Solving Minimum-Cost Reach Avoid using Reinforcement Learning

Oswin So*
Department of Aeronautics and Astronautics
MIT
oswinso@mit.edu

Cheng Ge*
Department of Aeronautics and Astronautics
MIT
gec_mike@mit.edu

Chuchu Fan
Department of Aeronautics and Astronautics
MIT
chuchu@mit.edu

Abstract

Current reinforcement-learning methods are unable to *directly* learn policies that solve the minimum cost reach-avoid problem to minimize cumulative costs subject to the constraints of reaching the goal and avoiding unsafe states, as the structure of this new optimization problem is incompatible with current methods. Instead, a *surrogate* problem is solved where all objectives are combined with a weighted sum. However, this surrogate objective results in suboptimal policies that do not directly minimize the cumulative cost. In this work, we propose **RC-PPO**, a reinforcement-learning-based method for solving the minimum-cost reach-avoid problem by using connections to Hamilton-Jacobi reachability. Empirical results demonstrate that RC-PPO learns policies with comparable goal-reaching rates to while achieving up to 57% lower cumulative costs compared to existing methods on a suite of minimum-cost reach-avoid benchmarks on the Mujoco simulator.

1 Introduction

Many real-world tasks can be framed as a constrained optimization problem where reaching a goal at the terminal state and ensuring safety (i.e., reach-avoid) is desired while minimizing some cumulative cost as an objective function, which we term the *minimum-cost* reach-avoid problem.

The cumulative cost, which differentiates this from the traditional reach-avoid problem, can be used to model desirable aspects of a task such as minimizing energy consumption, maximizing smoothness, or any other pseudo-energy function, and allows for choosing the most desirable policy among the many policies that can satisfy the reach-avoid requirements. For example, energy-efficient autonomous driving [52, 74] can be seen as a task where the vehicle must reach a destination, follow traffic rules, and minimize fuel consumption. Minimizing fuel use is also a major concern for low-thrust or energy-limited systems such as spacecraft [53] and quadrotors [36]. For example, quadrotors often have to choose limited battery life to meet the payload capacity. Hence, minimizing their energy consumption, which can be done by taking advantage of wind patterns, is crucial for keeping them airborne to complete more tasks.

If only a single control trajectory is desired, this class of problems can be solved using numerical trajectory optimization by either optimizing the timestep between knot points [55] or a bilevel

*These authors contributed equally to this work optimization approach that adjusts the number of knot points in an outer loop [50, 32, 59]. However, in this setting, the dynamics are assumed to be known, and only a single trajectory is obtained. Therefore, the computation will need to be repeated when started from a different initial state. The computational complexity of trajectory optimization prevents it from being used in real time. Moreover, the use of nonlinear numerical optimization may result in poor solutions that lie in suboptimal local minima [19].

Alternatively, to obtain a control policy, reinforcement learning (RL) can be used. However, existing methods are unable to directly solve the minimum-cost reach-avoid problem. Although RL has been used to solve many tasks where reaching a goal is desired, goal-reaching is encouraged as a *reward* instead of as a *constraint* via the use of either a sparse reward at the goal [4, 51, 67], or a surrogate dense reward [67, 37][1]. However, posing the reach constraint as a reward then makes it difficult to optimize for the cumulative cost at the same time. In many cases, this is done via a weighted sum of the two terms [11, 40, 28]. However, the optimal policy of this new *surrogate* objective may not necessarily be the optimal policy of the original problem. Another method of handling this is to treat the cumulative cost as a constraint and solve for a policy that maximizes the reward while keeping the cumulative cost under some fixed threshold, resulting in a new constrained optimization problem that can be solved as a constrained Markov decision process (CMDP) [3]. However, the choice of this fixed threshold becomes key: too small and the problem is not feasible, destabilizing the training process. Too large, and the resulting policy will simply ignore the cumulative cost.

To tackle this issue, we propose **Reach Constrained Proximal Policy Optimization(RC-PPO)**, a new algorithm that targets the minimum-cost reach-avoid problem. We first convert the reach-avoid problem to a reach problem on an augmented system and use the corresponding *reach* value function to compute the optimal policy. Next, we use a novel two-step PPO-based RL-based framework to learn this value function and the corresponding optimal policy. The first step uses a PPO-inspired algorithm to solve for the optimal value function and policy, conditioned on the cost upper bound. The second step fine-tunes the value function and solves for the least upper bound on the cumulative cost to obtain the final optimal policy. Our main contributions are summarized below:

- We prove that the minimum-cost reach-avoid problem can be solved by defining a set of augmented dynamics and a simplified constrained optimization problem.
- We propose RC-PPO, a novel algorithm based on PPO that targets the minimum-cost reach-avoid problem, and prove that our algorithm converges to a locally optimal policy.
- Simulation experiments show that RC-PPO achieves reach rates comparable with the baseline method with the highest reach rate while achieving significantly lower cumulative costs.

2 Related Works

Terminal-horizon state-constrained optimization Terminal state constraints are quite common in the dynamic optimization literature. For the finite-horizon case, for example, one method of guaranteeing the stability of model predictive control (MPC) is with the use of a terminal state constraint [21]. Since MPC is implemented as a discrete-time finite-horizon numerical optimization problem, the terminal state constraints can be easily implemented in an optimization program as a normal state constraint. The case of a flexible-horizon constrained optimization is not as common but can still be found. For example, one method of time-optimal control is to treat the integration timestep as a control variable while imposing state constraints on the initial and final knot points [55]. Another method is to consider a bilevel optimization problem, where the number of knot points is optimized for in the outer loop [50, 32, 59].

Goal-conditioned Reinforcement Learning There have been many works on goal-conditioned reinforcement learning. These works mainly focus on the challenges of tackling sparse rewards [4, 67, 70, 37] or even learning without rewards completely, either via representation learning objectives [44, 27, 45, 68, 61, 46, 13, 48, 42, 20] or by using contrastive learning to learn reward functions [23, 16, 71, 25, 12, 35, 34, 72, 75, 47], often in imitation learning settings [31, 24]. However, the *manner* in which these goals are reached is not considered, and it is difficult to extend these works to additionally minimize some cumulative cost.

[1]If the dense reward is not specified correctly, however, it can lead to unwanted local minima [67] that optimize the reward function in an undesirable manner, i.e., *reward hacking* [17, 2]

Constrained Reinforcement Learning One way of using existing techniques to approximately tackle the minimum-cost reach-avoid problem is to flip the role of the cumulative-cost objective and the goal-reaching constraint by treating the goal-reaching constraint as an objective via a (sparse or dense) reward and the cumulative-cost objective as a constraint with a cost threshold, turning the problem into a CMDP [3]. In recent year, there has been significant interest in deep RL methods for solving CMDPs [1, 64, 60]. While these methods are effective at solving the transformed CMDP problem, the optimal policy to the CMDP may not be the optimal policy to the original minimum-cost reach-constrained problem, depending on the choice of the cost constraint.

Reachability Analysis Reachability analysis looks for solutions to the reach-avoid problem. That is, to solve for the set of initial conditions and an appropriate control policy to drive a system to a desired goal set while avoiding undesireable states. Hamilton-Jacobi (HJ) reachability analysis [66, 38, 43, 39, 5] provides a methodology for the case of dynamics in *continuous-time* via the solution of a partial differential equation (PDE) and is conventionally solved via numerical PDE techniques that use state-space discretization [43]. This has been extended recently to the case of discrete-time dynamics and solved using off-policy [22, 33] and on-policy [58, 26] reinforcement learning. While reachability analysis concerns itself with the reach-avoid problem, we are instead interested in solutions to the *minimum-cost* reach-avoid problem.

3 Problem Formulation

In this paper, we consider a class of *minimum-cost* reach-avoid problems defined by the tuple $\mathcal{M} \coloneqq \langle \mathcal{X}, \mathcal{U}, f, c, g, h \rangle$. For the state space $\mathcal{X} \subseteq \mathbb{R}^n$ and action space $\mathcal{U} \subseteq \mathbb{R}^m$, the control objective for the system states $x_t \in \mathcal{X}$ evolving under the *deterministic* discrete dynamics $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ as $x_{t+1} = f(x_t, u_t)$ is to reach the goal region $\mathcal{G}$ and avoid the unsafe set $\mathcal{F}$ while minimizing the cumulative cost $\sum_{t=0}^{T-1} c(x_t, \pi(x_t))$ under control input $u_t = \pi(x_t)$ for a designed control policy $\pi : \mathcal{X} \to \mathcal{U}$, where T denotes the first timestep that the agent reaches the goal $\mathcal{G}$. The sets $\mathcal{G}$ and $\mathcal{F}$ are given as the 0-sublevel and strict 0-superlevel sets $g : \mathcal{X} \to \mathbb{R}$ and $h : \mathcal{X} \to \mathbb{R}$ respectively, i.e., $\mathcal{G} \coloneqq \{x \in \mathcal{X} \mid g(x) \leq 0\}$ and $\mathcal{F} \coloneqq \{x \in \mathcal{X} \mid h(x) > 0\}$. This can be formulated formally as finding a policy π that solves the following constrained *flexible* final-time optimization problem for a given initial state x_0:

$$\begin{aligned} \min_{\pi, T} \quad & \sum_{t=0}^{T-1} c\big(x_t, \pi(x_t)\big) && \text{(1a)} \\ \text{s.t.} \quad & x_T \in \mathcal{G}, && \text{(1b)} \\ & x_t \notin \mathcal{F} \quad \forall t \in \{0, \dots, T\}, && \text{(1c)} \\ & x_{t+1} = f\big(x_t, \pi(x_t)\big). && \text{(1d)} \end{aligned}$$

Note that as opposed to either traditional finite-horizon constrained optimization problems where T is fixed or infinite-horizon problems where $T = \infty$, the time horizon T is also a decision variable. Moreover, the goal constraint (1b) is only enforced at the terminal timestep T. These two differences prevent the straightforward application of existing RL methods to solve (1).

3.1 Reachability Analysis for Reach-Avoid Problems

In discrete time, the set of initial states that can reach the goal $\mathcal{G}$ without entering the avoid set $\mathcal{F}$ can be represented by the 0-sublevel set of a reach-avoid value function $V^\pi_{g,h}$ [33]. Given functions g, h describing $\mathcal{G}$ and $\mathcal{F}$ and a policy π, the reach-avoid value function $V^\pi_{g,h} : \mathcal{X} \to \mathbb{R}$ is defined as

$$V^\pi_{g,h}(x_0) = \min_{T \in \mathbb{N}} \max \Big\{ g(x^\pi_T),\ \max_{t \in \{0,\dots,T\}} h(x^\pi_t) \Big\}, \tag{2}$$

where x^π_t denote the system state at time t under a policy π starting from an initial state $x^\pi_0 = x_0$. In the rest of the paper, we suppress the argument x_0 for brevity whenever clear from the context. It can be shown that the reach-avoid value function satisfies the following recursive relationship via the reach-avoid Bellman equation (RABE) [33]

$$V^\pi_{g,h}(x^\pi_t) = \max \Big\{ h(x^\pi_t),\ \min\{ g(x^\pi_t),\ V^\pi_{g,h}(x^\pi_{t+1}) \} \Big\} \quad \forall t \geq 0. \tag{3}$$

The Bellman equation (3) can then be used in a reinforcement learning framework (e.g., via a modification of soft actor-critic[30, 29]) as done in [33] to solve the reach-avoid problem.

Note that existing methods of solving reach-avoid problems through this formulation focus on minimizing the value function $V^\pi_{g,h}$. This is not necessary as any policy that results in $V^\pi_{g,h} \leq 0$ solves the reach-avoid problem, albeit without any cost considerations. However, it is often the case that we wish to minimize a cumulative cost (e.g., (1a)) on top of the reach-avoid constraints (1b)-(1c) for a *minimum-cost* reach-avoid problem. To address this class of problems, we next present a modification to the reach-avoid framework that additionally enables the minimization of the cumulative cost.

3.2 Reachability Analysis for Minimum-cost Reach-Avoid Problems

We now provide a new framework to solve the minimum-cost reach-avoid by lifting the original system to a higher dimensional space and designing a set of augmented dynamics that allow us to convert the original problem into a reachability problem on the augmented system.

Let $\mathbb{I}$ denote the shifted indicator function defined as $\mathbb{I}_{b\in B} := \begin{cases} +1 & b \in B, \\ -1 & b \notin B. \end{cases}$ Define the *augmented* state as $\hat{x} = (x, y, z) \subseteq \hat{\mathcal{X}} := \mathcal{X} \times \{-1, 1\} \times \mathbb{R}$. We now define a corresponding augmented dynamics function $f' : \hat{\mathcal{X}} \times \mathcal{U} \to \hat{\mathcal{X}}$ as

$$\hat{f}\big(x_t, y_t, z_t, u_t\big) = \big(f(x_t),\ \max\{\mathbb{I}_{f(x_t)\in\mathcal{F}},\ y_t\},\ z_t - c(x_t,\ u_t)\big), \tag{4}$$

where $y_0 = \mathbb{I}_{x_0\in\mathcal{F}}$. Note that $y_t = 1$ if the state has entered the avoid set $\mathcal{F}$ at some timestep from 0 to t and is *unsafe*, and $y_t = 0$ if the state has not entered the avoid set $\mathcal{F}$ at any timestep from 0 to t and is *safe*. Moreover, z_t is equal to z_0 minus the cost-to-come, i.e., for state trajectory $x_{0:t}$ and action trajectory $u_{0:t}$, i.e., $z_{t+1} = z_0 - \sum_{k=0}^{t} c(x_t, u_t)$. Under the augmented dynamics, we now define the following augmented goal function $\hat{g} : \hat{\mathcal{X}} \to \mathbb{R}$ as

$$\hat{g}(x, y, z) := \max\{g(x),\ Cy,\ -z\}, \tag{5}$$

where $C > 0$ is an arbitrary constant.[2] With this definition of $\hat{g}$, an augmented goal region $\hat{\mathcal{G}}$ can be defined as $\hat{\mathcal{G}} := \{\hat{x} \mid \hat{g}(\hat{x}) \leq 0\} = \{(x, y, z) \mid x \in \mathcal{G},\ y = -1,\ z \geq 0\}$. In other words, starting from initial condition $\hat{x}_0 = (x_0, y_0, z_0)$, reaching the goal on the augmented system $\hat{x}_T \in \hat{g}$ at timestep T implies that 1) the goal is reached at x_T for the original system, 2) the state trajectory remains safe and does not enter the avoid set $\mathcal{F}$, and 3) z_0 is an upper-bound on the total cost-to-come: $\sum_{t=0}^{T-1} c(x_t, u_t) \leq z_0$. We call this the upper-bound property. The above intuition on the newly defined augmented system is formalized in the following theorem, whose proof is provided in Appendix C.1.

Theorem 1. For given initial conditions $x_0 \in \mathcal{X}$, $z_0 \in \mathbb{R}$ and control policy π, consider the trajectory for the original system $\{x_0, \ldots x_T\}$ and its corresponding trajectory for the augmented system $\{(x_0, y_0, z_0), \ldots (x_T, y_T, z_T)\}$ for some $T > 0$. Then, the reach constraint $x_T \in \mathcal{G}$ (1b), avoid constraint $x_t \notin \mathcal{F}$ $\forall t \in \{0, 1, \ldots, T\}$ (1c) and the upper-bound property $z_0 \geq \sum_{k=0}^{T-1} c\big(x_k, \pi(x_k)\big)$ hold if and only if the augmented state reaches the augmented goal at time T, i.e., $(x_T, y_T, z_T) \in \hat{\mathcal{G}}$.

With this construction, we have folded the avoid constraints $x_t \notin \mathcal{F}$ (1c) into the reach specification on the augmented system. In other words, solving the reach problem on the augmented system results in a reach-avoid solution of the original system. As a result, we can simplify the value function (2) and Bellman equation (3), resulting in the following definition of the reach value function $\tilde{V}_{\hat{g}} : \hat{\mathcal{X}} \to \mathbb{R}$

$$\tilde{V}^\pi_{\hat{g}}(\hat{x}_0) = \min_{t\in\mathbb{N}} \hat{g}(\hat{x}^\pi_t). \tag{6}$$

Similar to (2), the 0-sublevel set of $\tilde{V}_{\hat{g}}$ describes the set of augmented states $\hat{x}$ that can reach the augmented goal $\hat{\mathcal{G}}$. We can also similarly obtain a recursive definition of the reach value function $\tilde{V}_{\hat{g}}$ given by the reachability Bellman equation (RBE)

$$\tilde{V}^\pi_{\hat{g}}(x^\pi_t, y^\pi_t, z^\pi_t) = \min\big\{\hat{g}(x^\pi_t,\ y^\pi_t, z^\pi_t), \tilde{V}^\pi_{\hat{g}}(x^\pi_{t+1}, y^\pi_{t+1}, z^\pi_{t+1})\big\} \quad \forall t \geq 0, \tag{7}$$

[2]In practice, we use $C = \max_{x\in\mathcal{X}} g(x)$.

whose proof we provide in Appendix C.2.

We now solve the minimum-cost reach-avoid problem using this augmented system. By Theorem 1, the z_0 is an upper bound on the cumulative cost to reach the goal while avoiding the unsafe set if and only if the augmented state $\hat{x}$ reaches the augmented goal. Since this upper bound is tight, the least upper bound z_0 that still reaches the augmented goal thus corresponds to the minimum-cost policy that satisfies the reach-avoid constraints. In other words, the minimum-cost reach-avoid problem for a given initial state x_0 can be reformulated as the following optimization problem.

$$\min_{\pi, z_0} \quad z_0 \tag{8a}$$

$$\text{s.t.} \quad \tilde{V}_{\hat{g}}^{\pi}(x_0, \mathbb{I}_{x_0 \in \mathcal{F}}, z_0) \leq 0. \tag{8b}$$

Remark 1 (Connections to the epigraph form in constrained optimization). The resulting optimization problem (8) can be interpreted as an epigraph reformulation [9] of the minimum-cost reach-avoid problem (1). The epigraph reformulation results in a problem with *linear* objective but yields the same solution as the original problem [9]. The construction we propose in this work can be seen as a *dynamic* version of this epigraph reformulation technique originally developed for static problems and is similar to recent results that also make use of the epigraph form for solving infinite-horizon constrained optimization problems [58].

4 Solving with Reinforcement Learning

In the previous section, we reformulated the minimum-cost reach-avoid problem by constructing an augmented system and used its reach value function (6) in a new constrained optimization problem (8) over the cost upper-bound z_0. In this section, we propose an RL-based method for solving (8) via a two-phase method.

In the first step, we learn the optimal $\tilde{V}_g^{\pi}$ over all possible policies π using reinforcement learning. We consider the common policy gradient framework [63]. In this framework, we extend the class of control policies considered to stochastic ones to encourage exploration. To this end, we re-define the reach value function $V_{\hat{g}}^{\pi}$ using a similar Bellman equation under a stochastic policy as follows.

Definition 1. Stochastic Reachability Bellman equation (SRBE) Given function $\hat{g}$ in (5), a stochastic policy π, and initial conditions $x_0 \in \mathcal{X}$, $z_0 \in \mathbb{R}$, the stochastic reach value function $\tilde{V}_{\hat{g}}^{\pi}$ is defined as the solution to the following stochastic Bellman equation:

$$\tilde{V}_{\hat{g}}^{\pi}(\hat{x}_t) = \mathbb{E}_{\tau \sim \pi}[\min\{\hat{g}(\hat{x}_t), \tilde{V}_{\hat{g}}^{\pi}(\hat{x}_{t+1})\}] \quad \forall t \geq 0, \tag{9}$$

where $\hat{x}_0 = (x_0, y_0, z_0)$ with $y_0 = \mathbb{I}_{x_0 \in \mathcal{F}}$.

For this stochastic Bellman equation, the Q function [62] is defined as

$$\tilde{Q}_{\hat{g}}^{\pi}(\hat{x}_t, u_t) = \min\{\hat{g}(\hat{x}_t), \tilde{V}_g(\hat{x}_{t+1})\}. \tag{10}$$

We can now derive the corresponding policy gradient theorem for $\tilde{V}_{\hat{g}}^{\pi}$ with stationary distribution on the following Markov Decision Process with absorbing state under deterministic transition function and stochastic policy π.

Definition 2. (Reachability Markov Decision Process) The Reachability Markov Decision Process is defined on the augmented dynamic in Equation (5) with an added absorbing state s_0. We define the transition function f_r' with the absorbing state as

$$f_r'(\hat{x}, u) = \begin{cases} \hat{f}(\hat{x}, u), & \text{if } \tilde{V}_{\hat{g}}^{\pi}(\hat{x}) > \hat{g}(\hat{f}(\hat{x}, u)) \\ s_0, & \text{if } \tilde{V}_{\hat{g}}^{\pi}(\hat{x}) \leq \hat{g}(\hat{f}(\hat{x}, u)). \end{cases} \tag{11}$$

We denote the stationary distribution starting at $\hat{x} \in \mathcal{X} \times \{-1, 1\} \times \mathbb{R}$ under stochastic policy π as $d_\pi'(\hat{x})$.

We now derive a policy gradient theorem for the Reachability MDP in Definition 2 which yields an almost identical expression for the policy gradient.

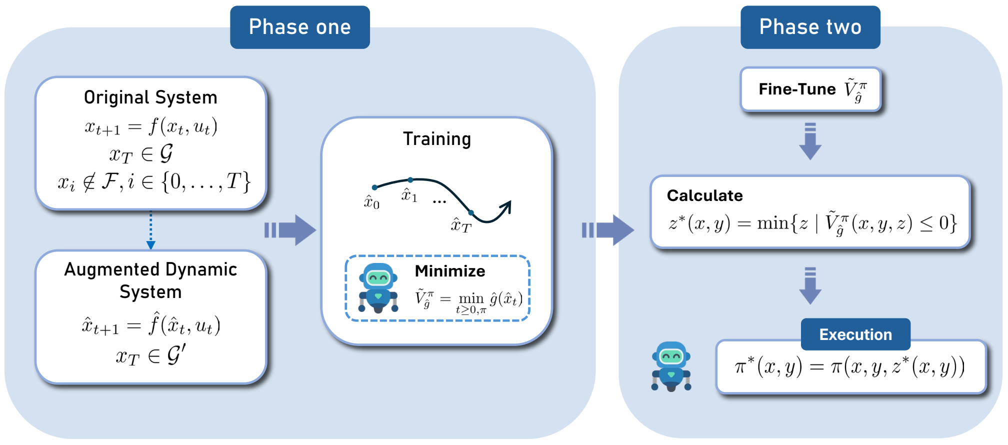

Phase one
Original System
$x_{t+1} = f(x_t, u_t)$
$x_T \in \mathcal{G}$
$x_i \notin \mathcal{F}, i \in \{0, \dots, T\}$
Augmented Dynamic System
$\hat{x}_{t+1} = \hat{f}(\hat{x}_t, u_t)$
$x_T \in \mathcal{G}'$
Training
$\hat{x}_0$
$\hat{x}_1$
$\hat{x}_T$
Minimize
$\tilde{V}_{\hat{g}}^{\pi} = \min_{t \geq 0, \pi} \hat{g}(\hat{x}_t)$
Phase two
Fine-Tune $\tilde{V}_{\hat{g}}^{\pi}$
Calculate
$z^*(x, y) = \min\{z \mid \tilde{V}_{\hat{g}}^{\pi}(x, y, z) \leq 0\}$
Execution
$\pi^*(x, y) = \pi(x, y, z^*(x, y))$

Figure 1: Summary of the **RC-PPO** algorithm. In phase one, the original dynamic system is transformed into the augmented dynamic system defined in (4). Then RL is used to optimize value function $\tilde{V}_{\hat{g}}^{\pi}$ and learn a stochastic policy π. In phase two, we fine-tune $\tilde{V}_{\hat{g}}^{\pi}$ on a deterministic version of π and compute the optimal upper-bound z^* to obtain the optimal deterministic policy π^*.

Theorem 2. (Policy Gradient Theorem) For policy π_θ parameterized by θ, the gradient of the policy value function $\tilde{V}_{\hat{g}}^{\pi_\theta}$ satisfies

$$\nabla_\theta \tilde{V}_{\hat{g}}^{\pi_\theta}(\hat{x}) \propto \mathbb{E}_{\hat{x}' \sim d'_\pi(\hat{x}), u \sim \pi_\theta}\left[\tilde{Q}^{\pi_\theta}(\hat{x}', u)\, \nabla_\theta \ln \pi_\theta(u \mid \hat{x}')\right], \tag{12}$$

under the stationary distribution $d'_\pi(\hat{x})$ for Reachability MDP in Definition 2

The proof of this new policy gradient theorem Theorem 2 follows the proof of the normal policy gradient theorem [62], differing only in the expression of the stationary distribution, which we provide in Appendix C.3.

In practice, we cannot get the 'true' $\tilde{V}_{\hat{g}}^{\pi_\theta}$ for given π_θ. Thus, the stationary distribution $d'_\pi(\hat{x})$ in Definition 2 is hard to simulate during the learning process. Instead, we consider the stationary distribution under the original augmented dynamic system. Also, note that Definition 1 does not induce a contraction map. This harms the performance of the outcome policy. We apply the same trick as [33] by introducing an additional discount factor γ into the Bellman equation with (7) as

$$\tilde{V}_{\hat{g}}^{\pi}(\hat{x}_t) = (1-\gamma)\hat{g}(\hat{x}_t) + \gamma \mathbb{E}_{\hat{x}_{t+1} \sim \tau}[\min\{\hat{g}(\hat{x}_t), \tilde{V}_{\hat{g}}^{\pi}(\hat{x}_{t+1})\}]. \tag{13}$$

This provides us with a contraction map (proved in [33]) and we leave the discussion of choosing γ in Appendix A.

Under this Bellman equation, we could derive the Q function as

$$\tilde{Q}_{\hat{g}}^{\pi}(\hat{x}_t, u_t) = (1-\gamma)\hat{g}(\hat{x}_t) + \gamma \min\{\hat{g}(\hat{x}_t), \tilde{V}_{\hat{g}}^{\pi}(\hat{x}_{t+1})\}. \tag{14}$$

Following proximal policy optimization (PPO) [57], we use generalized advantage estimation (GAE) [56] to compute a variance-reduced advantage function $\tilde{A}_{\hat{g}}^{\pi} = \tilde{Q}_{\hat{g}}^{\pi} - \tilde{V}_{\hat{g}}^{\pi}$ for the policy gradient theorem Theorem 2 using the λ-return [62]. We refer to Appendix B for the definition of $\hat{A}_{\hat{g}}^{\pi(\text{GAE})}$ and denote the loss function when $\theta = \theta_l$ as $\mathcal{J}_\pi(\theta) = \mathbb{E}_{\hat{x}, u \sim \pi_{\theta_l}}\left[\overline{A^{\pi_{\theta_l}}}(\hat{x}, u)\right]$, where

$$\begin{gathered}\overline{A^{\pi_{\theta_l}}}(\hat{x}, u) = \max\left(-\frac{\pi_\theta(u \mid \hat{x})}{\pi_{\theta_l}(u \mid \hat{x})}\hat{A}_{\hat{g}}^{\pi_{\theta_l}(GAE)}(\hat{x}, u), CLIP\left(\epsilon, -\hat{A}_{\hat{g}}^{\pi_{\theta_l}(GAE)}(\hat{x}, u)\right)\right), \\ CLIP(\epsilon, A) = \begin{cases}(1+\epsilon)A, & A \geq 0, \\ (1-\epsilon)A, & A < 0.\end{cases}\end{gathered} \tag{15}$$

An empirical comparison of the GAE estimator is also provided in Appendix B.

The phase one of our algorithm Reachability Constraint Proximal Policy Gradient (RC-PPO) is presented in Algorithm 1.

Algorithm 1 RC-PPO Algorithm (Phase One)

```
Require: Initial policy parameter $\theta_0$, value function parameter $\omega_0$
 1: for k = 0, ... do
 2:     Collect set of trajectories $\mathcal{D}$ with policy $\pi_{\theta_k}$ for $T$ timesteps
 3:     Compute advantage estimates $\overline{A^{\pi_{\theta_k}}}$
 4:     Fit value function by regression on mean-square error
 5:     Update the policy parameter $\theta$ by minimizing $\mathcal{J}_\pi(\theta)$
 6: end for
 7: return Parameter $\theta, \omega$
```

In phase two, because we derive our framework based on deterministic policy scenarios, to calculate optimal z for all $(x, y) \in \mathcal{X} \times \{-1, 1\}$, we define $\pi^*(\hat{x}) := \arg\max_{u \in \mathcal{U}} p(u \mid \hat{x})$. After fixing our policy as deterministic one, we fine-tune $V_{\hat{g}}^{\pi}$ based on π^* and obtain $\tilde{V}_{\hat{g}}^{*}$. In this way, Problem 8 becomes a 1D root-finding problem. This could be easily tackled through the bisection method. To further boost the bisection step, we apply regression on tuple $((x, y), z^*)$ where (x, y) is uniformly sampled in $\mathcal{X} \times \{-1, 1\}$ and z^* is minimal value that satisfies $V_{\hat{g}}^{*}(x, y, z^*) \leq 0$ got through bisection method. We conclude the second phase of **RC-PPO** as

Algorithm 2 RC-PPO Algorithm (Phase Two)

```
Require: Policy parameter $\theta$, value function parameter $\omega$ and estimate maximal energy budget $z_{max}$
 1: for k = 0, ... do
 2:     Collect set of trajectories $\mathcal{D}$ with deterministic policy $\pi^*$ for $T$ timesteps
 3:     Fine-tune value function by regression on mean-square error
 4: end for
 5: for k = 0, ... do
 6:     Sample $(x, y) \in \mathcal{X} \times \{-1, 1\}$
 7:     Calculate $z^*$ through bisection method over $[0, z_{max}]$
 8: end for
 9: Train network $\tilde{z}(x, y)$ on previous samples
10: return Optimal policy $\pi^*(x, y, \tilde{z}(x, y))$
```

We leave the theoretical analysis in Appendix A for the choice of γ. We refer to Appendix D for a convergence proof of an actor-critic version of our method without the GAE estimator.

5 Experiments

Baselines We consider two categories of RL baselines. The first is goal-conditioned reinforcement learning which focuses on goal-reaching but does not consider minimization of the cost. For this category, we consider the Contrastive Reinforcement Learning (CRL) [20] method. We also compare against safe RL methods that solve CMDPs. As the minimum-cost reach-avoid problem (1) cannot be posed as a CMDP, we reformulate (1) into the following *surrogate* CMDP

$$\min_{\pi} \quad \mathbb{E}_{x_t, u_t \sim d_\pi} \sum_t \big[-\gamma^t r(x_t, u_t) \big] \tag{16a}$$

$$\text{s.t.} \quad \mathbb{E}_{x_t, u_t \sim d_\pi} \sum_t \big[\gamma^t \mathbb{1}_{x_t \in \mathcal{F}} \times C_{\text{fail}} \big] \leq 0, \tag{16b}$$

$$\mathbb{E}_{x_t, u_t \sim d_\pi} \sum_t \big[\gamma^t c(x_t, u_t) \big] \leq \mathcal{X}_{\text{threshold}} \tag{16c}$$

where the reward r incentivies goal-reaching, C_{fail} is a term balancing two constraint terms, and $\mathcal{X}_{\text{threshold}}$ is a hyperparameter on the cumulative cost. For this category, we consider the CPPO [60] and RESPO [26]. Note that RESPO also incorporates reachability analysis to adapt the Lagrange multipliers for each constraint term. We implement the above CMDP-based baselines with three different choices of $\mathcal{X}_{\text{thresholds}}$: $\mathcal{X}_{\text{L}}$, $\mathcal{X}_{\text{M}}$ and $\mathcal{X}_{\text{H}}$. For RESPO, we found $\mathcal{X}_{\text{M}}$ to outperform both $\mathcal{X}_{\text{L}}$ and $\mathcal{X}_{\text{H}}$ and thus only report results for $\mathcal{X}_{\text{M}}$.

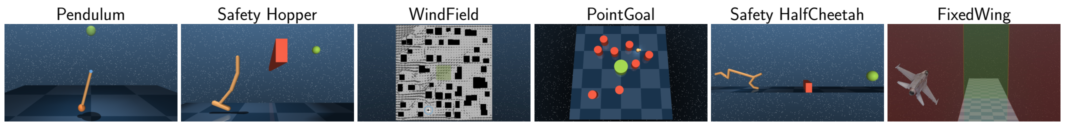


Figure 2: Illustrations of the benchmark tasks. In each picture, **red** denotes the unsafe region to be avoided, while **green** denotes the goal region to be reached.

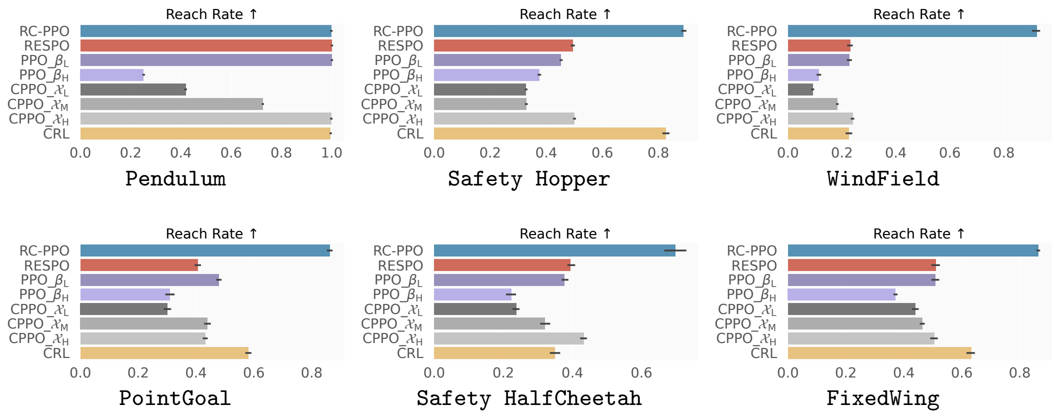


Figure 3: Reach rates of the final converged policies. RC-PPO consistently achieves the highest reach rates in all benchmark tasks.

We also consider the static Lagrangian multiplier case. In this setting, the reward function becomes $r(x_t) - \beta(\mathbb{1}_{x_t \in \mathcal{F}} \times C_{\text{fail}} + c(x_t, u_t))$ for a constant Lagrange multiplier β. We consider two different levels of β (β_{L}, β_{H}) in our experiments, resulting in the baselines PPO_β_{L} and PPO_β_{H}. More details are provided in Appendix E.

Benchmarks We compare **RC-PPO** with baseline methods on several minimum-cost reach-avoid environments. We consider an inverted pendulum (`Pendulum`), an environment from Safety Gym [54] (`PointGoal`) and two custom environments from MuJoCo [65], (`Safety Hopper`, `Safety HalfCheetah`) with added hazard regions and goal regions. We also consider a 3D quadrotor navigation task in a simulated wind field for an urban environment [69, 8] (`WindField`) and an Fixed-Wing avoid task from [58] with an additional goal region (`FixedWing`). More details on the benchmark can be found in Appendix F.

Evaluation Metrics Since the goal of RC-PPO is minimizing cost consumption while reaching goal without entering the unsafe region $\mathcal{F}$. We evaluate algorithm performance based on (i) reach rate, (ii) cost. The **reach rate** is the ratio of trajectories that enter goal region $\mathcal{G}$ without violating safety along the trajectory. The **cost** denotes the cumulative cost over the trajectory $\sum_{k=0}^{T} c(x_k, \pi(x_k))$.

5.1 Main Experiments

We first compare our algorithm with other baseline algorithms under a sparse reward setting. The comparison result is summarized in Figure 3. In all environments, the reach rate for all baseline algorithms is very low. Also, there is a general trend between the reach rate and Lagrangian coefficient. CPPO_$\mathcal{X}_{\text{L}}$ and PPO_β_{hi} have higher Lagrangian coefficients which leads to a lower reach rate.

5.2 Compare with Baselines with Reward Shaping Term

Reward shaping is a common method that can be used to improve the performance of RL algorithms, especially in the sparse reward setting [41, 49]. To see whether the same conclusions still hold even in the presence of reward shaping, we retrain the baseline methods but with reward shaping using a distance function-based potential function (see Appendix E for more details).

The results in Figure 4 demonstrate that RC-PPO remains competitive against the best baseline algorithms in reach rate while achieving significantly lower cumulative costs. The baseline methods

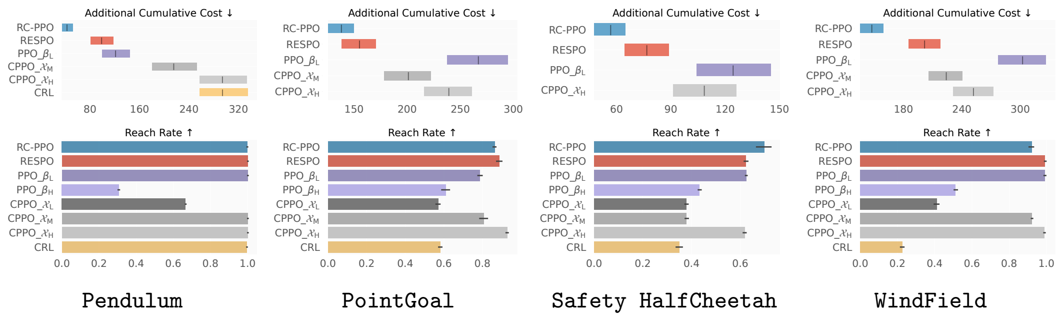
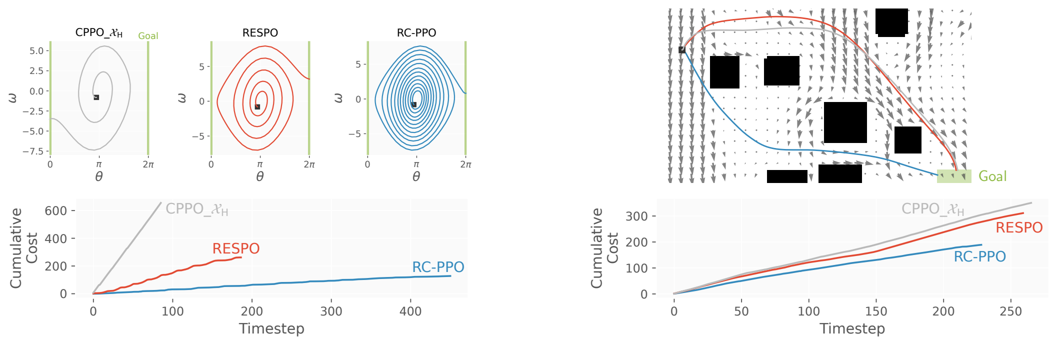


Figure 4: Cumulative cost (IQM) and reach rates of the final converged policies when reward shaping is used on four selected benchmarks. RC-PPO achieves significantly lower cumulative costs while retaining comparable reach rates even when compared with baseline methods that use reward shaping.



(a) `Pendulum` (b) `WindField`

Figure 5: On `Pendulum`, RC-PPO performs an extensive energy pumping strategy to reach the goal upright position (green line), resulting in vastly lower cumulative energy. On `WindField`, RC-PPO takes advantage instead of fighting against the wind field, resulting in a faster trajectory to the goal region (green box) that uses lower cumulative energy. The start of the trajectory is marked by ■.

(PPO_β_H, CPPO_$\mathcal{X}_L$) fail to achieve a high reach rate due to the large weights placed on minimizing the cumulative cost. CRL can reach the goal for simpler environments (`Pendulum`) but struggles with more complex environments. However, since goal-conditioned methods do not consider minimize cumulative cost, it achieves a higher cumulative cost relative to other methods. Other baselines focus more on goal-reaching tasks while putting less emphasis on the cost part. As a result, they suffer from higher costs than RC-PPO. We can also observe that RESPO achieves lower cumulative cost compared to CPPO_$\mathcal{X}_M$ which shares the same $\mathcal{X}_{\text{threshold}}$. This is due to RESPO making use of reachability analysis to better satisfy constraints.

To see how RC-PPO achieves lower costs, we visualize the resulting trajectories for `Pendulum` and `WindField` in Figure 5. For `Pendulum`, we see that RC-PPO learns to perform energy pumping to reach the goal in more time but with a smaller cumulative cost. The optimal behavior is opposite in the case of `WindField`, which contains an additional constant term in the cost to model the energy draw of quadcopters (see Appendix F). Here, we see that RC-PPO takes advantage of the wind at the beginning by moving *downwind*, arriving at the goal faster and with less cumulative cost.

We also visualize the learned RC-PPO policy for different values of z on the `Pendulum` benchmark (see Appendix G.2). For small values of z, the policy learns to minimize the cost, but at the expense of not reaching the goal. For large values of z, the policy reaches the goal quickly but at the expense of a large cost. The optimal z_{opt} found using the learned value function $\tilde{V}_{\hat{g}}^{\pi_\theta}$ finds the z that minimizes the cumulative cost but is still able to reach the goal.

6 Conclusion and Limitations

We have proposed RC-PPO, a novel reinforcement learning algorithm for solving minimum-cost reach-avoid problems. We have demonstrated the strong capabilities of RC-PPO over prior methods in solving a multitude of challenging benchmark problems, where RC-PPO learns policies that match the reach rates of existing methods while achieving significantly lower cumulative costs.

However, it should be noted that RC-PPO is not without limitations. First, the use of augmented dynamics enables folding the safety constraints within the goal specifications through an additional binary state variable. While this reduces the complexity of the resulting algorithm, it also means that two policies that are both unable to reach the goal can have the same value $\tilde{V}_{g'}^{\pi}$ even if one is unsafe, which can be undesirable. Next, the theoretical developments of RC-PPO are dependent on the assumptions of deterministic dynamics, which can be quite restrictive as it precludes the use of commonly used techniques for real-world deployment such as domain randomization. We acknowledge these limitations and leave resolving these challenges as future work.

References

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.

[2] Pulkit Agrawal. The task specification problem. In *Conference on Robot Learning*, pages 1745–1751. PMLR, 2022.

[3] Eitan Altman. *Constrained Markov decision processes*. Routledge, 2004.

[4] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[5] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253. IEEE, 2017.

[6] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[7] Vivek S Borkar and Vivek S Borkar. *Stochastic approximation: a dynamical systems viewpoint*, volume 9. Springer, 2008.

[8] Sanjeeb T Bose and George Ilhwan Park. Wall-modeled large-eddy simulation for complex turbulent flows. *Annual review of fluid mechanics*, 50:535–561, 2018.

[9] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[10] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[11] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[12] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR, 2019.

[13] Andres Campero, Roberta Raileanu, Heinrich Küttler, Joshua B Tenenbaum, Tim Rocktäschel, and Edward Grefenstette. Learning with amigo: Adversarially motivated intrinsic goals. *arXiv preprint arXiv:2006.12122*, 2020.

[14] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research*, 18(167):1–51, 2018.

[15] Yinlam Chow, Ofir Nachum, Aleksandra Faust, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031*, 2019.

[16] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[17] Jack Clark and Dario Amodei. Faulty reward functions in the wild, 2016.

[18] Philip G Drazin. *Nonlinear systems*. Number 10. Cambridge University Press, 1992.

[19] Damien Ernst, Mevludin Glavic, Florin Capitanescu, and Louis Wehenkel. Reinforcement learning versus model predictive control: a comparison on a power system problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):517–529, 2008.

[20] Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

[21] Lorenzo Fagiano and Andrew R Teel. Generalized terminal state constraint for model predictive control. *Automatica*, 49(9):2622–2631, 2013.

[22] Jaime F Fisac, Neil F Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8550–8556. IEEE, 2019.

[23] David Fischinger, Markus Vincze, and Yun Jiang. Learning grasps for unknown objects in cluttered scenes. In *2013 IEEE international conference on robotics and automation*, pages 609–616. IEEE, 2013.

[24] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[25] Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. *Advances in neural information processing systems*, 31, 2018.

[26] Milan Ganai, Zheng Gong, Chenning Yu, Sylvia Herbert, and Sicun Gao. Iterative reachability estimation for safe reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[27] Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019.

[28] Dhawal Gupta, Yash Chandak, Scott Jordan, Philip S Thomas, and Bruno C da Silva. Behavior alignment via reward function optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

[29] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[30] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[31] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[32] Haichao Hong, Arnab Maity, and Florian Holzapfel. Free final-time constrained sequential quadratic programming–based flight vehicle guidance. *Journal of Guidance, Control, and Dynamics*, 44(1):181–189, 2021.

[33] Kai Chieh Hsu, Vicenç Rubies-Royo, Claire J Tomlin, and Jaime F Fisac. Safety and liveness guarantees through reach-avoid reinforcement learning. In *17th Robotics: Science and Systems, RSS 2021*. MIT Press Journals, 2021.

[34] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[35] Ksenia Konyushkova, Konrad Zolna, Yusuf Aytar, Alexander Novikov, Scott Reed, Serkan Cabi, and Nando de Freitas. Semi-supervised reward learning for offline reinforcement learning. *arXiv preprint arXiv:2012.06899*, 2020.

[36] Jack Langelaan. Long distance/duration trajectory optimization for small uavs. In *AIAA guidance, navigation and control conference and exhibit*, page 6737, 2007.

[37] Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*, 2022.

[38] John Lygeros. On reachability and minimum cost optimal control. *Automatica*, 40(6):917–927, 2004.

[39] Kostas Margellos and John Lygeros. Hamilton–jacobi formulation for reach–avoid differential games. *IEEE Transactions on automatic control*, 56(8):1849–1861, 2011.

[40] Gabriel B Margolis, Tao Chen, Kartik Paigwar, Xiang Fu, Donghyun Kim, Sangbae Kim, and Pulkit Agrawal. Learning to jump from pixels. *arXiv preprint arXiv:2110.15344*, 2021.

[41] Maja J Mataric. Reward functions for accelerated learning. In *Machine learning proceedings 1994*, pages 181–189. Elsevier, 1994.

[42] Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. Discovering and achieving goals via world models. *Advances in Neural Information Processing Systems*, 34:24379–24391, 2021.

[43] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 50(7):947–957, 2005.

[44] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[45] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *Advances in neural information processing systems*, 31, 2018.

[46] Suraj Nair and Chelsea Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. *arXiv preprint arXiv:1909.05829*, 2019.

[47] Suraj Nair, Eric Mitchell, Kevin Chen, Silvio Savarese, Chelsea Finn, et al. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Conference on Robot Learning*, pages 1303–1315. PMLR, 2022.

[48] Suraj Nair, Silvio Savarese, and Chelsea Finn. Goal-aware prediction: Learning to model what matters. In *International Conference on Machine Learning*, pages 7207–7219. PMLR, 2020.

[49] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287, 1999.

[50] Robin M Pinson and Ping Lu. Trajectory design employing convex optimization for landing on irregularly shaped asteroids. *Journal of Guidance, Control, and Dynamics*, 41(6):1243–1256, 2018.

[51] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[52] Xuewei Qi, Yadan Luo, Guoyuan Wu, Kanok Boriboonsomsin, and Matthew Barth. Deep reinforcement learning enabled self-learning control for energy efficient driving. *Transportation Research Part C: Emerging Technologies*, 99:67–81, 2019.

[53] Mirco Rasotto, Roberto Armellin, and Pierluigi Di Lizia. Multi-step optimization strategy for fuel-optimal orbital transfer of low-thrust spacecraft. *Engineering Optimization*, 48(3):519–542, 2016.

[54] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7(1):2, 2019.

[55] Christoph Rösmann, Frank Hoffmann, and Torsten Bertram. Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control. In *2015 european control conference (ECC)*, pages 3352–3357. IEEE, 2015.

[56] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[57] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[58] Oswin So and Chuchu Fan. Solving stabilize-avoid optimal control via epigraph form and deep reinforcement learning. *arXiv preprint arXiv:2305.14154*, 2023.

[59] Kyle Stachowicz and Evangelos A Theodorou. Optimal-horizon model predictive control with differential dynamic programming. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1440–1446. IEEE, 2022.

[60] Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pages 9133–9143. PMLR, 2020.

[61] Hao Sun, Zhizhong Li, Xiaotong Liu, Bolei Zhou, and Dahua Lin. Policy continuation with hindsight inverse dynamics. *Advances in Neural Information Processing Systems*, 32, 2019.

[62] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[63] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[64] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018.

[65] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[66] Claire J Tomlin, John Lygeros, and S Shankar Sastry. A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949–970, 2000.

[67] Alexander Trott, Stephan Zheng, Caiming Xiong, and Richard Socher. Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

[68] David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. *arXiv preprint arXiv:1811.11359*, 2018.

[69] Steven Waslander and Carlos Wang. Wind disturbance estimation and rejection for quadrotor position control. In *AIAA Infotech@ Aerospace conference and AIAA unmanned... Unlimited conference*, page 1983, 2009.

[70] Yifan Wu, George Tucker, and Ofir Nachum. The laplacian in rl: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*, 2018.

[71] Annie Xie, Avi Singh, Sergey Levine, and Chelsea Finn. Few-shot goal inference for visuomotor learning and planning. In *Conference on Robot Learning*, pages 40–52. PMLR, 2018.

[72] Danfei Xu and Misha Denil. Positive-unlabeled reward learning. In *Conference on Robot Learning*, pages 205–219. PMLR, 2021.

[73] Dongjie Yu, Haitong Ma, Shengbo Li, and Jianyu Chen. Reachability constrained reinforcement learning. In *International Conference on Machine Learning*, pages 25636–25655. PMLR, 2022.

[74] Ying Zhang, Tao You, Jinchao Chen, Chenglie Du, Zhaoyang Ai, and Xiaobo Qu. Safe and energy-saving vehicle-following driving decision-making framework of autonomous vehicles. *IEEE Transactions on Industrial Electronics*, 69(12):13859–13871, 2021.

[75] Konrad Zolna, Scott Reed, Alexander Novikov, Sergio Gomez Colmenarejo, David Budden, Serkan Cabi, Misha Denil, Nando de Freitas, and Ziyu Wang. Task-relevant adversarial imitation learning. In *Conference on Robot Learning*, pages 247–263. PMLR, 2021.

A Optimal Reach Value Function

As shown in (13), we introduce an additional discount factor into the estimation of $\tilde{V}_{\hat{g}}^{\pi}$. It will incur imprecision on the calculation of $\tilde{V}_{\hat{g}}^{\pi}$ defined in Definition 1. In this section, we show that for a large enough discount factor $\gamma < 1$, we could reach unbiased $\tilde{z}$ in phase two of **RC-PPO**.

Theorem 3. We denote $\max_{\hat{x}\in\hat{\mathcal{X}}}\{\hat{g}(\hat{x})\} = G_{max}$ and maximal episode length T_{max}. If there exists a positive value ϵ where

$$\hat{g}(\hat{x}) < 0 \Rightarrow \hat{g}(\hat{x}) < -\epsilon.$$

Then for any $\frac{\gamma^{T_{max}}}{1-\gamma^{T_{max}}} > \frac{G_{max}}{\epsilon}$, for any deterministic policy π satisfies 13. If there exists a trajectory under given policy π leading to the extended goal region $\hat{\mathcal{G}}$. We have

$$\tilde{V}_{\hat{g}}^{\pi}(\hat{x}) < 0.$$

The proof for Theorem 3 is provided in Appendix C.4.

B GAE estimator Definition

Note, however, that the definition of return (14) is *different* from the original definition and hence will result in a different equation for the GAE.

To simplify the form of the GAE, we first define a "reduction" function $\phi^{(n)} : \mathbb{R}^n \to \mathbb{R}$ that applies itself recursively to its n arguments, i.e.,

$$\phi^{(n)}(x_1, x_2, \dots, x_n) \coloneqq \phi^{(1)}\left(x_1,\, \phi^{(n-1)}(x_2, \dots, x_n)\right)$$

where

$$\phi^{(1)}(x, y) \coloneqq (1-\gamma)x + \gamma \min\{x, y\}.$$

The k-step advantage function $\hat{A}_{\hat{g}}^{\pi(k)}$ can then be written as

$$\hat{A}_{\hat{g}}^{\pi(k)}(\hat{x}_t) = \phi^{(k)}\left(\hat{g}(\hat{x}_t), \dots, \hat{g}(\hat{x}_{t+k-1}), \tilde{V}_{\hat{g}}^{\pi}(\hat{x}_{t+k})\right) - \tilde{V}_{g}^{\pi}(\hat{x}_t).$$

We can then construct the GAE $\hat{A}_{\hat{g}}^{\pi(\mathrm{GAE})}$ as the λ^k-weighted sum over the k-step advantage functions $\hat{A}_{\hat{g}}^{\pi(k)}$: Overall, the GAE estimator can be described as

$$\hat{A}_{\hat{g}}^{\pi(\mathrm{GAE})}(\hat{x}_t) = \frac{1}{1-\lambda}\sum_{k=1}^{\infty} \lambda^k \hat{A}_{\hat{g}}^{\pi(k)}(\hat{x}_t).$$

C Proofs

C.1 Proof for Theorem 1

Proof. We separately consider three elements in augmented state (x_T, y_T, z_T). First, note that 1b holds if and only if $x_T \in \mathcal{G}$. For the second element y, from the definition of the augmented dynamics 4, it holds that

$$y_T = \max_{i\in\{0,\dots,T\}} \mathbb{I}_{x_i \in \mathcal{F}} \tag{17}$$

As a result 1c holds if and only if $y_T = -1$. For the third element z, note that $z_T = z_0 - \sum_{k=0}^{T-1} c(x_k, u(x_k))$. Hence, $z_T \geq 0$ if and only if $z_0 \geq \sum_{k=0}^{T-1} c(x_k, u(x_k))$. □

C.2 Proof for Property 7

Proof. From Definition 7, we know

$$\begin{aligned}\tilde{V}_{\hat{g}}^{\pi}(\hat{x}) &= \min_{t\in\mathbb{N}} \hat{g}(\hat{x}_t \mid \hat{x}_0 = \hat{x}) \\ &= \min\{\hat{g}(\hat{x}), \min_{t\in\mathbb{N}^+} \hat{g}(\hat{x}_t \mid \hat{x}_0 = \hat{x})\} \\ &= \min\{\hat{g}(\hat{x}), \tilde{V}_{g}^{\pi}(\hat{x}_{t+1})\}\end{aligned}$$

□

C.3 Proof for Theorem 2

We first derive the state value function in a recursive form similar as [62]

Proof.

$$
\begin{aligned}
\nabla_\theta \tilde{V}_{\hat{g}}^{\pi_\theta}(\hat{x}) =& \nabla_\theta \left(\sum_{u \in \mathcal{U}} \pi_\theta(u \mid \hat{x}) \tilde{Q}_{\hat{g}}^{\pi_\theta}(\hat{x}, u) \right) \\
=& \sum_{u \in \mathcal{U}} \left(\nabla_\theta \pi_\theta(u \mid \hat{x}) \tilde{Q}_{\hat{g}}^{\pi_\theta}(, u) + \pi_\theta(u \mid \hat{x}) \nabla_\theta \tilde{Q}_{\hat{g}}^{\pi_\theta}(\hat{x}, u) \right) \\
=& \sum_{u \in \mathcal{U}} \Bigg(\nabla_\theta \pi_\theta(u \mid \hat{x}) \tilde{Q}_{\hat{g}}^{\pi_\theta}(\hat{x}, u) \\
& \qquad + \pi_\theta(u \mid \hat{x}) \nabla_\theta \min\{\hat{g}(\hat{x}), \tilde{V}_g^{\pi}(\hat{x}')\} \Bigg) \\
=& \sum_{u \in \mathcal{U}} \Bigg(\nabla_\theta \pi_\theta(u \mid \hat{x}) \tilde{Q}_{\hat{g}}^{\pi_\theta}(\hat{x}, u) \\
& \qquad + \pi_\theta(u \mid \hat{x}) \mathbb{1}_{\hat{g}(\hat{x}) > \tilde{V}_g^{\pi_\theta}(\hat{x}')} \nabla_\theta \tilde{V}_g^{\pi_\theta}(\hat{x}') \Bigg)
\end{aligned}
$$

where $\hat{x}' = \hat{f}(\hat{x}, u)$

Next, we consider unrolling $\tilde{V}_g^{\pi_\theta}(\hat{x}')$ under Reachability MDP in Definition 2. We define $\Pr(\hat{x} \to \hat{x}^\dagger, k, \pi_\theta)$ as the probability of transitioning from state $\hat{x}$ to $\hat{x}^\dagger$ in k steps under policy π_θ in 2. Note that $\mathbb{1}_{\hat{g}(\hat{x}) > \tilde{V}_g^{\pi_\theta}(\hat{x}')}$ is absorbed using the absorbing state in 2. Then we can get

$$
\begin{aligned}
\nabla_\theta \tilde{V}_{\hat{g}}^{\pi_\theta}(\hat{x}) =& \sum_{\hat{x}^\dagger \in \hat{\mathcal{X}}} \left(\sum_{k=0}^{\infty} \Pr\left(\hat{x} \to \hat{x}^\dagger, k, \pi\right) \right) \sum_{u \in \mathcal{U}} \nabla \pi_\theta(u \mid \hat{x}^\dagger) \tilde{Q}_{\hat{g}}^{\pi_\theta}(\hat{x}^\dagger, u) \\
& \propto \mathbb{E}_{\hat{x}' \sim d'_\pi(\hat{x}), u \sim \pi_\theta} \left[\tilde{Q}^{\pi_\theta}\left(\hat{x}', u\right) \nabla_\theta \ln \pi_\theta\left(u \mid \hat{x}'\right) \right]
\end{aligned}
$$

□

C.4 Proof for Theorem 3

Proof. Consider trajectory $\{\hat{x}_0, \ldots, \hat{x}_T\}$ where $\hat{x}_T \in \hat{\mathcal{G}}$. We consider the worst-case scenario where $\hat{g}(\hat{x}_t) = g_{max}$ for $t \in \{0, \ldots, T-1\}$. Then

$$
\begin{aligned}
\tilde{V}^\pi(\hat{x}_0) &= (1-\gamma)\hat{g}(\hat{x}_0) + \gamma \min\{\tilde{V}^\pi(\hat{x}_1), \hat{g}(\hat{x}_1)\} \\
&\leq (1-\gamma) g_{max} + \gamma \tilde{V}^\pi(\hat{x}_1) \\
&\leq (1-\gamma) g_{max} + \gamma((1-\gamma) g_{max} + \gamma \tilde{V}^\pi(\hat{x}_1)) \\
&\leq \sum_{i=0}^{T-1} \gamma^i (1-\gamma) g_{max} + \gamma^T \tilde{V}^\pi(\hat{x}_T) \\
&< (1-\gamma^T) g_{max} + \gamma^T \epsilon \\
&< 0
\end{aligned}
$$

□

D Convergence Guarantee on an Actor-Critic Version of Our Method

In this section, we provide the convergence proof of phase one of our method under the actor-critic framework. Notice that similar to Bellman equation (13) for $\tilde{V}_{\hat{g}}^{\pi}$. We could also derive the Bellman equation for $\tilde{Q}_{\hat{g}}^{\pi}$ as

$$\tilde{Q}_{\hat{g}}^{\pi}(\hat{x}_t, u_t) = (1-\gamma)\hat{g}(\hat{x}_t) + \gamma \mathbb{E}_{\hat{x}_{t+1}\sim\tau, u_{t+1}\sim\pi}[\min\{\hat{g}(\hat{x}_t), \tilde{Q}_{g}^{\pi}(\hat{x}_{t+1}, u_{t+1})\}]$$

Next, we show our method under the actor-critic framework without GAE estimator in Algorithm 3

Algorithm 3 RC-PPO (Actor Critic)

```
Require: Initial policy parameter $\theta_0$, Q function parameter $\omega_0$, horizon T, convex projection operator
    $\Gamma_\Theta$, and value function learning rate $\beta_1(k)$, policy learning rate $\beta_2(k)$
1: for k = 0, 1, ... do
2:     for t = 0 to T-1 do
3:         Sample trajectories $\tau_t : \{\hat{x}_t, u_t, \hat{x}_{t+1}\}$
4:         Critic update: $\omega_{k+1} = \omega_k - \beta_1(k)\nabla_\omega \tilde{Q}_{\hat{g}}(\hat{x}_t, u_t; \omega_k) \cdot$
5:         $\left[\tilde{Q}_{\hat{g}}(\hat{x}_t, u_t; \omega_k) - \left((1-\gamma)\hat{g}(\hat{x}_t) + \gamma \min\left\{\hat{g}(\hat{x}_t), \tilde{Q}_{\hat{g}}(\hat{x}_{t+1}, u_{t+1}; \omega_k)\right\}\right)\right]$
6:         Actor Update: $\theta_{k+1} = \Gamma_\Theta\left(\theta_k + \beta_2(k)\tilde{Q}_{\hat{g}}(\hat{x}_t, u_t; \omega_k)\nabla_\theta \log \pi_\theta(u_t \mid \hat{x}_t)\right)$
7:     end for
8: end for
9: return parameter $\theta, \omega$
```

In this algorithm, the $\Gamma_\Theta(\theta)$ operator projects a vector $\theta \in \mathbb{R}^k$ to the closest point in a compact and convex set $\Theta \subset \mathbb{R}^k$, i.e., $\Gamma_\Theta(\theta) = \arg\min_{\theta' \in \Theta} \|\theta' - \theta\|^2$.

Next, we provide the convergence analysis for Algorithm 3 under the following assumptions.

Assumption 1. (Step Sizes) The step size schedules $\{\beta_1(k)\}$ and $\{\beta_2(k)\}$ have below properties:

$$\sum_k \beta_1(k) = \sum_k \beta_2(k) = \infty$$
$$\sum_k \beta_1(k)^2, \sum_k \beta_2(k)^2 < \infty$$
$$\beta_2(k) = o(\beta_1(k))$$

Assumption 2. (Differentiability and and Lipschitz Continuity) For any state and action pair $(\hat{x}, u)$, $\tilde{Q}_{\hat{g}}(\hat{x}, u; \omega)$ and $\pi(\hat{x}; \theta)$ are continuously differentiable in ω and θ. Furthermore, for any state and action pair $(\hat{x}, u)$, $\nabla_\omega \tilde{Q}_{\hat{g}}(\hat{x}, u; \omega)$ and $\pi(\hat{x}; \theta)$ are Lipschitz function in ω and θ.

Also, we assume that $\mathcal{X}$ and $\mathcal{U}$ are finite and bounded and the horizon T is also bounded by $T_{\max}$, then the cost function c can be bounded by $C_{\max}$ and g can be bounded within $G_{\max}$. We can limit the space of cost upper bound $z \in [-G_{\max}, T \cdot C_{\max}]$ instead of $\mathbb{R}$. This is due to $\hat{g}(\hat{x}) = -z$ for $z \leq -G_{\max}$. Next, we could do discretization on $[-G_{\max}, T \cdot C_{\max}]$ and cost function c to make the augmented state set $\hat{\mathcal{X}}$ finite and bounded.

With the above assumptions, we can provide a convergence guarantee for Algocrithm 3.

Theorem 4. Under Assumptions 1 and 2, the policy update in Algorithm 3 converge almost surely to a locally optimal policy.

Proof. We show our algorithm convergence to the optimal policy by utilizing the proof framework of multi-time scale presented in [7, 14, 15, 26, 73]. Specifically, we have 2 time scales for the critics and the policy listed in order from fastest to slowest. The overview of each timescale proof step is as follows:

- First, we prove that the critic parameter almost surely converges to a fixed point ω^*.

- Second, due to the fast convergence of ω^*, we can show policy paramter θ converge almost surely to a stationary point θ^* which can be further proved to be a locally optimal solution.

Step 1: (convergence of the critics ω updates): From the multi-time scale assumption, we know that ω will convergence on a faster time scale than the other parameters θ. Therefore, we can leverage Lemma 1 of Chapter 6 of [7] to analyze the convergence properties while updating ω_k by treating θ as fixed constant θ_k. In other words, the policy are fixed while computing $\tilde{Q}_{\hat{g}}(\hat{x}, u)$. With the Finite MDP assumption and policy evaluation convergence results of [62], and assuming sufficiently expressive function approximator (i.e. wide enough neural networks) to ensure convergence to global minimum, we can use the fact that the bellman operators $\mathcal{B}$ which is defined as

$$\mathcal{B}[\tilde{Q}_{\hat{g}}(\hat{x}, u)] = (1-\gamma)\hat{g}(\hat{x}) + \gamma \mathbb{E}_{\hat{x}' \sim \tau, u \sim \pi}[\min\{\hat{g}(\hat{x}), \tilde{Q}_{\hat{g}}(\hat{x}', u)\}]$$

is γ-contraction mappings, and therefore as k approaches ∞, we can be sure that $\tilde{Q}_{\hat{g}}(\hat{x}, u; \omega) \to \tilde{Q}_{\hat{g}}(\hat{x}, u; \omega^*) = \tilde{Q}_{\hat{g}}^{\pi_\theta}(\hat{x}, u)$. So since ω_k converges to ω^*, we prove the convergence of the critic in Time scale 1.

Step 2: (convergence of the policy π_θ update): In Time scale 2, we have $\left\|\tilde{Q}_{\hat{g}}(\hat{x}, u; \omega) - \tilde{Q}_{\hat{g}}(\hat{x}, u; \omega^*)\right\| \to 0$ almost surely. Now the update of the policy θ using the gradient from Algorithm 3 is:

$$\begin{aligned}
\theta_{k+1} &= \Gamma_\Theta\left[\theta_k + \beta_2(k)\left(\nabla_\theta L\left(\theta, \omega_k\right)|_{\theta=\theta_k}\right)\right] \\
&= \Gamma_\Theta\left[\theta_k + \beta_2(k)\left(\tilde{Q}_{\hat{g}}\left(\hat{x}_t, u_t; \omega_k\right) \nabla_\theta \log \pi\left(u_t \mid \hat{x}_t; \theta\right)|_{\theta=\theta_k}\right)\right] \\
&= \Gamma_\Theta\left[\theta_k + \beta_2(k)\left(\nabla_\theta L\left(\theta, \omega^*\right)|_{\theta=\theta_k} + \delta\theta_{k+1} + \delta\theta_\epsilon\right)\right]
\end{aligned}$$

where

$$\begin{aligned}
\delta\theta_{k+1} = &- \mathbb{E}_{\hat{x} \sim \mathcal{D}}\left[\tilde{Q}_{\hat{g}}(\hat{x}, u; \omega_k)|_{u=\pi(\hat{x}_t;\theta_k)} \nabla_\theta \log \pi(\hat{x}_t; \theta)|_{\theta=\theta_k}\right] \\
&+ \tilde{Q}_{\hat{g}}(\hat{x}, u; \omega_k)|_{u=u_t} \cdot \nabla_\theta \log \pi\left(\hat{x}_t; \theta\right)|_{\theta=\theta_k}
\end{aligned}$$

and

$$\begin{aligned}
\delta\theta_\epsilon = \mathbb{E}_{\hat{x} \sim \mathcal{D}}\Big[&\tilde{Q}_{\hat{g}}(\hat{x}, u; \omega_k)|_{u=\pi(\hat{x}_t;\theta_k)} \nabla_\theta \log \pi(\hat{x}_t; \theta)|_{\theta=\theta_k} \\
&- \tilde{Q}_{\hat{g}}(\hat{x}, u; \omega^*)|_{u=\pi(\hat{x}_t;\theta_k)} \nabla_\theta \log \pi(\hat{x}_t; \theta)|_{\theta=\theta_k}\Big]
\end{aligned}$$

Lemma 1: We can first demonstrate that $\delta\theta_{k+1}$ is square integrable. In particular,

$$\begin{aligned}
&\mathbb{E}\left[\|\delta\theta_{k+1}\|^2 \mid \mathcal{F}_{\theta,k}\right] \\
&\leq 4\left\| \nabla_\theta \log \pi(u \mid \hat{x}_t; \theta)|_{\theta=\theta_k} \mathbb{1}_{\pi(u|\hat{x}_t;\theta_k)>0}\right\|_\infty^2 \cdot \left\|\tilde{Q}_{\hat{g}}(\hat{x}, u; \omega_k)\right\|_\infty^2 \\
&\leq 4\frac{\left\| \nabla_\theta \pi(u \mid \hat{x}_t; \theta)|_{\theta=\theta_k}\right\|_\infty^2}{\min\left\{\pi\left(u \mid \hat{x}_t; \theta_k\right) \mid \pi\left(u \mid \hat{x}_t; \theta_k\right) > 0\right\}} \cdot \left\|\tilde{Q}_{\hat{g}}(\hat{x}, u; \omega_k)\right\|_\infty^2
\end{aligned}$$

Note that $\mathcal{F}_{\theta,k} = \sigma\left(\theta_m, \delta\theta_m, m \leq k\right)$ is the filtration for θ_k generated by different independent trajectories [14]. From Assumptions 2, finite MDP, bounded cost function c, and goal indication function g, we can bound the values of the functions and the gradients of functions. Specifically

$$\begin{aligned}
\left\| \nabla_\theta \pi(u \mid \hat{x}; \theta)|_{\theta=\theta_k}\right\|_\infty^2 &\leq K_1\left(1 + \|\theta_k\|_\infty^2\right), \\
\left\|\tilde{Q}_{\hat{g}}(\hat{x}, u; \omega_k)\right\|_\infty^2 &\leq G_{\max}
\end{aligned}$$

where K_1 is a Lipschitz constant. Furthermore, note that because we are sampling, $\pi(u \mid \hat{x}; \theta_k)$ will take on only a finite number of values, so its nonzero values will be bounded away from zero. Thus we can say

$$\frac{1}{\min\{\pi(a \mid s; \theta_k) \mid \pi(a \mid s; \theta_k) > 0\}} \leq K_2$$

for some large enough K_2. Thus using the bounds from these conditions, we can demonstrate

$$\mathbb{E}\left[\|\delta\theta_{k+1}\|^2 \mid \mathcal{F}_{\theta,k}\right] \leq 4 \cdot K_1 \cdot K_2 \cdot G_{\max}\left(1 + \|\theta_k\|_\infty^2\right) < \infty$$

Therefore $\delta\theta_{k+1}$ is square integrable.

Lemma 2: Secondly, we can demonstrate $\delta\theta_\epsilon \to 0$.

$$\begin{aligned}\delta\theta_\epsilon =& \mathbb{E}_{\hat{x}\sim\mathcal{D}}\Big[\tilde{Q}_{\hat{g}}(\hat{x}, u; \omega_k)|_{u=\pi(\hat{x}_t;\theta_k)} \nabla_\theta \log \pi(\hat{x}_t; \theta)|_{\theta=\theta_k} \\ & - \tilde{Q}_{\hat{g}}(\hat{x}, u; \omega^*)|_{u=\pi(\hat{x}_t;\theta_k)} \nabla_\theta \log \pi(\hat{x}_t; \theta)|_{\theta=\theta_k}\Big] \\ \leq & \mathbb{E}_{\hat{x}\sim\mathcal{D}}\left[4\frac{\|\nabla_\theta \pi(u \mid \hat{x}_t; \theta)|_{\theta=\theta_k}\|_\infty^2}{\min\{\pi(u \mid \hat{x}_t; \theta_k) \mid \pi(u \mid \hat{x}_t; \theta_k) > 0\}} \cdot \|Q(\hat{x}, u; \omega_k) - Q(\hat{x}, u; \omega^*)\|\right]\end{aligned}$$

And because we have $\|Q(\hat{x}, u; \omega_k) - Q(\hat{x}, u; \omega^*)\| \to 0$, we can therefore say $\delta\theta_\epsilon \to 0$.

Lemma 3: Finally, since $\bar{\nabla}_\theta J_\pi(\theta)\big|_{\theta=\theta_k}$ is a sample of $\nabla_\theta L(\theta, \omega_k)|_{\theta=\theta_k}$ based on the history of sampled trajectories, we conclude that $\mathbb{E}[\delta\theta_{k+1} \mid \mathcal{F}_{\theta,k}] = 0$.

From the 3 above lemmas, the policy θ update is a stochastic approximation of a continuous system $\theta(t)$ [7], described by an ODE

$$\dot{\theta} = \Upsilon_\Theta\left[-\nabla_\theta L(\theta, \omega_k)\right] \tag{18}$$

in which

$$\Upsilon_\Theta[M(\theta)] \triangleq \lim_{\psi\to 0^+} \frac{\Gamma_\Theta(\theta + \psi M(\theta)) - \Gamma_\Theta(\theta)}{\psi}$$

or in other words the left directional derivative of $\Gamma_\Theta(\theta)$ in the direction of $M(\theta)$. Using the left directional derivative $\Upsilon_\Theta[-\nabla_\theta L(\theta, \omega)]$ in the gradient descent algorithm for learning the policy π_θ ensures the gradient will point in the descent direction along the boundary of Θ when the θ update hits its boundary. Using Step 2 in Appendix A. 2 from [7], we have that $dL(\theta, \omega)/dt = -\nabla_\theta L(\theta, \omega)^T \cdot \Upsilon_\Theta[-\nabla_\theta L(\theta, \omega)] \leq 0$ and the value is non-zero if $\|\Upsilon_\Theta[-\nabla_\theta L(\theta, \omega)]\| \neq 0$. Now consider the continuous system $\theta(t)$. For some fixed ω, define a Lyapunov function

$$\mathcal{L}_\omega(\theta) = L(\theta, \omega) - L(\theta^*, \omega)$$

where θ^* is a local minimum point. Then there exists a ball centered at θ^* with a radius ρ such that $\forall\theta \in \mathfrak{B}_{\theta^*}(\rho) = \{\theta \mid \|\theta - \theta^*|\ | \leq \rho\}$, $\mathcal{L}_\omega(\theta)$ is a locally positive definite function, that is $\mathcal{L}_\omega(\theta) \geq 0$. Using Proposition 1.1.1 from [6], we can show that $\Upsilon_\Theta[-\nabla_\theta L(\theta, \omega)]|_{\theta=\theta^*} = 0$ meaning θ^* is a stationary point. Since $dL(\theta, \omega)/dt \leq 0$, through Lyapunov theory for asymptotically stable systems presented in Chapter 4 of [18], we can use the above arguments to demonstrate that with any initial conditions of $\theta(0) \in \mathfrak{B}_{\theta^*}(\rho)$, the continuous state trajectory of $\theta(t)$ converges to θ^*. Particularly, $L(\theta^*, \omega) \leq L(\theta(t), \omega) \leq L(\theta(0), \omega)$ for all $t > 0$.

Using these aforementioned properties and below facts

- $\nabla_\theta L(\theta, \omega)$ is a Lipschitz function (using Proposition 17 from [7])
- the step-sizes of Assumption 1

- $\delta\theta_{k+1}$ is a square integrable Martingale difference sequence and $\delta\theta_\epsilon$ is a vanishing error almost surely
- $\theta_k \in \Theta, \forall k$ implying that $\sup_k \|\theta_k\| < \infty$ almost surely

We can invoke Theorem 2 of chapter 6 in [7] to demonstrate the sequence $\{\theta_k\}, \theta_k \in \Theta$ converges almost surely to the solution of the ODE defined by (18), which additionally converges almost surely to the local minimum $\theta^* \in \Theta$.

□

E Implementation Details of Algorithms

In this section, we will provide more details about CMDP-based baselines (different between optimization goal with multiple constraints) and other hyperparameter settings like $\mathcal{X}_{\text{threshold}}$.

E.1 CMDP-based Baselines

In this section, we will clarify the optimization target for CPPO and RESPO under CMDP formulation of both hard and soft constraints. Recall that our formulation of CMDP is

$$\min_\pi \quad \mathbb{E}_{x_t,u_t\sim d_\pi} \sum_t \left[-\gamma^t r(x_t,u_t)\right] \tag{19a}$$

$$\text{s.t.} \quad \mathbb{E}_{x_t,u_t\sim d_\pi} \sum_t \left[\gamma^t \mathbb{1}_{x_t\in\mathcal{F}} \times C_{fail}\right] \le 0, \tag{19b}$$

$$\mathbb{E}_{x_t,u_t\sim d_\pi} \sum_t \left[\gamma^t c(x_t,u_t)\right] \le \mathcal{X}_{\text{threshold}} \tag{19c}$$

We then denote

$$V_r^\pi(x_t) \coloneqq \mathbb{E}_{x_t,u_t\sim d_\pi} \sum_t \left[\gamma^t r(x_t,u_t)\right]$$

$$V_f^\pi(x_t) \coloneqq \mathbb{E}_{x_t,u_t\sim d_\pi} \sum_t \left[\gamma^t \mathbb{1}_{x_t\in\mathcal{F}} \times C_{cost}\right]$$

$$V_c^\pi(x_t) \coloneqq \mathbb{E}_{x_t,u_t\sim d_\pi} \sum_t \left[\gamma^t c(x_t,u_t)\right]$$

The optimization goal formulation for CPPO is as follows:

$$\min_\pi \max_\lambda \left(L(\pi,\lambda) = -V_r^\pi(x) + \lambda_c \cdot (V_c^\pi(x) - \mathcal{X}_{threshold}) + \lambda_f \cdot V_f^\pi(x)\right)$$

In this formulation, the soft constraint V_c^π has the same priority as the hard constraint V_f^π. This leads to a potential imbalance between soft constraints and hard constraints. Instead, the optimization goal for RESPO is as follows:

$$\min_\pi \max_\lambda L(\pi,\lambda) = \big(-V_r^\pi(x) + \lambda_c \cdot (V_c^\pi(x) - \mathcal{X}_{threshold}) + \lambda_f \cdot V_f^\pi(x)\big) \cdot (1-p(x)) + p(x) \cdot V_f^\pi(x)$$

where $p(x)$ denotes the probability of entering the unsafe region $\mathcal{F}$ start from state x. It is called the reachability estimation function (REF). This formulation prioritizes the satisfaction of hard constraints but still suffers from balancing soft constraints and reward terms.

E.2 Hyperparameters

We first clarify how we set proper $\mathcal{X}_{\text{threshold}}$ for each environment. First, we will run our method RC-PPO and calculate the average cost, we denote it as $c_{average}$. We set $\mathcal{X}_{\text{low}} = \frac{c_{average}}{10}$, $\mathcal{X}_{\text{medium}} = \frac{c_{average}}{3}$ and $\mathcal{X}_{\text{high}} = c_{average}$. For static lagrangian multiplier β, we set $\beta_{\text{lo}} = 0.1$ and $\beta_{\text{hi}} = 10$. Also, we set $C_{fail} = 20$ in every environment.

Note that CRL is an off-policy algorithm, while RC-PPO and other baselines are on-policy algorithms. We provide Table 1 showing hyperparameters for on-policy algorithms and Table 2 showing hyperparameters for off-policy algorithm (CRL).

Table 1: Hyperparameter Settings for On-policy Algorithms

Hyperparameters for On-policy Algorithms	Values
On-policy parameters	
Network Architecture	MLP
Units per Hidden Layer	256
Numbers of Hidden Layers	2
Hidden Layer Activation Function	tanh
Entropy coefficient	Linear Decay 1e-2 $\to$ 0
Optimizer	Adam
Discount factor γ	0.99
GAE lambda parameter	0.95
Clip Ratio	0.2
Total Env Interactions	2e7
Actor Learning rate	Linear Decay 3e-4 $\to$ 0
Reward/Cost Critic Learning rate	Linear Decay 3e-4 $\to$ 0
RESPO specific parameters	
REF Output Layer Activation Function	sigmoid
Lagrangian multiplier Output Layer Activation function	softplus
Lagrangian multiplier Learning rate	Linear Decay 5e-5 $\to$ 0
REF Learning Rate	Linear Decay 1e-4 $\to$ 0
CPPO specific parameters	
K_P	1
K_I	1e-4
K_D	1

Table 2: Hyperparameter Settings for Off-policy Algorithms

Hyperparameters for Off-policy Algorithms	Values
Off-policy parameters	
Network Architecture	MLP
Units per Hidden Layer	256
Numbers of Hidden Layers	2
Hidden Layer Activation Function	tanh
Entropy target	-2
Optimizer	Adam
Discount factor γ	0.99
Total Env Interactions	2e6
Actor Learning rate	Linear Decay 3e-4 $\to$ 0
Critic Learning rate	Linear Decay 3e-4 $\to$ 0
Actor Target Entropy	0
Replay Buffer Size	1e6 transitions
Replay Batch Size	256
Train-Collect Interval	16
Target Smoothing Term	0.005

E.3 Implementation of the baselines

The implementation of the baseline follows their original implementations:

- RESPO: https://github.com/milanganai/milanganai.github.io/tree/main/NeurIPS2023/code (No license)
- CRL: `https://github.com/google-research/google-research/tree/master/contrastive_rl` (No License)

F Experiment Details

In this section, we provide more details about the benchmarks and the choice of reward function r, g, cost function c and C_{cost} in each environment. Under the sparse reward setting, we apply the following structure of reward design

$$r(x_t, u_t, x_{t+1}) = R_{goal} \times \mathbb{1}_{x_{t+1} \in \mathcal{G}}$$

where R_{goal} is an constant. After doing reward shaping, we add an extra term $\gamma\phi(x_{t+1}) - \phi(x_t)$ and the reward becomes

$$r(x_t, u_t, x_{t+1}) = R_{goal} \times \mathbb{1}_{x_{t+1} \in \mathcal{G}} + \gamma\phi(x_{t+1}) - \phi(x_t)$$

where γ denotes the discount factor.

Note that we set $R_{goal} = C_{cost} = 20$ in all the environments. Note that if there is a gap between $\max\{g(x) \mid g(x) < 0\}$, we could get unbiased $\tilde{z}$ during phase two of RC-PPO guaranteed by Theorem 3. To achieve better performance in phase two of RC-PPO, we set

$$g(x) = -300$$

for all $x \in \mathcal{G}$ to maintain such a gap. Also, we implement all the environments in Jax [10] for better scalability and parallelization.

F.1 Pendulum

The Pendulum environment is taken from Gym [11] and the torque limit is set to be 1. The state space is given by $x = [\theta, \dot{\theta}]$ where $\theta \in [-\pi, \pi], \dot{\theta} \in [-8, 8]$. In this task, we do not consider unsafe regions and set

$$\mathcal{G} \coloneqq \{[\theta, \dot{\theta}] \mid \theta \cdot (\theta + \dot{\theta} \cdot dt) < 0\}$$

where $dt = 0.05$ is the time interval during environment simulation. This is for preventing environment overshooting during simulation.

In the Pendulum environment, cost function c is given by

$$c(x_t, u_t, x_{t+1}) = \begin{cases} 0 & \text{if} \quad \|u_t\| < 0.1 \\ 8\|u\|^2 & \text{if} \quad \|u_t\| \geq 0.1 \end{cases}$$

for better visualization of policies with different energy consumption. g is given by

$$g(x) = \begin{cases} 100\theta^2 & \text{if} \quad x \notin \mathcal{G} \\ -300 & \text{if} \quad x \in \mathcal{G} \end{cases}$$

F.2 Safety Hopper

The Safety Hopper environment is taken from Safety Mujoco, we add static obstacles in the environment to increase the difficulty of the task. We use x to denote the x-axis position of the head of Hopper, y to be the y-axis position of the head of Hopper. Then the goal region can be described as

$$\mathcal{G} \coloneqq \{(x, y) \mid \|[x, y] - [2.0, 1.4]\| < 0.1\}$$

The unsafe set is described as

$$\mathcal{F} \coloneqq \{(x, y) \mid 0.95 \leq x \leq 1.05, y \geq 1.3\}$$

We use $\tilde{x}^{thigh}, \tilde{x}^{leg}, \tilde{x}^{foot}$ to denote the angular velocity of the thigh, leg, foot hinge. The cost function is described as

$$c(x_t, u_t, x_{t+1}) = l(x_t^{thigh}, u_t^1) + l(x_t^{leg}, u_t^2) + l(x_t^{foot}, u_t^3)$$

where

$$l(a,b) = \begin{cases} 0 & \text{if} \quad \|a \cdot b\| < 0.4 \\ 0.15a^2 \cdot b^2 & \text{if} \quad \|a \cdot b\| > 0.4 \end{cases}$$

g is given by

$$g(\tilde{x}) = \begin{cases} 100\sqrt{(x-2)^2 + 100(y-1.4)^2} - 40 & \text{if} \quad \tilde{x} \notin \mathcal{G} \\ -300 & \text{if} \quad \tilde{x} \in \mathcal{G} \end{cases}$$

F.3 Safety HalfCheetah

The Safety HalfCheetah environment is taken from Safety Mujoco, we add static obstacles in the environment to increase the difficulty of the task. We use x_{front} to denote the x-axis position of the front foot of Halfcheetah, y_{front} to be the y-axis position of the back foot of Halfcheetah, x_{back} to denote the x-axis position of the back foot of Halfcheetah, y_{back} to be the y-axis position of the back foot of Halfcheetah, x_{head} to denote the x-axis position of the head of Halfcheetah, y_{head} to be the y-axis position of the head of Halfcheetah. Then the goal region can be described as

$$\mathcal{G} := \{(x_{head}, y_{head}) \mid \|[x_{head}, y_{head}] - [5.0, 0.0]\| < 0.2\}$$

The unsafe set is described as

$$\begin{aligned} \mathcal{F} :=& \{(x_{front}, y_{front}) \mid y_{front} < 0.25, 2.45 < x_{front} < 2.55\} \\ & \cup \{(x_{back}, y_{back}) \mid y_{back} < 0.25, 2.45 < x_{back} < 2.55\} \end{aligned}$$

The cost function is described as

$$c(x_t, u_t, x_{t+1}) = \|u_t\|^2$$

g is given by

$$g(\tilde{x}) = \begin{cases} 100\sqrt{(x_{head}-2)^2 + (y_{head}-1.4)^2} - 20 & \text{if} \quad \tilde{x} \notin \mathcal{G} \\ -300 & \text{if} \quad \tilde{x} \in \mathcal{G} \end{cases}$$

F.4 FixedWing

FixedWing environment is taken from [58] and we follow the same design of $\mathcal{F}$ as [58]. We denote the x_{PE} as the eastward displacement of F16 with given state x. Then the goal region $\mathcal{G}$ is given by

$$\mathcal{G} := \{x \mid 1975 \le x_{PE} \le 2025\}$$

The cost c is given by

$$c(x_t, u_t, x_{t+1}) = 4\|u_t/[1, 25, 25, 25]\|^2$$

and g is given by

$$g(x) = \begin{cases} \frac{\|x_{PE} - 2000\| - 25}{4} & \text{if} \quad x \notin \mathcal{G} \\ -300 & \text{if} \quad x \in \mathcal{G} \end{cases}$$

F.5 Quadrotor in Wind Field

We take quadrotor dynamics from crazyflies and wind field environments in the urban area from [69]. The wind field will disturb the quadrotor with extra movement on both x-axis and y-axis. There are static building obstacles in the environment and we treat them as the unsafe region $\mathcal{F}$. The goal for the quadrotor is to reach the mid-point of the city. We divide the whole city into four sections and train single policy on each of the sections. We use $x \in [-30, 30]$ to denote the x-axis position of quadrotor, $y \in [-30, 30]$ to be the y-axis position of quadrotor.

$$\mathcal{G} := \{(x, y) \mid \|[x, y]\| \le 4\}$$

The cost c is given by

$$c(x_t, u_t, x_{t+1}) = \frac{\|u_t\|^2}{2}$$

g is given by

$$g(\tilde{x}) = \begin{cases} 10\sqrt{(x - x_{goal})^2 + 10(y - y_{goal})^2} - 40 & \text{if} \quad x \notin \mathcal{G} \\ -300 & \text{if} \quad x \in \mathcal{G} \end{cases}$$

F.6 PointGoal

The PointGoal environment is taken from Safety Gym [54] We implement `PointGoal` environments in Jax. In Safety Gym environment, we do not perform reward-shaping and use the original reward defined in Safety Gym environments. In this case, the distance reward is set also to be 20 in order to align* with C_{goal} and C_{cost}. Different from sampling outside the hazard region which is implemented in Safety Gym, we allow Point to be initialized within the hazard region. We use x to denote the x-axis position of Point, y to be the y-axis position of Point, x_{goal} to denote the x-axis position of Goal, and y_{goal} to denote the y-axis position of Goal. The goal region is given by

$$\mathcal{G} \coloneqq \{(x, y) \mid \|[x, y] - [x_{goal}, y_{goal}]\| \le 0.3\}$$

The cost c is given by

$$c(x_t, u_t, x_{t+1}) = \frac{\|u_t\|^2}{2}$$

g is given by

$$g(\tilde{x}) = \begin{cases} 100\sqrt{(x - x_{goal})^2 + (y - y_{goal})^2} - 30 & \text{if} \quad x \notin \mathcal{G} \\ -300 & \text{if} \quad x \in \mathcal{G} \end{cases}$$

F.7 Experiment Harware

We run all our experiments on a computer with CPU AMD Ryzen Threadripper 3970X 32-Core Processor and with 4 GPUs of RTX3090. It takes at most 4 hours to train on every environment.

G Additional Experiment Results

We put additional experiment results in this section.

G.1 Additional Cumulative Cost and Reach Rates

We show the cumulative cost and reach rates of the final converged policies for additional environments (`F16` and `Safety Hopper`) in Figure 6.

G.2 Visualization of learned policy for different z

To obtain better intuition for how the learned policy depends on z, we rollout the policy choices of z_0 in the `Pendulum` environment and visualize the results in Figure 7.

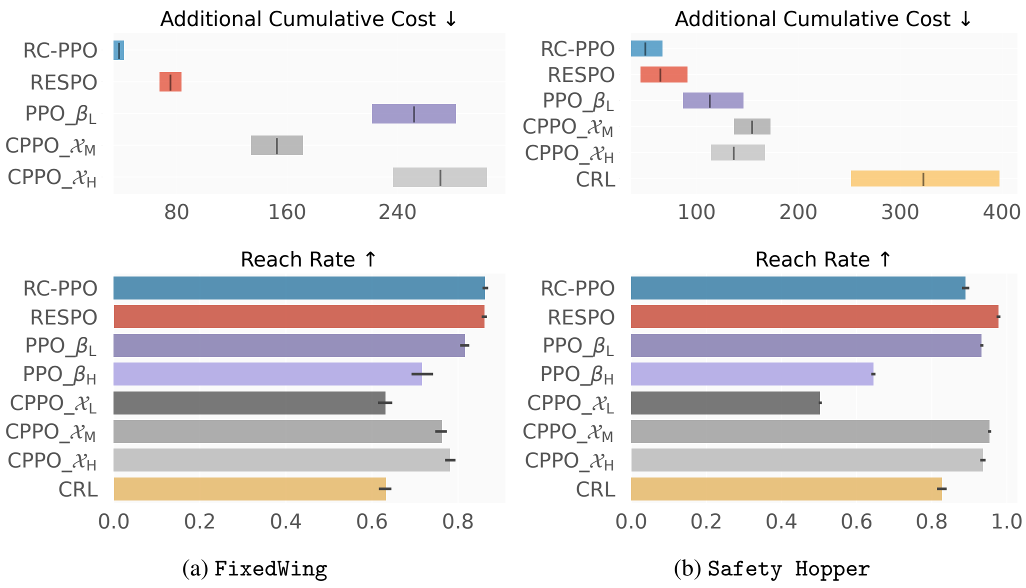
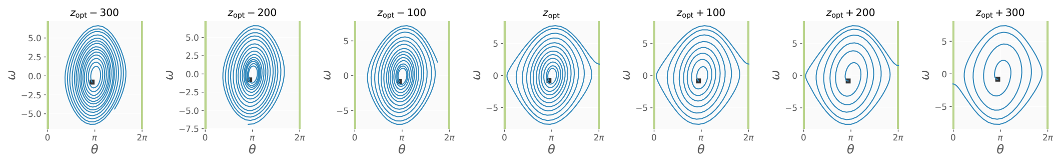


Figure 6: Cumulative cost and reach rates of the final converged policies.



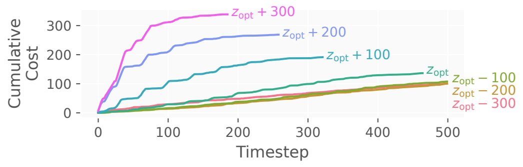


Figure 7: Learned RC-PPO policy for different z on `Pendulum`. For a smaller cost lower-bound z, cost minimization is prioritized at the expense of not reaching the goal. For a larger cost lower-bound z, the goal is reached using a large cumulative cost. Performing rootfinding to solve for the optimal z_{opt} *automatically* finds the policy that minimizes cumulative costs while still reaching the goal.

H Broader impact

Our proposed algorithm solves an important problem that is widely applicable to many different real-world tasks including robotics, autonomous driving, and drone delivery. Solving this brings us one step closer to more feasible deployment of these robots in real life. However, the proposed algorithm requires GPU training resources, which could contribute to increased energy usage.