Shielded Planning Guided Data-Efficient and Safe Reinforcement Learning

Hao Wang, Jiahu Qin, *Senior Member, IEEE*, and Zhen Kan, *Senior Member, IEEE*

***Abstract*— Safe reinforcement learning (RL) has shown great potential for building safe general-purpose robotic systems. While many existing works have focused on post-training policy safety, it remains an open problem to ensure safety during training as well as to improve exploration efficiency. Motivated to address these challenges, this work develops shielded planning guided policy optimization (SPPO), a new model-based safe RL 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.**

***Index Terms*— Data-efficient exploration, model-based reinforcement learning (RL), safe policy optimization, shielded planning.**

NOMENCLATURE

Symbol	Description
$d(s_t, u_t)$	Dynamics model with s_t and u_t denoting the state and control input at time t, respectively.
$\Gamma = (u_0, u_1, \ldots, u_H)$	Control input trajectory.
Φ_Γ	Return of a sampled trajectory Γ.
$\{\Phi^\star_{\Gamma,i}\}_{i=1}^k$	Top-k returns of sampled trajectories.
μ^j, σ^j	Mean and standard deviation at the jth iteration.
$\mathcal{M}$	Reward-based MDP.
$\overline{\mathcal{M}}$	Cost-based MDP.
$\mathcal{M}_c$	Constrained Markov decision process (CMDP).
$\widetilde{\mathcal{M}}$	Absorbing MDP.
$\mathcal{S}$, $\mathcal{A}$, P, r, c, γ	State space, action apace, transition dynamics, reward function, cost function, and discount factor, respectively.
$\widetilde{\mathcal{S}}$, $\widetilde{P}$, $\widetilde{r}$	State space, transition dynamics, and reward function of the absorbing MDP, respectively.
$s_\dagger$	Absorbing state.
S_{safe}, S_{unsafe}	Safe state space and unsafe state space, respectively.
V^π	Discounted return.
$\overline{V}^\pi$	Discounted cost.
$\mathcal{G}$	Shielding rule.
$\overline{Q}$, ζ, $\overline{A}$, η, and $\mathcal{I}$	State-action cost value estimator, backup policy, advantage-like function, advantage threshold, and shielding set, respectively.
π_θ, h_ϕ, d_ψ, R_χ, and Q_ω	Policy network, representation network, latent dynamics network, reward network, and value network, respectively.
θ, ϕ, ψ, χ, and ω	Network parameters of policy, representation, latent dynamics, reward, and value, respectively.
Π_θ	Planning-oriented policy.
Π'_θ	Shielded safe policy.
$\mathcal{B}$	Replay buffer for storing the experience of running Π'_θ in $\mathcal{M}$.
$\widetilde{\mathcal{B}}$	Replay buffer for storing the experience of running Π_θ in $\widetilde{\mathcal{M}}$.
J	Local weighted objective function.
L	Single-step loss function.
L_r, L_v, and L_c	Reward prediction loss, value prediction loss, and latent state consistency loss, respectively.
c_1, c_2, and c_3	Tuning weights balancing the losses.

Manuscript received 23 August 2023; revised 3 December 2023; accepted 23 January 2024. Date of publication 6 February 2024; date of current version 6 February 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62173314 and Grant U2013601. *(Corresponding author: Zhen Kan.)*

The authors are with the Department of Automation, University of Science and Technology of China, Hefei, Anhui 230026, China (e-mail: zkan@ustc.edu.cn).

Digital Object Identifier 10.1109/TNNLS.2024.3359031

I. 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 an Unmanned Aerial Vehicle (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 [1], [2], [3], [4], or developing safe RL [5], [6], [7] to synthesize safe policies, how to improve exploration efficiency or ensure safety during training 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: 1) planning [8], [9], [10], [11], [12], [13], [14] and 2) improving the sample efficiency of model-free methods via the learned models [3], [15], [16]. They both fully exploit the key advantages of model-based learning and have been extremely successful in application fields, such as game-playing [13], [17] and continuous control [9], [16], [18], [19]. 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, authors [13], [16], and [20] 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 RL [5] 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) [21] 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 [22], [23], [24] are popular frameworks inspired by constrained optimization algorithms. There have been related works focusing on developing such frameworks. The works of [25], [26], and [27] solve the stochastic nonconvex saddle point problem based on first-order primal-dual optimization (PDO). Despite the fact that such methods can eventually synthesize a safe policy, they struggle to guarantee the safety of the training process. To solve this problem, authors [23], [24], [28], [29], and [30] perform safety constraints in each iteration by solving restricted optimization problems to ensure safety during training. However, these methods are less robust compared with the unconstrained RL methods, due to the numerical instability associated with solving stochastic nonconvex problems [31]. While [32] and [33] 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 [34], [35], [36], [37], [38]. The most prominent methods are those based on the control barrier function (CBF) [6], [7], [39], [40], [41] and reachability [42], [43], [44]. 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.

Can we instead augment safe RL with the advantages of model-based planning to achieve data-efficient and safe policy optimization, not only in deployment but also during the training process? Motivated to address this issue, in this work, we propose shielded planning guided policy optimization (SPPO), a new policy optimization algorithm that augments RL with model predictive control and shielding mechanism to improve the exploration efficiency and ensure the safety during training. On one hand, 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. In contrast to the random exploration of classical RL, shielded planning can greatly reduce invalid interactions between the agent and the environment via guided exploration, thus improving data efficiency. Furthermore, based on the shielding rule, SPPO transforms the CMDP problem into an unconstrained MDP problem, which is easier to implement and more stable than typical constraint-based optimization methods. On the other hand, SPPO employs a task-oriented parameter optimization (TOPO) method for policy improvement, as well as the observation-independent latent dynamics enhancement. TOPO completely avoids the dependence of model learning on environmental observations by optimizing the policy and environment model parameters with potential latent state consistency loss, overcoming the challenge that most model-based RL methods require the network to simulate everything in the environment to accurately predict the future. In addition, it is worth noting that our SPPO provides explicit theoretical guarantees, i.e., clear theoretical bounds for training safety, deployment safety, and the learned policy performance.

A. Contributions

The main contributions of this work are summarized as follows.

1) We introduce an SPPO algorithm that augments RL with MPPI and shielding rules to improve exploration efficiency and ensure policy safety even during training.
2) We construct absorbing MDP based on the shielding rule, which in turn transforms a constrained optimization

problem into an unconstrained one, making the optimization process simpler, more stable, and easier to solve.

3) We provide explicit theoretical guarantees for SPPO, including clear theoretical bounds for training safety, deployment safety, and the learned policy performance.
4) A thorough empirical study validates that SPPO significantly outperforms baselines in terms of policy performance, learning efficiency, and safety performance during training.

II. Preliminaries and Problem Formulation

In this section, we first review relevant preliminaries and then formally formulate the problem to be addressed. To enhance the readability, we summarize the main symbol appointment of this article in Nomenclature.

A. Model Predictive Path Integral

MPPI is a model-based policy optimization algorithm that iteratively updates system parameters through importance sampling [45]. 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. [16], 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 being 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 follows:

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

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

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

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

where $\Omega_i = e^{\tau(\Phi^\star_{\Gamma,i} - \max_g(\Phi^\star_{\Gamma,g}))}$, τ is a temperature parameter that controls the weighted "sharpness," and $\Gamma_i^\star$ indicates the ith top-k trajectory corresponding to return estimate $\Phi^\star_\Gamma$. 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.

B. Constrained Markov Decision Process

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 CMDP [21], [25]. In particular, given a state space S, let $S_{\text{safe}}, S_{\text{unsafe}} \subset S$ be the safe and unsafe states, respectively, such that $S_{\text{safe}} = S \setminus S_{\text{unsafe}}$, the unsafe states can be rewritten as $S_{\text{unsafe}} := \{s_\triangleright, s_\circ\}$, 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 defined as follows.

Definition 1: 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 $S = S_{\text{safe}} \cup S_{\text{unsafe}}$ with $S_{\text{unsafe}} = \{s_\triangleright, s_\circ\}$ is a state space, $\mathcal{A}$ is an action apace, $P(s' \mid s, a)$ is the transition dynamics, $r(s, a) \in [0, 1]$ is a reward function where the reward on S_{unsafe} is 0, $c(s, a) = \mathbb{1}\{s = s_\triangleright\}$ is the cost function where $\mathbb{1}$ denotes the indicator function, and $\gamma \in [0, 1]$ is a discount factor.

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

$$\begin{aligned} \pi^* = &\arg\max_\pi V^\pi \\ &\text{s.t.} \quad \overline{V}^\pi \le \delta \end{aligned} \tag{4}$$

to maximize the expected discounted return $V^\pi = \mathbb{E}_\pi[\sum_{i=0}^\infty \gamma^t r(s_t, a_t)]$ over the reward-based MDP $\mathcal{M}$ while satisfying the discounted cost $\overline{V}^\pi = \mathbb{E}_\pi[\sum_{i=0}^\infty \gamma^t c(s_t, a_t)] \le \delta$ over the cost-based MDP $\overline{\mathcal{M}}$. As discussed in [46], requiring the agent to find a policy with a 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 (4).

C. Challenges and Problem Formulation

There are two main challenges to solve the CMDP problem in (4). The first challenge is to ensure the agent's safety during the training process. Although the CMDP problem in (4) has been extensively studied [23], [25], [26], [28], 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 [35], [36], [37], [38], 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 performs a large number of invalid interactions

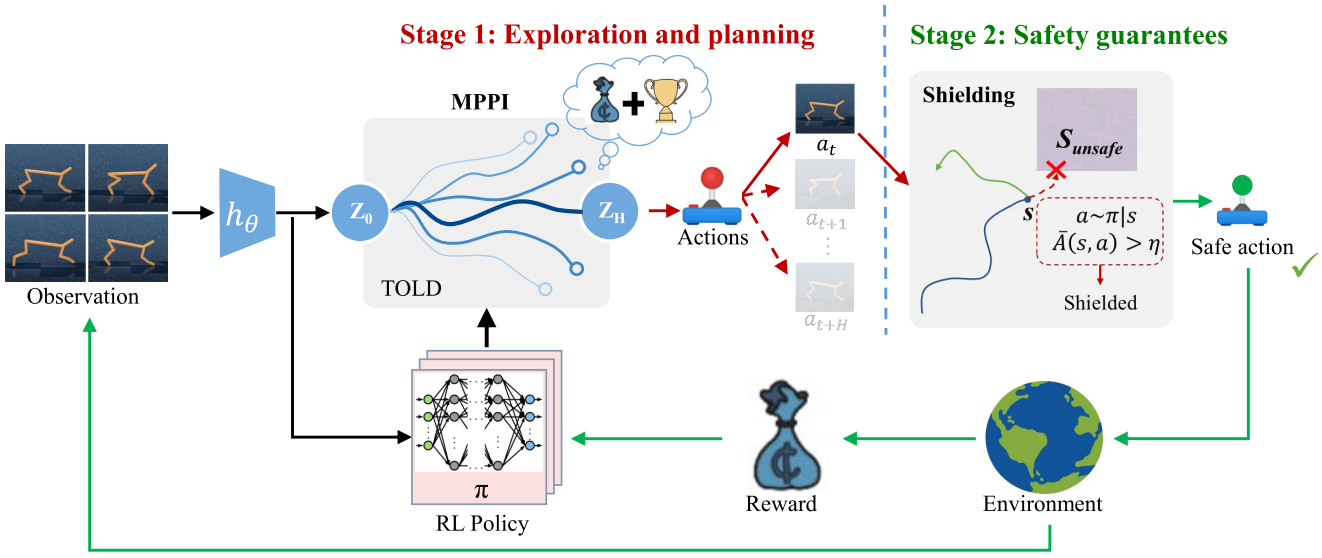


Fig. 1. Framework of SPPO that utilizes a shielded planning to improve exploration efficiency and ensure policy safety during training. SPPO consists of two stages. Stage 1: MPPI is employed for guided exploration and efficient data collection. Stage 2: advantage-based shielding rule is applied to keep the exploration process safe.

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 (4) while ensuring exploration safety along a state-action sequence $\mathcal{L} = (s_0, a_0, s_1, a_1, \ldots, s_\infty)$, where, at each decision step t, $s_{t+1} \sim P(\cdot \mid s_t, a_t)$ with $a_t \sim \Pi_\theta(\cdot \mid s_t)$.

III. Shielded Planning Guided Policy Optimization

In this section, we provide a new data-efficient and safe RL method, namely, SPPO, to address Problem 1 enabling agents to explore safely and efficiently. We first describe the overview of SPPO in Section III-A and then explain the technical details in Sections III-B–III-E.

A. 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. 1, 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 TOPO 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 (TOLD) 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 pseudocode of SPPO is outlined in Algorithm 1, including the exploration phase (lines 3–9) and the training phase (lines 10–13). In the exploration phase, a shielded planning algorithm (Algorithm 2) is first 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 dataset $\mathcal{B}$, while the absorbing action $\widetilde{a}$ running in $\widetilde{\mathcal{M}}$ collects the safety dataset $\widetilde{\mathcal{B}}$ (lines 5–8). In the training phase, $\widetilde{\mathcal{B}}$ is fed into a TOPO algorithm (Algorithm 3) 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 Θ^* and policy $\hat{\Pi}^*$ (lines 15 and 16).

In the following, we first present the advantage-based shielding rules and the absorbing MDP in Sections III-B and III-C, and detail the shielded planning algorithm based on the shielding mechanism and MPPI in Section III-D. The TOPO is introduced in Section III-E.

B. 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. [46], given a CMDP $\mathcal{M}_c$ in Definition 1, the corresponding shielding rule is defined as a five-tuple $\mathcal{G} = (\overline{Q}, \zeta, \overline{A}, \eta, \mathcal{I})$, where $\overline{Q} : S_{\text{safe}} \times \mathcal{A} \to [0, 1]$ 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 $\pi(\cdot \mid s)$ and the backup policy $\zeta(\cdot \mid s)$, $\eta \in [0, 1]$ is an advantage threshold, and $\mathcal{I} = \{(s, a) \in S_{\text{safe}} \times \mathcal{A} : \overline{A}(s, a) > \eta\}$

Algorithm 1 SPPO

procedure INPUT: $(\mathcal{B}, \widetilde{\mathcal{B}}, \mathcal{M}$, and $\widetilde{\mathcal{M}})$
  Output: $\Theta^* = (\theta^*, \phi^*, \psi^*, \chi^*, \omega^*)$, Π^*: Optimal network parameters and policy
  Initialization: $\Theta_0 = (\theta_0, \phi_0, \psi_0, \chi_0, \omega_0)$: initial network parameters
  while episode not terminated **do**
    for step $t = 0, 1, \ldots, T$ **do** ▷ Exploration Phase
      $a_t^{safe}, \widetilde{a} \sim ShieldedPlanning()$ ▷ Algorithm 2
      $(s_{t+1}, r_t) \sim \mathcal{T}\left(\cdot \mid s_t, a_t^{safe}\right), r\left(\cdot \mid s_t, a_t^{safe}\right)$
      $(\widetilde{s}_{t+1}, \widetilde{r}_t) \sim \widetilde{\mathcal{T}}(\cdot \mid s_t, \widetilde{a}_t), \widetilde{r}(\cdot \mid s_t, \widetilde{a}_t)$
      $\mathcal{B} \leftarrow \mathcal{B} \cup (s_t, a_t, s_{t+1}, r_t)$
      $\widetilde{\mathcal{B}} \leftarrow \widetilde{\mathcal{B}} \cup (s_t, a_t, \widetilde{s}_{t+1}, \widetilde{r}_t)$
    end for
    for training step $t = 0, 1, \ldots, K$ **do** ▷ Training Phase
      $\Theta \leftarrow TOPO(\widetilde{B})$ ▷ Algorithm 3
      $\mathcal{G} \leftarrow UpdateShieldingRule(\mathcal{B})$ (Optional)
    end for
  end while
  $\Theta^* \leftarrow GetOptimizeParameters()$
  $\hat{\Pi}^* \leftarrow GetOptimizePolicy(\Theta^*)$
end procedure

is a shielding set. In this work, when $\overline{A}(s,a) > \eta \geq 0$, i.e., $\overline{Q}(s,a) > \overline{Q}(s,a_\zeta)$, (s,a) is considered unsafe, and thus, the shielding set $\mathcal{I}$ is regarded as an unsafe set.

Proposition 1: If $\eta \geq 0$, $\forall (s,a) \in \mathcal{I}$, there always exists some $a' \in \mathcal{A}$ satisfying $(s,a') \notin \mathcal{I}$.

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

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

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

Proposition 1 guarantees the existence of safe actions, which is essential for the unconstrained MDP simplification in Section III-D. Given a policy π, the shielded policy $\pi' = \mathcal{G}(\pi)$ can be synthesized as follows:

$$\pi'(a \mid s) = \begin{cases} \pi(a \mid s), & \text{if}(s,a) \notin \mathcal{I} \\ \zeta(a \mid s) \cdot (1 - w(s)), & \text{otherwise} \end{cases} \tag{5}$$

where $w(s)$ denotes the probability that, when $(s,a) \notin \mathcal{I}$, π' still take actions in $\mathcal{I}$. As indicated in (5), when sampling an action a from $\pi'(a \mid s)$, an action a' is first sampled from $\pi(\cdot \mid s)$. If $(s,a') \notin \mathcal{I}$, $a = a'$, otherwise, a samples from $\zeta(\cdot \mid s)$. Since the shielding set $\mathcal{I}$ only allows actions that are not less safer 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 (5) may still take actions in $\mathcal{I}$, depending on the probability assigned to these actions by ζ. This design avoids the problem that direct intervention rules based on Q-based functions [28], [36] 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 2 (β-Adoptable Shielding Rule): A shielding rule $\mathcal{G} = (\overline{Q}, \zeta, \overline{A}, \eta, \mathcal{I})$ is *β-adoptable* if $\overline{Q}(s,a) \in [0, \gamma]$ and for some $\beta \geq 0$, the following holds for all $s \in S_{\text{safe}}$ and $a \in \mathcal{A}$:

$$\overline{Q}(s,a) + \beta \geq c(s,a) + \gamma \mathbb{E}_{s' \mid s,a}\left[\overline{Q}\left(s', \zeta\right)\right] \tag{6}$$

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

Proposition 2: 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 $\overline{Q}^\zeta$ being the state-action cost of ζ on $\overline{\mathcal{M}}$, then $\mathcal{G} = (\overline{Q}^\zeta, \zeta, \overline{A}, \eta, \mathcal{I})$ or $\mathcal{G} = (\overline{Q}^\zeta, \zeta^+, \overline{A}, \eta, \mathcal{I})$ 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 $\mathcal{G}^* = (\overline{Q}^*, \overline{\pi}^*, \overline{A}^*, \eta, \mathcal{I}^*)$ is adoptable.
3) Approximation: Consider a β-adoptable $\mathcal{G} = (\overline{Q}, \zeta, \overline{A}, \eta, \mathcal{I})$ and $\hat{Q} \in [0, \gamma]$ for all $s \in S_{\text{safe}}$ and $a \in \mathcal{A}$, if $\|\hat{Q} - \overline{Q}\|_\infty \leq \delta$, then $\hat{G} = (\hat{Q}, \zeta, \hat{A}, \eta, \hat{I})$ is $(\beta + (1+\gamma)\delta)$-adoptable.

The proof of Proposition 2 is omitted, since it is a trivial extension of [46, Proposition 3]. Based on Proposition 2, 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, $\mathcal{G}^* = (\overline{Q}^*, \overline{\pi}^*, \overline{A}^*, 0, \mathcal{I}^*)$ 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.

C. 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 $\mathcal{I}$ introduced in Section III-B, the absorbing MDP corresponding to $\mathcal{M}$ is constructed as $\widetilde{\mathcal{M}} = (\widetilde{\mathcal{S}}, \mathcal{A}, \widetilde{P}, \widetilde{r}, \gamma)$, where $\widetilde{\mathcal{S}} = \mathcal{S} \cup \{s_\dagger\}$. The transition reward is modified as follows:

$$\widetilde{r}(s,a) = \begin{cases} \widetilde{R}, & (s,a) \in \mathcal{I} \\ 0, & s = s_\dagger \\ r(s,a), & \text{otherwise} \end{cases} \tag{7}$$

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

$$\widetilde{P}\left(s' \mid s,a\right) = \begin{cases} \mathbb{1}\left\{s' = s_\dagger\right\}, & (s,a) \in \mathcal{I} \text{ or } s = s_\dagger \\ P\left(s' \mid s,a\right), & \text{otherwise.} \end{cases} \tag{8}$$

Without the loss of generality, when the agent is in an absorbing state $s_\dagger$, the policy $\pi(a \mid s_\dagger)$ 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 with the original one. When an agent performs some $(s,a) \in \mathcal{I}$, 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

Algorithm 2 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 $j = 1, 2, \ldots, J$ **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 $(a_t, a_{t+1}, \ldots, a_{t+H})$ **do**
7: **for** step $t = 0, 1, \ldots, H-1$ **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: $\mu^j = \frac{\sum_{i=1}^k \Omega_i \Gamma_i^\star}{\sum_{i=1}^k \Omega_i}$, $\sigma^j = \max\left(\sqrt{\frac{\sum_{i=1}^k \Omega_i \left(\Gamma_i^\star - \mu^j\right)^2}{\sum_{i=1}^k \Omega_i}}, \epsilon\right)$
14: **end for**
15: $a \sim \mathcal{N}\left(\mu^J, (\sigma^J)^2\mathbf{I}\right)$
16: $a^{safe} = \begin{cases} a, & \text{if}(s,a) \notin \mathcal{I} \\ a_\zeta \sim [\zeta(\cdot \mid s) \cdot (1 - w(s))], & \text{otherwise} \end{cases}$
17: $\widetilde{a} = \begin{cases} a, & \text{if } a^{safe} = a \\ a_T, & \text{if } a_T^{safe} = a_\zeta \\ rand(A), & \text{if } a_T^{safe} = a_\zeta \text{ and } t \geq T+1 \end{cases}$
return a^{safe},$\widetilde{a}$
18: **end procedure**

$\widetilde{\mathcal{M}}$ runs on $\mathcal{M}$. Since we simulate the experience of policy Π_θ in $\widetilde{\mathcal{M}}$ by running policy Π'_θ in $\mathcal{M}$ and collecting samples for SPPO as shown in Sections III-D and III-E, 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 IV.

D. 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 Algorithm 1, including policy guided planning based on MPPI (lines 3–14) as well as safe shielding based on advantage-based shielding mechanisms (lines 15–17).

1) Policy Guided Planning: Before presenting the shielded planning algorithm, we first introduce the TOLDs [16]

$$\begin{aligned} \text{Representation:} \quad & z_t = h_\phi(s_t) \\ \text{Latent dynamics:} \quad & z_{t+1} = d_\psi(z_t, a_t) \\ \text{Reward:} \quad & \hat{r}_t = R_\chi(z_t, a_t) \\ \text{Value:} \quad & \hat{q}_t = Q_\omega(z_t, a_t) \\ \text{Policy:} \quad & \hat{a}_t = \pi_\theta(z_t) \end{aligned} \tag{9}$$

where $\theta, \phi, \psi, \chi,$ and ω 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 III-E. 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.

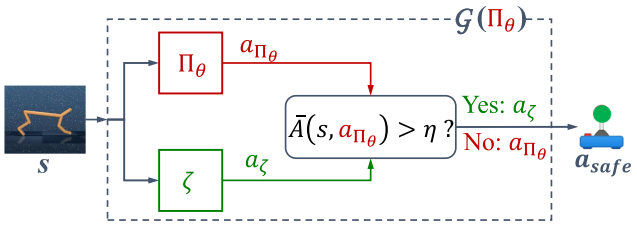


Fig. 2. Construction of shielded policy. The agent executes the action a_{Π_θ} sampled from Π_θ, if $(s, a_{\Pi_\theta}) \notin \mathcal{I}$, i.e., the agent is safe to execute a_{Π_θ} in state s. Otherwise, it executes the safe a_ζ sampled from the backup policy ζ.

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

$$\Phi_\Gamma = \mathbb{E}_\Gamma\left[\gamma^H Q_\omega(z_H, a_H) + \sum_{t=0}^{H-1} \gamma^t R_\chi(z_t, a_t)\right] \tag{10}$$

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 follows:

$$\sigma^j = \max\left(\sqrt{\frac{\sum_{i=1}^k \Omega_i \left(\Gamma_i^\star - \mu^j\right)^2}{\sum_{i=1}^k \Omega_i}}, \epsilon\right) \tag{11}$$

where $\epsilon \in \mathbb{R}_+$ 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 $\mathcal{N}(\mu^{j-1}, (\sigma^{j-1})^2\mathbf{I})$ 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 the 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.

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

$$a^{\text{safe}} = \begin{cases} a, & \text{if}(s,a) \notin \mathcal{I} \\ a_\zeta \sim [\zeta(\cdot \mid s) \cdot (1 - w(s))], & \text{otherwise} \end{cases} \tag{12}$$

where $(s, a) \in S_{\text{safe}} \times \mathcal{A}$, a_ζ is sampled according to the updated backup policy, i.e., $\zeta(\cdot \mid s) \cdot (1 - w(s))$ where $w(s)$ denotes the probability that a^{safe} still takes actions in $\mathcal{I}$ when $(s, a) \notin \mathcal{I}$. As indicated in (12), the agent executes the action a sampled from $\mathcal{N}(\mu^J, (\sigma^J)^2\mathbf{I})$ if $(s, a) \notin \mathcal{I}$, 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 (12), as illustrated in Fig. 2.

Given an MDP $\mathcal{M}$, the corresponding absorbing MDP $\widetilde{\mathcal{M}}$ can be constructed as discussed in Section III-C. Specifically, $\widetilde{\mathcal{S}} = \mathcal{S} \cup \{s_\dagger\}$, $\widetilde{r}$ and $\widetilde{P}$ are shown in (7) and (8), respectively. The policy Π_θ running in $\widetilde{\mathcal{M}}$ can be constructed by Π'_θ in $\mathcal{M}$ according to

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

where $\widetilde{a} \sim \Pi_\theta(\cdot \mid \widetilde{s})$, $a^{\text{safe}} \sim \Pi'_\theta(\cdot \mid s)$ and T is the first time of shielding intervention. Thus, we can simulate the experience of Π_θ in $\widetilde{\mathcal{M}}$ by running Π'_θ in $\mathcal{M}$ and collecting samples $\mathcal{B}$ and $\widetilde{\mathcal{B}}$ until the shielding intervention is triggered. For instance, if $\mathcal{L} = (s_0, a_0, s_1, a_1, \ldots, s_T, a'_T, \ldots)$ is a state-action sequence generated by running Π'_θ in $\mathcal{M}$, where a'_T is the first action sampled from the backup policy $\zeta(\cdot \mid s_T)$, and a_T denotes the overridden action sampled from $\Pi_\theta(\cdot \mid s_T)$. Then, the state-action sequence $\widetilde{\mathcal{L}} = (s_0, a_0, s_1, a_1, \ldots, s_T, a_T, \widetilde{s}_{T+1}, \widetilde{a}_{T+1}, \ldots)$ can be constructed to simulate running Π_θ in $\widetilde{\mathcal{M}}$, where $\widetilde{s}_\tau = s_\dagger$ and $\widetilde{a}_\tau \in \mathcal{A}$ are 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 IV.

E. Task-Oriented Parameter Optimization

As introduced in Section III-D, our shielded planning is based on TOLD. Since TOLD learns to model only the elements of the environment that predicts rewards, our parameter optimization approach is oriented toward task goal completion, which we refer to as TOPO.

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

$$J(\Theta, \Gamma) = \sum_{i=t}^{t+H} \lambda^{i-t} L(\Theta, \Gamma_i) \tag{13}$$

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

$$L(\Theta, \Gamma_i) = c_1 L_r(\chi, \Gamma_i) + c_2 L_v(\theta, \omega, \Gamma_i) + c_3 L_c(\phi, \psi, \Gamma_i) \tag{14}$$

which includes the reward prediction loss

$$L_r(\chi, \Gamma_i) = \| R_\chi(z_i, a_i) - r_i \|_2^2 \tag{15}$$

the value prediction loss

$$L_v(\theta, \omega, \Gamma_i) = \| Q_\omega(z_i, a_i) - (r_i + \gamma Q_{\omega^-}(z_{i+1}, \pi_\theta(z_{i+1}))) \|_2^2 \tag{16}$$

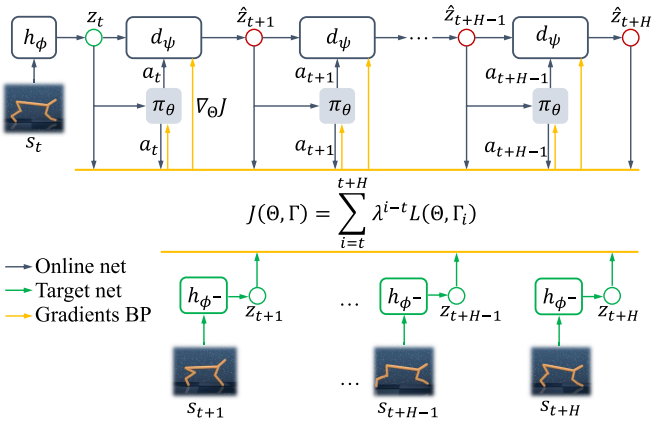


Fig. 3. Update of TOPO. In each update episode, a H-step trajectory $\Gamma_{t:t+H}$ is sampled from the replay buffer. The first state s_t is mapped to an latent representation z_t, and then the following latent states $\hat{z}_{t+1}, \hat{z}_{t+2}, \ldots, \hat{z}_{t+H}$, as well as the rewards, values, 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.

and the latent state consistency loss

$$L_c(\phi, \psi, \Gamma_i) = \| d_\psi(z_i, a_i) - h_{\phi^-}(s_{i+1}) \|_2^2 \tag{17}$$

where $c_{1:3}$ in (14) are the tuning weights balancing the losses.

Apparently, (13) is convex and can be solved by the gradient descent method. Algorithm 3 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 (13)–(17) (lines 5–11). Note that the recurrent prediction is completely in the latent space, i.e., $z_t = h_\phi(s_t)$, $z_{t+1} = d_\psi(s_t, a_t)$, $\ldots$, $z_{t+H} = d_\psi(s_{t+H-1}, a_{t+H-1})$. Only the first observation s_t is encoded with h_ϕ, and the others are predicted by d_ψ. Based on J, the parameters Θ are updated by back propagation through the gradient descent method (line 12), and the target network parameters Θ^- are updated by an exponential moving average of Θ (line 13). The above process is repeated until the update budget is exhausted, and then, the optimized Θ and Θ^- are returned. The update process of TOPO is graphically shown in Fig. 3.

Most of the existing model-based RL methods usually predict future observations directly [9], [13], [17]. 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 article, 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 in Section IV.

IV. Theoretical Analysis of SPPO

In this section, we present the theoretical guarantees of SPPO, including the safety of shielded planning policy (i.e.,

Algorithm 3 Task-Oriented Parameter Optimization

procedure INPUT: ($\eta, \tau, \lambda, \widetilde{\mathcal{B}}$: learning rate, coefficients, replay buffer of $\widetilde{\mathcal{M}}$)
  Output: $\Theta = (\theta, \phi, \psi, \chi, \omega), \Theta^{-}$: optimized network parameters, target network parameters
  Initialization: $J = 0$: initialize J for loss accumulation
  for num updates per episode **do**
    $\{s_t, a_t, r_t, s_{t+1}\}_{t:t+H} \sim \widetilde{\mathcal{B}}$
    $z_t = h_\phi(s_t)$
    for $i = t, t+1, \ldots, t+H$ **do**
      $\hat{r}_i = R_\chi(z_i, a_i)$
      $\hat{q}_i = Q_\omega(z_i, a_i)$
      $z_{i+1} = d_\psi(z_i, a_i)$
      $\hat{a}_i = \pi_\theta(z_i)$
      $J \leftarrow J + \lambda^{i-1} L(z_{i+1}, \hat{r}_i, \hat{q}_i, \hat{a}_i)$
    end for
    $\Theta \leftarrow \Theta - \frac{1}{H}\eta \nabla_\Theta J$
    $\Theta^{-} \leftarrow (1-\tau)\Theta^{-} + \tau\Theta$
  end for
    return Θ, Θ^{-}
end procedure

the safety guarantee in the 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.

A. 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).

Theorem 1 (Safety of Shielded Planning Policy): If $\mathcal{G} = (\overline{Q}, \zeta, \overline{A}, \eta, \mathcal{I})$ 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 2 (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 $P_{\mathcal{G}}(\pi^*) = (1-\gamma)\sum_{h=0}^{\infty} \gamma^h \Pr(\xi^h \cap \mathcal{I} \neq \emptyset \mid \pi^*, \mathcal{M})$ denotes the probability that π^* visits $\mathcal{I}$ in $\mathcal{M}$.

Since Theorems 1 and 2 are trivial extensions of Theorem 2 and [46, Proposition 7], their proofs are omitted here. Theorem 1 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 $\mathcal{G} = (\overline{Q}, \zeta, \overline{A}, \eta, \mathcal{I})$ 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 2 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 Theorems 1 and 2, the following corollary is developed.

Corollary 1 (Performance and Safety of the Learned Policy): If $\widetilde{R} = -1$ and $\mathcal{G} = (\overline{Q}, \zeta, \overline{A}, \eta, \mathcal{I})$ 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 2, one has

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

For the safety bound, one has

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

where the second inequality is derived from Theorem 1. ■

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.

B. 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 RL 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 article, 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 RL has been widely studied, existing CMDP-based safe RL methods [23], [25], [26], [28] hardly guarantee the safety of the exploration process and suffer from the computational cost and poor numerical stability, while control-theoretic techniques [35], [36], [37], [38] 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.

V. Experiments

In this section, SPPO is evaluated against previous works. Particularly, we investigate the following.

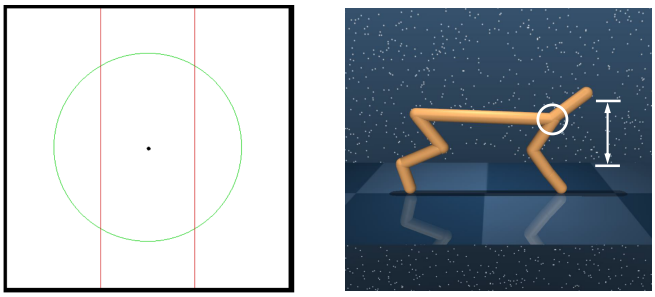
1) *Policy Performance:* Whether the policy learned by SPPO can obtain higher returns than the baselines.
2) *Learning Efficiency:* Whether SPPO outperforms previous methods in terms of exploration efficiency.
3) *Training Safety:* Whether SPPO can guarantee the safety of agents during training.

A. Experimental Setup

1) Environment: We consider two different environments in Fig. 4. One is the Point Robot [23], 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 [47], 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 follows:

$$\hat{c} = \begin{cases} \mathbb{1}\{\mathsf{dist}(s, S_{\text{unsafe}}) = 0\}, & \alpha = 0 \\ \max\left\{0, 1 - \dfrac{1}{\alpha}\mathsf{dist}(s, S_{\text{unsafe}})\right\}, & \text{otherwise} \end{cases} \tag{18}$$

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



Point Robot Half-Cheetah

Fig. 4. Point Robot: the black point corresponds to agent, the green circle indicates the desired path, and red lines denote the constraint at the 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.

2) Shielding Rules: To ensure the safety of the exploration process, we construct the shielding rule $\mathcal{G} = (\overline{Q}, \zeta, \overline{A}, \eta, \mathcal{I})$. For Point Robot, the backup policy ζ applies a deceleration force to slow down the velocity of the agent to 0. 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 follows:

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

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

For Half-Cheetah, the MPC algorithm [8] 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 follows:

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

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

3) Baselines: To demonstrate the performance of the proposed SPPO, several baselines are evaluated for comparison. The first classical algorithm is constrained policy optimization (CPO) [23], which is a CMDP-based method that enforces safety constraints by solving restricted optimization problems in each iteration. Another CMDP-based algorithm is PDO [25], 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 conservative safety critics (CSC) [28] is also considered as a baseline, in which well-learned conservative critics are applied to filter unsafe behaviors. Furthermore, as an important baseline, Safe Advantage-based Intervention for Learning policies with Reinforcement (SAILR) [46], 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 eight (Point Robot) or five (Half-Cheetah) random seeds.

B. Main Experimental Results

In this section, experimental results demonstrate that SPPO shows improved learning efficiency and well-learned policy

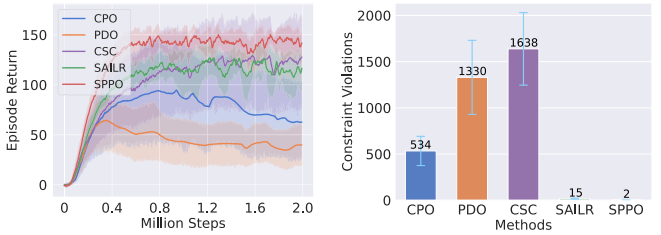


Fig. 5. Experimental results for Point Robot. Episode return learning curves that show the mean and standard deviation of the episode return throughout training (left). Histogram of the cumulative constraint violations upon training completion (right).

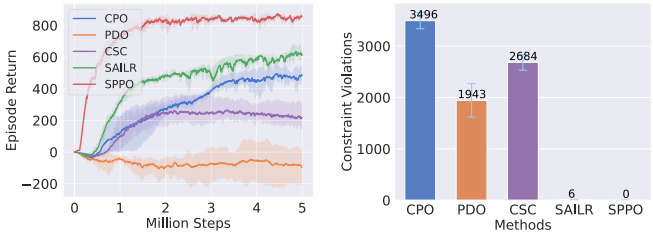


Fig. 6. Experimental results for Half-Cheetah. Episode return learning curves that show the mean and standard deviation of the episode return throughout training (left). Histogram of the cumulative constraint violations upon training completion (right).

performance compared with all baselines, and performs well in terms of safety guarantee during training.

1) Policy Performance and Learning Efficiency: SPPO and four baseline methods are employed to perform the two constrained tasks, as shown in Fig. 4. The episode returns over the course of training for SPPO and baselines are shown in Figs. 5 and 6 (left), from which it can be observed that the following holds.

1) The shielding-based safe RL algorithms (SAILR and SPPO) perform better policy performance and higher learning efficiency 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 and 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 IV.
2) Compared with 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 with the stochastic exploration of SAILR, thus improving data efficiency. In addition, as already demonstrated in Section III-D, 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.

2) Safety Performance During Training: In this section, 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.

TABLE I
ORDERS OF MAGNITUDE OF CUMULATIVE CONSTRAINT VIOLATIONS AND SAFETY RATES FOR POINT ROBOT

Method	Violations (10^*)	Safety Rate (%)
CPO [23]	$10^2 \uparrow$	88.99 ± 5.05
PDO [25]	$10^3 \uparrow$	74.53 ± 7.53
CSC [28]	$10^3 \uparrow$	69.84 ± 7.28
SAILR [46]	$10^2 \downarrow$	97.56 ± 0.95
SPPO (Ours)	$\mathbf{10} \downarrow$	$\mathbf{99.73 \pm 0.31}$

TABLE II
ORDERS OF MAGNITUDE OF CUMULATIVE CONSTRAINT VIOLATIONS AND SAFETY RATES FOR HALF-CHEETAH

Method	Violations (10^*)	Safety rate (%)
CPO [23]	$10^3 \uparrow$	54.85 ± 1.31
PDO [25]	$10^3 \uparrow$	72.17 ± 4.05
CSC [28]	$10^3 \uparrow$	65.10 ± 1.59
SAILR [46]	$10 \downarrow$	99.94 ± 0.03
SPPO (Ours)	$\mathbf{0}$	$\mathbf{100.00 \pm 0.00}$

Figs. 5 and 6 (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 environments, 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 I and II. 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. In contrary, both SAILR and SPPOs 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%).

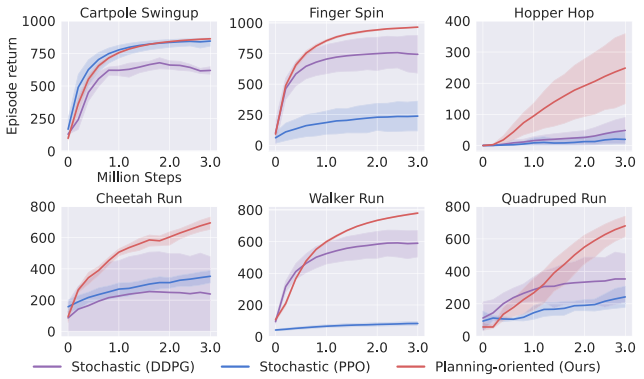


Fig. 7. Episode return curves for additional experiments on the role of planning-oriented exploration.

We conducted additional experiments to demonstrate how planning-oriented exploration contributes to the performance of SPPO. In particular, six RL tasks (including cartpole swingup, finger spin, hopper hop, cheetah run, walker run, and quadruped run) with different levels of complexity are selected for the evaluation in the MuJoCo-based robotics suite dm-control [47]. Our proposed planning-oriented RL (SPPO) is compared with the classical RL algorithms (Deep Deterministic Policy Gradient (DDPG) [48] and PPO [49]) via stochastic exploration. Fig. 7 shows the episode return curves during training, in which we can observe that the planning-oriented exploration can significantly improve learning efficiency and policy performance. The reason for this result is that the planning-oriented exploration can significantly reduce ineffective interactions between the agent and the environment compared to conventional stochastic exploration as discussed in Section IV-B.

C. 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 RL using path planning and shielding mechanisms 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.

VI. Conclusion

In this work, we develop an SPPO for data-efficient and safe exploration. In particular, SPPO is equipped with shielded planning for directed exploration and efficient data collection via MPPI, as well as an advantage-based shielding rule to keep the above processes safe. A TOPO method is then used for both policy improvement and the observation-independent latent dynamics enhancement. It is worth mentioning that we provide 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 toward more complex RL tasks, such as safe manipulation skill learning.

References

[1] J. Co-Reyes, Y. Liu, A. Gupta, B. Eysenbach, P. Abbeel, and S. Levine, "Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1009–1018.

[2] B. Eysenbach, R. R. Salakhutdinov, and S. Levine, "Search on the replay buffer: Bridging planning and reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 15246–15257.

[3] L. Dong, Y. Li, X. Zhou, Y. Wen, and K. Guan, "Intelligent trainer for dyna-style model-based deep reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 6, pp. 2758–2771, Jun. 2021.

[4] H. Wang, H. Zhang, L. Li, Z. Kan, and Y. Song, "Task-driven reinforcement learning with action primitives for long-horizon manipulation skills," *IEEE Trans. Cybern.*, early access, 2023, doi: 10.1109/TCYB.2023.3298195.

[5] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 1437–1480, 2015.

[6] Y. Zhang et al., "Barrier Lyapunov function-based safe reinforcement learning for autonomous vehicles with optimized backstepping," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, 2022, doi: 10.1109/TNNLS.2022.3186528.

[7] A. Modares, N. Sadati, B. Esmaeili, F. A. Yaghmaie, and H. Modares, "Safe reinforcement learning via a model-free safety certifier," *IEEE Trans. Neural Netw. Learn. Syst.*, 2023, doi: 10.1109/TNNLS.2023.3264815.

[8] G. Williams et al., "Information theoretic MPC for model-based reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 1714–1721.

[9] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 12498–12509.

[10] D. Hafner et al., "Learning latent dynamics for planning from pixels," in *Proc. Int. Conf. Mach. Learn.*, May 2019, pp. 2555–2565.

[11] H. Lin, Y. Sun, J. Zhang, and Y. Yu, "Model-based reinforcement learning with multi-step plan value estimation," 2022, *arXiv:2209.05530*.

[12] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. Johnson, and S. Levine, "SOLAR: Deep structured representations for model-based reinforcement learning," in *Proc. Int. Conf. Machin. Learn.*, 2019, pp. 7444–7453.

[13] J. Schrittwieser et al., "Mastering Atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, Dec. 2020.

[14] T. D. Nguyen, R. Shu, T. Pham, H. Bui, and S. Ermon, "Temporal predictive coding for model-based planning in latent space," in *Proc. Int. Conf. Machin. Learn.*, 2021, pp. 8130–8139.

[15] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak, "Planning to explore via self-supervised world models," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 8583–8592.

[16] N. A. Hansen, H. Su, and X. Wang, "Temporal difference learning for model predictive control," in *Proc. Int. Conf. Machin. Learn.*, 2022, pp. 8387–8406.

[17] L. Kaiser et al., "Model-based reinforcement learning for Atari," 2019, *arXiv:1903.00374*.

[18] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 4906–4913.

[19] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Proc. Adv. Neural Inf. Proces. Syst.*, vol. 31, 2018, pp. 4759–4770.

[20] W. Ye, S. Liu, T. Kurutach, P. Abbeel, and Y. Gao, "Mastering Atari games with limited data," in *Proc. Adv. Neural Inf. Proces. Syst.*, vol. 34, 2021, pp. 25476–25488.
[21] E. Altman, *Constrained Markov Decision Processes*, vol. 7. Boca Raton, FL, USA: CRC Press, 1999.
[22] V. S. Borkar, "An actor-critic algorithm for constrained Markov decision processes," *Syst. Control Lett.*, vol. 54, no. 3, pp. 207–213, Mar. 2005.
[23] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 22–31.
[24] H. Le, C. Voloshin, and Y. Yue, "Batch policy learning under constraints," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3703–3712.
[25] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, "Risk-constrained reinforcement learning with percentile risk criteria," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6070–6120, 2017.
[26] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," 2018, *arXiv:1805.11074*.
[27] B. Peng et al., "Model-based chance-constrained reinforcement learning via separated proportional-integral Lagrangian," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 1, pp. 466–478, Jan. 2024.
[28] H. Bharadhwaj, A. Kumar, N. Rhinehart, S. Levine, F. Shkurti, and A. Garg, "Conservative safety critics for exploration," in *Proc. Int. Conf. Learn. Represent.*, 2021.
[29] A. K. Jayant and S. Bhatnagar, "Model-based safe deep reinforcement learning via a constrained proximal policy optimization algorithm," in *Proc. Adv. Neural Inf. Proces. Syst.*, vol. 35, 2022, pp. 24432–24445.
[30] Z. Liu et al., "Constrained variational policy optimization for safe reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 13644–13668.
[31] T. Lin, C. Jin, and M. Jordan, "On gradient descent ascent for nonconvex-concave minimax problems," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 6083–6093.
[32] T. Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge, "Accelerating safe reinforcement learning with constraint-mismatched baseline policies," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 11795–11807.
[33] L. Yang et al., "Constrained update projection approach to safe policy optimization," in *Proc. Adv. Neural Inf. Proces. Syst.*, vol. 35, 2022, pp. 9111–9124.
[34] S. Carr, N. Jansen, S. Junges, and U. Topcu, "Safe reinforcement learning via shielding under partial observability," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2023, vol. 37, no. 12, pp. 14748–14756.
[35] S. Li and O. Bastani, "Robust model predictive shielding for safe reinforcement learning with stochastic dynamics," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 7166–7172.
[36] B. Thananjeyan et al., "Recovery RL: Safe reinforcement learning with learned recovery zones," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4915–4922, Jul. 2021.
[37] L. Zhang, Q. Zhang, L. Shen, B. Yuan, X. Wang, and D. Tao, "Evaluating model-free reinforcement learning toward safety-critical tasks," in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, no. 12, pp. 15313–15321.
[38] Z. Zhou, O. S. Oguz, M. Leibold, and M. Buss, "Learning a low-dimensional representation of a safe region for safe reinforcement learning on dynamical systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 5, pp. 2513–2527, May 2023.
[39] K. P. Wabersich and M. N. Zeilinger, "Predictive control barrier functions: Enhanced safety mechanisms for learning-based control," *IEEE Trans. Autom. Control*, vol. 68, no. 5, pp. 2638–2651, May 2023.
[40] C. Dawson, S. Gao, and C. Fan, "Safe control with learned certificates: A survey of neural Lyapunov, barrier, and contraction methods for robotics and control," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1749–1767, Jun. 2023.
[41] M. H. Cohen and C. Belta, "Safe exploration in model-based reinforcement learning using control barrier functions," *Automatica*, vol. 147, Jan. 2023, Art. no. 110684.
[42] D. Yu, H. Ma, S. Li, and J. Chen, "Reachability constrained reinforcement learning," in *Proc. Int. Conf. Machin. Learn.*, 2022, pp. 25636–25655.
[43] M. Selim, A. Alanwar, S. Kousik, G. Gao, M. Pavone, and K. H. Johansson, "Safe reinforcement learning using black-box reachability analysis," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 10665–10672, Oct. 2022.
[44] K.-C. Hsu, A. Z. Ren, D. P. Nguyen, A. Majumdar, and J. F. Fisac, "Sim-to-lab-to-real: Safe reinforcement learning with shielding and generalization guarantees," *Artif. Intell.*, vol. 314, Jan. 2023, Art. no. 103811.
[45] G. Williams, A. Aldrich, and E. Theodorou, "Model predictive path integral control using covariance variable importance sampling," 2015, *arXiv:1509.01149*.
[46] N. C. Wagener, B. Boots, and C.-A. Cheng, "Safe reinforcement learning using advantage-based intervention," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 10630–10640.
[47] S. Tunyasuvunakool et al., "Dm_control: Software and tasks for continuous control," *Softw. Impacts*, vol. 6, Nov. 2020, Art. no. 100022.
[48] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[49] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

Hao Wang received the B.S. degree in mechanical engineering from the China University of Mining and Technology, Xuzhou, Jiangsu, China, in 2017. He is currently pursuing the Ph.D. degree in automation with the University of Science and Technology of China, Hefei, Anhui, China.

His current research interests include motion planning in robotics, manipulation skill learning, and deep reinforcement learning.

Jiahu Qin (Senior Member, IEEE) received the first Ph.D. degree in control science and engineering from the Harbin Institute of Technology, Harbin, China, in 2012, and the second Ph.D. degree in systems and control from The Australian National University, Canberra, ACT, Australia, in 2014.

He is currently a Professor with the Department of Automation, University of Science and Technology of China, Hefei, China. His current research interests include networked control systems, autonomous intelligent systems, and human–robot interaction.

Zhen Kan (Senior Member, IEEE) received the Ph.D. degree from the Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL, USA, in 2011.

He was a Post-Doctoral Research Fellow with the Air Force Research Laboratory, Eglin AFB, FL, USA; the University of Florida REEF, Shalimar, FL, USA, from 2012 to 2016; and an Assistant Professor with the Department of Mechanical Engineering, University of Iowa, Iowa City, IA, USA, from 2016 to 2019. He is currently a Professor with the Department of Automation, University of Science and Technology of China, Hefei, China. His research interests include networked control systems, nonlinear control, formal methods, and robotics.

Dr. Kan currently serves on program committees for several internationally recognized scientific and engineering conferences and is an Associate Editor of IEEE TRANSACTIONS ON AUTOMATIC CONTROL.