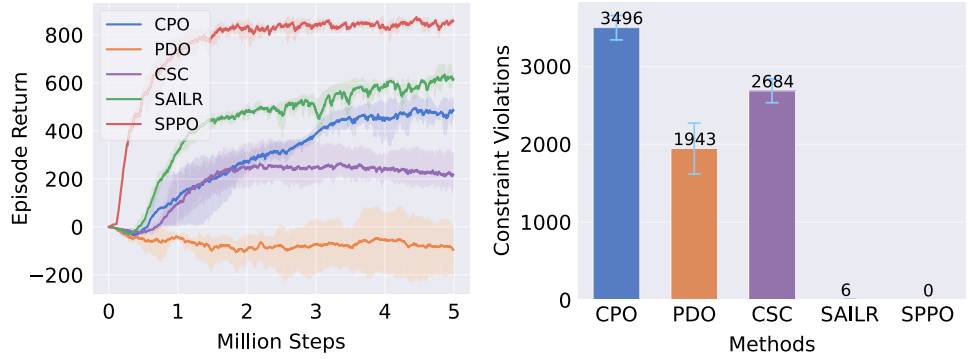


Fig. 17 Experimental results for Half-Cheetah. Left: Episode return learning curves that show the mean and standard deviation of the episode return throughout training; Right: Histogram of the cumulative constraint violations upon training completion.

Table 4 Orders of magnitude of cumulative constraint violations and safety rates for Point Robot

Method	*Violations* (10^*)	*Safety rate (%)*
CPO (Achiam *et al.*, 2017)	$10^2\uparrow$	88.99 ± 5.05
PDO (Chow *et al.*, 2017)	$10^3\uparrow$	74.53 ± 7.53
CSC (Bharadhwaj *et al.*, 2021)	$10^3\uparrow$	69.84 ± 7.28
SAILR (Wagener *et al.*, 2021)	$10^2\downarrow$	97.56 ± 0.95
SPPO (Ours)	$10\downarrow$	$\mathbf{99.73 \pm 0.31}$

Table 5 Orders of magnitude of cumulative constraint violations and safety rates for Half-Cheetah

Method	*Violations* (10^*)	*Safety rate (%)*
CPO (Achiam *et al.*, 2017)	$10^3\uparrow$	54.85 ± 1.31
PDO (Chow *et al.*, 2017))	$10^3\uparrow$	72.17 ± 4.05
CSC (Bharadhwaj *et al.*, 2021)	$10^3\uparrow$	65.10 ± 1.59
SAILR (Wagener *et al.*, 2021)	$10\downarrow$	99.94 ± 0.03
SPPO (Ours)	$\mathbf{0}$	$\mathbf{100.00 \pm 0.00}$

than those classical optimization-based safe RL methods (i.e., CPO, PDO and CSC). This is due to the fact that optimization-based safe RL algorithms can suffer from high computational complexity (CPO) and low numerical stability (PDO, CSC). In contrast, SAILR and SPPO can theoretically guarantee the policy performance as close as possible to unconstrained RL methods while guaranteeing the exploration safety, and thus achieving reliable convergence, as shown in Section "Theoretical Analysis of SPPO"; (2) SPPO exhibits better policy performance and higher learning efficiency compared to SAILR, and this advantage is more significant in Half-Cheetah with higher task complexity. The reason is that SPPO performs guided exploration through planning-oriented sampling, which greatly reduces ineffective interactions between the agent and the environment compared to the stochastic exploration of SAILR, thus improving data efficiency. In addition, as already demonstrated in Subsection "Shielded planning"., the policy-guided planning in SPPO overcomes the shortcomings of the original MPPI algorithm, and avoids falling into local optima as much as possible, hence further improving the performance of the policy. Overall, the above evaluation results indicate that SPPO outperforms the baselines in terms of well-learned policy performance and learning efficiency.

In this subsection, we discuss another important evaluation indicator, i.e., evaluating the safety performance during training by cumulative constraint violations and safety rates. The cumulative constraint violations is the sum of the number of times that the robot violates the constraints while interacting with the environment throughout the training process, and the safety rate is defined as the ratio of unsafe episodes to the total number of episodes. Note that an episode is considered unsafe if there are constraint violations when the robot interacts with the environment.

Fig. 16 (right) and **Fig. 17** (right) visualize the histograms of the cumulative constraint violations for each baseline algorithm as well as SPPO during the training process in Point Roboot and Half-Cheetah environment, respectively. Furthermore, to better compare the safety performance of each baseline with our method during training, we also include the orders of magnitude of cumulative constraint violations and the safety rate of these algorithms for the above constrained tasks in **Tables 4** and **5**. From these experimental results, it can be observed that SAILR and SPPO exhibit much higher performance than the optimization-based safe RL algorithms (CPO, PDO and CSC) in terms of safety during training. This is because optimization-based methods struggle to provide safety guarantees for each agent-environment interaction due to numerical instability and error accumulation. On the contrary, both SAILR and SPPO are shielding-based methods, which ensure safety during each interaction by enforcing interventions, thus guaranteeing safety during the whole exploration process. In addition, although the safety rate of SAILR is close to ours, SPPO has a slightly better safety rate than SAILR overall. Thanks to the planning-oriented exploration, SPPO is able to anticipate future dangerous information and avoid it promptly, which enables SPPO to exhibit better safety performance during training (i.e., nearly 100%).

Limitations

Experimental results demonstrate that, with shielded-planning-oriented sampling, SPPO can improve policy performance and learning efficiency while ensuring exploration safety. We highlight that SPPO is a novel attempt to facilitate safe reinforcement learning using path planning and shielding mechanism, and provides explicit theoretical guarantees. While this has great potential for building a more robust safe RL framework, there remain several issues. The first issue is that SPPO is a model-based safe RL algorithm that performs planning exploration through MPPI based on importance sampling, which is a double-edged sword, i.e., it will consume more computational resources while improving exploration efficiency. More advanced and efficient path planning algorithms for planning-oriented exploration are effective solutions to the above problem. Another potential issue is that it can be challenging to construct appropriate and effective shielding rules for complex RL tasks (e.g., manipulation tasks with complex logic and temporal constraints). Finding more general shielding rule paradigms could be the key to solve this issue.

Conclusion

In this section, we develop a Shielded Planning guided Policy Optimization (SPPO) for data-efficient and safe exploration. In particular, SPPO is equipped with shielded planning for directed exploration and efficient data collection via model predictive path integral (MPPI), as well as an advantage-based shielding rule to keep the above processes safe. A task-oriented parameter optimization (TOPO) method is then used for both policy improvement and the observation-independent latent dynamics enhancement. It is worth mentioning that we provides SPPO with explicit theoretical bounds on training safety, deployment safety, and the learned policy performance. Extensive studies demonstrate that our SPPO outperforms most existing methods in terms of policy performance, learning efficiency and safety performance during training. In future work, we will focus on finding more general shielding rule paradigms to extend this approach towards more complex RL tasks, such as safe manipulation skill learning.

Outlook

In this chapter, we introduce three safe policy optimization methods for reinforcement learning in robotics. These approaches employ data-driven control theory, projection, and shielded planning to ensure safety during exploration and deployment while simultaneously improving learning efficiency and performance. In the future, we believe that safe policy optimization algorithms have a significant role to play in integrating learning-based approaches to physical robotic systems. However, parallel developments in this field have given rise to a range of methodological variants. This chapter focuses on some of the work done by the authors in this area. There is still a lot of work to be done (e.g., more generalized constraint frameworks and more complex task scenarios), and we hope that readers will be inspired by this chapter to continue to move towards building safer and smarter robots.

References

Achiam, J., Held, D., Tamar, A., Abbeel, P., 2017. Constrained policy optimization. Int. Conf. Machin. Learn. 22–31.
Altman, E., 1999. Constrained Markov Decision Processes, vol. 7. CRC press.
Ames, A.D., Xu, X., Grizzle, J.W., Tabuada, P., 2017. Control barrier function based quadratic programs for safety critical systems. IEEE Trans. Autom. Control 62 (8), 3861–3876.
Baier, C., Katoen, J.P., 2008. Principles of Model Checking. MIT Press.
Bharadhwaj, H., Kumar, A., Rhinehart, N., *et al.*, 2021. Conservative safety critics for exploration. In: Proceedings of the International Conference on Learning Representations.
Borkar, V.S., 2005. An actor-critic algorithm for constrained markov decision processes. Syst. Control Lett. 54 (3), 207–213.
Bozkurt, A.K., Wang, Y., Zavlanos, M.M., Pajic, M., 2020. Control synthesis from linear temporal logic specifications using model-free reinforcement learning. Int. Conf. Robot. Autom.. 10349–10355.
Cai, M., Peng, H., Li, Z., Kan, Z., 2021. Learning-based probabilistic LTL motion planning with environment and motion uncertainties. IEEE Trans. Autom. Control 66 (5), 2386–2392. https://doi.org/10.1109/TAC.2020.3006967.
Cai, M., Vasile, C.I., 2021. Safe-critical modular deep reinforcement learning with temporal logic through gaussian processes and control barrier functions. arXiv preprint arXiv:2109.02791.
Carr, S., Jansen, N., Junges, S., Topcu, U., 2023. Safe reinforcement learning via shielding under partial observability. AAAI Conf. Artif. Intell. 37 (12), 14748–14756.
Chow, Y., Ghavamzadeh, M., Janson, L., Pavone, M., 2017. Risk-constrained reinforcement learning with percentile risk criteria. J. Mach. Learn. Res. 18 (1), 6070–6120.
Chua, K., Calandra, R., McAllister, R., Levine, S., 2018. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. Adv. Neural Inf. Proces. Syst. 31.
Co-Reyes, J., Liu, Y., Gupta, A., *et al.*, 2018. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. Int. Conf. Machin. Learn. 1009–1018.
Cohen, M.H., Belta, C., 2023. Safe exploration in model-based reinforcement learning using control barrier functions. Automatica 147, 110684.
Dawson, C., Gao, S., Fan, C., 2023. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. IEEE Trans. Robot.
Dong, L., Li, Y., Zhou, X., Wen, Y., Guan, K., 2021. Intelligent trainer for dyna-style model-based deep reinforcement learning. IEEE Trans. Neural Networks Learn. Sys. 32 (6), 2758–2771.
Eysenbach, B., Salakhutdinov, R.R., Levine, S., 2019. Search on the replay buffer: Bridging planning and reinforcement learning. Adv. Neural Inf. Proces. Syst. 32.
Forgione, M., Piga, D., 2021. Continuous-time system identification with neural networks: Model structures and fitting criteria. Europ. J. Control 59, 69–81.
Fujimoto, S., Hoof, H., Meger, D., 2018. Addressing function approximation error in actor-critic methods. Int. Conf. Mach. Learn. 1587–1596.
Garca, J., Fernández, F., 2015. A comprehensive survey on safe reinforcement learning. J. Mach. Learn. Res. 16 (1), 1437–1480.
Gastin, P., Oddoux, D., 2001. Fast LTL to Büchi automata translation. Int. Conf. Comput. Aided Verif.. 53–65.
Haarnoja, T., Zhou, A., Abbeel, P., Levine, S., 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. Int. Conf. Machin. Learn.. 1861–1870.
Hafner, D., Lillicrap, T., Fischer, I., *et al.*, 2019. Learning latent dynamics for planning from pixels. Int. Conf. Machin. Learn., 2555–2565.
Hansen, N.A., Su, H., Wang, X., 2022. Temporal difference learning for model predictive control. Int. Conf. Machin. Learn., 8387–8406.
Hsu, K.C., Ren, A.Z., Nguyen, D.P., Majumdar, A., Fisac, J.F., 2023. Sim-to-lab-to-real: Safe reinforcement learning with shielding and generalization guarantees. Artif. Intell. 314, 103811.
Icarte, R.T., Klassen, T., Valenzano, R., McIlraith, S., 2018. Using reward machines for high-level task specification and decomposition in reinforcement learning. Int. Conf. Machin. Learn. 2107–2116.
Janner, M., Fu, J., Zhang, M., Levine, S., 2019. When to trust your model: Model-based policy optimization. Adv. Neural Inf. Proces. Syst. 32.
Jayant, A.K., Bhatnagar, S., 2022. Model-based safe deep reinforcement learning via a constrained proximal policy optimization algorithm. Adv. Neural Inf. Proces. Syst. 35, 24432–24445.
Kaiser, L., Babaeizadeh, M., Milos, P., *et al.*, 2019. Model-based reinforcement learning for atari.arXiv preprint arXiv:1903.00374.
Kakade, S., Langford, J., 2002. Approximately optimal approximate reinforcement learning. Int. Conf. Machin. Learn. 267–274.
Le, H., Voloshin, C., Yue, Y., 2019. Batch policy learning under constraints. Int. Conf. Machin. Learn.. 3703–3712.
Lin, T., Jin, C., Jordan, M., 2020, On gradient descent ascent for nonconvex-concave minimax problems. Int. Conf. Machin. Learn., 6083–6093.

.Lin, H., Sun, Y., Zhang, J., Yu Y., 2022. Model-based reinforcement learning with multi-step plan value estimation. arXiv preprint arXiv:2209.05530.
Liu, Z., Cen, Z., Isenbaev, V., *et al.*, 2022. Constrained variational policy optimization for safe reinforcement learning. Int. Conf. Machin. Learn. 13644–13668.
Li, S., Bastani, O., 2020. Robust model predictive shielding for safe reinforcement learning with stochastic dynamics. Int. Conf. Robot. Autom. 7166–7172.
Li, X., Serlin, Z., Yang, G., Belta, C., 2019. A formal methods approach to interpretable reinforcement learning for robotic planning. Sci. Robot. 4 (37).
Modares, A., Sadati, N., Esmaeili, B., Yaghmaie, F.A., Modares, H., 2023. Safe reinforcement learning via a model-free safety certifier. IEEE Trans. Neural Networks Learn. Sys.
Nguyen, T.D., Shu, R., Pham, T., Bui, H., Ermon, S., 2021. Temporal predictive coding for model-based planning in latent space. Int. Conf. Machin. Learn. 8130–8139.
Peng, B., Duan, J., Chen, J., *et al.*, 2022. Model-based chance-constrained reinforcement learning via separated proportional-integral lagrangian. IEEE Trans. Neural Networks Learn. Sys.
Qin, Z., Sun, D., Fan, C., 2022. Sablas: Learning safe control for black-box dynamical systems. IEEE Robot. Autom. Lett. 7 (2), 1928–1935.
Schrittwieser, J., Antonoglou, I., Hubert, T., *et al.*, 2020. Mastering atari, go, chess and shogi by planning with a learned model. Nature 588 (7839), 604–609.
Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P., 2015. Trust region policy optimization. Int. Conf. Machin. Learn., 1889–1897.
Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.
Sekar, R., Rybkin, O., Daniilidis, K., *et al.*, 2020. Planning to explore via self-supervised world models. Int. Conf. Machin. Learn. 8583–8592.
Selim, M., Alanwar, A., Kousik, S., *et al.*, 2022. Safe reinforcement learning using black-box reachability analysis. IEEE Robot. Autom. Lett. 7 (4), 10665–10672.
.Tassa, Y., Erez, T., Todorov, E., 2012. Synthesis and stabilization of complex behaviors through online trajectory optimization. IEEE/RSJ Int. Conf. Intell. Robot. Syst., 4906–4913.
.Tessler, C., Mankowitz, D.J., Mannor, S., 2018. Reward constrained policy optimization. arXiv preprint arXiv:1805.11074.
Thananjeyan, B., Balakrishna, A., Nair, S., *et al.*, 2021. Recovery rl: Safe reinforcement learning with learned recovery zones. IEEE Robot. Autom. Lett. 6 (3), 4915–4922.
Toro Icarte, R., Klassen, T.Q., Valenzano, R., McIlraith, S.A., 2018. Teaching multiple tasks to an rl agent using ltl. Proc. Int. Joint Conf. Auton. Agents Multiagent Syst. 452–461.
Tunyasuvunakool, S., Muldal, A., Doron, Y., *et al.*, 2020. dm_control: Software and tasks for continuous control. Softw. Imp. 6, 100022.
Vaezipoor, P., Li, A.C., Icarte, R.A.T., Mcilraith, S.A., 2021. Ltl2action: Generalizing ltl instructions for multi-task rl. Int. Conf. Machin. Learn. 10497–10508.
Wabersich, K.P., Zeilinger, M.N., 2022. Predictive control barrier functions: Enhanced safety mechanisms for learning-based control. IEEE Trans. Autom. Control.
Wagener, N.C., Boots, B., Cheng, C.A., 2021. Safe reinforcement learning using advantage-based intervention. Int. Conf. Machin. Learn.. 10630–10640.
Wang, H., Zhang, H., Li, L., Kan, Z., Song, Y., 2023. Task-driven reinforcement learning with action primitives for long-horizon manipulation skills. IEEE Trans. Cybern.
Williams, G., Aldrich, A., Theodorou, E., 2015. Model predictive path integral control using covariance variable importance sampling. arXiv preprint arXiv:1509.01149.
Williams, G., Wagener, N., Goldfain, B., *et al.*, 2017. Information theoretic mpc for model-based reinforcement learning. Int. Conf. Robot. Autom. 1714–1721.
Yang, L., Ji, J., Dai, J., *et al.*, 2022. Constrained update projection approach to safe policy optimization. Adv. Neural Inf. Proces. Syst. 35, 9111–9124.
Yang, T.Y., Rosca, J., Narasimhan, K., Ramadge, P.J., 2019. Projection-based constrained policy optimization. ICLR - Int. Conf. Learn. Represent.
Yang, T.Y., Rosca, J., Narasimhan, K., Ramadge, P.J., 2021. Accelerating safe reinforcement learning with constraint-mismatched baseline policies. Int. Conf. Machin. Learn. 11795–11807.
Ye, W., Liu, S., Kurutach, T., Abbeel, P., Gao, Y., 2021. Mastering atari games with limited data. Adv. Neural Inf. Proces. Syst. 34, 25476–25488.
Yu, D., Ma, H., Li, S., Chen, J., 2022. Reachability constrained reinforcement learning. Int. Conf. Machin. Learn., 25636–25655.
Zhang, Y., Liang, X., Li, D., *et al.*, 2022. Barrier lyapunov function-based safe reinforcement learning for autonomous vehicles with optimized backstepping. IEEE Trans. Neural Netw. Learn. Sys.
Zhang, M., Vikram, S., Smith, L., *et al.*, 2019. Solar: Deep structured representations for model-based reinforcement learning. Int. Conf. Machin. Learn., 7444–7453.
Zhang, L., Zhang, Q., Shen, L., *et al.*, 2023. Evaluating model-free reinforcement learning toward safety-critical tasks. AAAI Conf. Artif. Intell. 37 (12), 15313–15321.
Zhou, Z., Oguz, O.S., Leibold, M., Buss, M., 2023. Learning a low-dimensional representation of a safe region for safe reinforcement learning on dynamical systems. IEEE Trans. Neural Netw. Learn. Sys. 34 (5), 2513–2527.