Improving Stochastic Action-Constrained Reinforcement Learning via Truncated Distributions

Roland Stolz, Michael Eichelbeck, Matthias Althoff

Technical University of Munich
{roland.stolz, michael.eichelbeck, althoff}@tum.de

Abstract

In reinforcement learning (RL), it is often advantageous to consider additional constraints on the action space to ensure safety or action relevance. Existing work on such action-constrained RL faces challenges regarding effective policy updates, computational efficiency, and predictable runtime. Recent work proposes to use truncated normal distributions for stochastic policy gradient methods. However, the computation of key characteristics, such as the entropy, log-probability, and their gradients, becomes intractable under complex constraints. Hence, prior work approximates these using the non-truncated distributions, which severely degrades performance. We argue that accurate estimation of these characteristics is crucial in the action-constrained RL setting, and propose efficient numerical approximations for them. We also provide an efficient sampling strategy for truncated policy distributions and validate our approach on three benchmark environments, which demonstrate significant performance improvements when using accurate estimations.

1 Introduction

Reinforcement learning (RL) natively operates on a static action space from which the agent can choose an action in each time step. However, a dynamic restriction of the action space can be advantageous to exclude irrelevant actions (Stolz et al. 2024) or even necessary to guarantee safety (Krasowski et al. 2023). The field of action-constrained RL has developed various approaches to achieve this with zero constraint violations. Most methods handle constraints by projecting actions onto sets formed by the action constraints (Kasaura et al. 2023), which can lead to zero gradients, because multiple actions outside the corresponding set might be mapped to the same action (Lin et al. 2021; Kasaura et al. 2023). Some approaches address this issue by attempting to retain the gradient through learning a flow model (Brahmanage, Ling, and Kumar 2023, 2025), or using Franke-Wolfe optimization (Lin et al. 2021).

Other action-constrained RL methods use different mappings from actions outside the constraints to within the set, such as α-projection (Sanket et al. 2020), replacement with fail-safe actions (Krasowski et al. 2023), or radially contracting the action space (Kasaura et al. 2023; Stolz et al. 2024). One proposal by (Stolz et al. 2024) is to learn actions in a box-constrained latent space that is mapped to the constrained action space via a linear transformation, which is efficient, but restricted to constraints that can be formulated as zonotopes. The work by (Theile et al. 2024) uses a similar approach, but learns the mapping from the latent space to the action constraints. Another recent work relies on rejection sampling to obtain feasible actions (Hung, Sun, and Hsieh 2025). While a multi-objective algorithm is developed in that work to incentivize the agent to follow trajectories with large feasible action sets, rejection sampling cannot provide guarantees with regard to computation time, which we illustrate as part of our discussion in Sec. 4.3.

Instead of defining a mapping to constraint-satisfying actions, the study in (Stolz et al. 2024) proposes to directly truncate the policy distribution using the action constraints. However, this has two limitations. First, the policy update is approximated using the non-truncated distribution, because the metrics required by the RL objectives, such as entropy and log-probability, are generally intractable under the constrained distribution. Second, the sampling approach is based on a geometric random walk, which has high computational costs per sample and prevents using the reparameterization trick (Kingma and Welling 2014) for gradient estimation in stochastic RL.

This paper builds on the work employing truncated distributions for RL (Stolz et al. 2024) and tackles key limitations by developing more expressive policy updates and efficient sampling methods. In particular, our contributions are:

- Approximations of the intractable log-probability and entropy for truncated distributions applicable to convex, non-convex, and disjoint sets;
- An efficient, hybrid sampling algorithm for truncated distributions, merging rejection sampling and geometric random walks for improved performance;
- A novel application of the reparameterization trick to geometric random walks, enabling differentiable sampling.

The remainder of this study is organized as follows. After introducing preliminaries (Sec. 2), we formalize our problem statement (Sec. 3), followed by our proposed methods for leveraging truncated distributions in RL (Sec 4). Finally, we compare the developed mechanisms on three RL benchmarks (Sec. 5) and provide concluding remarks (Sec. 6).

2 Preliminaries

2.1 Action-Constrained Markov Decision Processes

We consider problems that can be modeled by a Markov decision processes, defined as a tuple $(\mathcal{S}, \mathcal{A}, T, r, \gamma)$ comprising the following components: an observable and continuous state space $\mathcal{S} \subseteq \mathbb{R}^m$, an action space $\mathcal{A} \subseteq \mathbb{R}^n$, a stationary state-transition distribution $T(s'|a, s)$ that characterizes the probability of transitioning to a subsequent state $s' \in \mathcal{S}$ given the current state $s \in \mathcal{S}$ and executed action $a \in \mathcal{A}$, a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and a discount factor γ for future rewards (Sutton and Barto 2018). Action-constrained RL further considers a state-dependent feasible action space $\mathcal{A}^s(s) \subseteq \mathcal{A}$. The goal is to learn a policy $\pi^s_\theta(a|s)$ parameterized by θ that maximizes the expected discounted return $\max_\theta \mathbb{E}_{\pi_\theta} \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$, while only selecting actions from the feasible action space $\mathcal{A}^s(s_t)$ at each time step t.

2.2 Stochastic Policy RL

Stochastic, on-policy algorithms, such as proximal policy optimization (PPO), learn the optimal policy $\pi^*_\theta(a|s)$ by updating the parameters with $\theta \leftarrow \theta + \beta \nabla_\theta J(\pi_\theta)$, according to the policy gradient (Sutton et al. 1999, Thm. 2)

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) A_{\pi_\theta}(a, s) \right], \tag{1}$$

where $A_{\pi_\theta}(a, s)$ denotes the advantage function, which quantifies the expected improvement in return from executing action a in state s relative to the expected performance under the current policy $\pi_\theta(a|s)$, and is typically approximated using a neural network.

As an alternative, the stochastic, off-policy algorithm soft actor-critic (SAC) (Haarnoja et al. 2018) incorporates entropy regularization into the optimization objective. SAC aims to maximize the expected cumulative return augmented with an entropy term

$$J(\pi_\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{T} \gamma^t \Big(r(s_t, a_t) + \alpha \mathcal{H}\big(\pi_\theta(\cdot|s_t)\big) \Big) \right], \tag{2}$$

where $\mathcal{H}(\pi_\theta(\cdot|s_t)) = -\mathbb{E}_{a \sim \pi_\theta}[\log \pi_\theta(a_t|s_t)]$ represents the policy entropy and α is a temperature parameter controlling the trade-off between exploration and exploitation. The optimal policy $\pi^*_\theta(a|s)$ is obtained via gradient descent on the parameters θ with the gradient

$$\begin{aligned} \nabla_\theta J(\pi_\theta) = \mathbb{E}_{s \sim \mathcal{D}} \big[& \nabla_\theta \log \pi_\theta(a|s) + \\ & \big(\alpha \nabla_a \log \pi_\theta(a|s) - \nabla_a Q_\phi(s, a) \big) \nabla_\theta a \big|_{a \sim \pi_\theta} \big], \end{aligned} \tag{3}$$

where $\mathcal{D}$ denotes the replay buffer containing experience tuples, and Q_ϕ is the soft Q-function approximated by critics with parameters ϕ. Evaluating this gradient requires backpropagating through $a \sim \pi_\theta$, which is achieved using the reparameterization trick (Kingma and Welling 2014).

2.3 Set Representations

A multidimensional interval $\mathcal{I} \subset \mathbb{R}^d$ is defined by lower and upper bounds $l, u \in \mathbb{R}^d$, such that

$$\mathcal{I} = [l, u] = \{x \in \mathbb{R}^d : l \leqslant x \leqslant u\}. \tag{4}$$

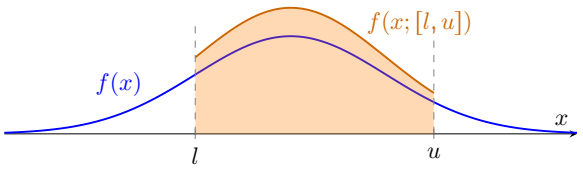


Figure 1: The PDF of the standard normal distribution $f(x)$ is truncated to the interval $[l, u]$ to obtain the truncated PDF $f(x; [l, u])$. The area under the curve is normalized to one.

A polytope $\mathcal{P} \subset \mathbb{R}^d$ can be represented as the intersection of m halfspaces and is denoted by

$$\mathcal{P} = \{x \in \mathbb{R}^d : Ax \leqslant b\}, \tag{5}$$

where $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$.

2.4 Truncated Distributions

We write a continuous probability density function (PDF) $f(x) \in \mathbb{R}^d$ truncated to a set $\mathcal{W}$ as

$$f(x; \mathcal{W}) = \frac{f(x) \cdot \phi(x; \mathcal{W})}{Z_{\mathcal{W}}}, \tag{6}$$

where $\phi(x; \mathcal{W})$ is the indicator function that returns 1 if $x \in \mathcal{W}$ and 0 otherwise, and $Z_{\mathcal{W}} = \int_{x \in \mathcal{W}} f(x) \mathrm{d}x$ is the normalizing constant. The function is generally intractable due to the integral in the denominator. However, for univariate $f(x)$, and when $\mathcal{W} = [l, u] \subset \mathbb{R}$, it can be written as (Johnson, Kotz, and Balakrishnan 1994, Eq. 13.133)

$$f(x; [l, u]) = \begin{cases} \frac{f(x)}{Z_{[l,u]}} & \text{if } l \leqslant x \leqslant u, \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

where $Z_{[l,u]} = F(u) - F(l)$, and $F(x)$ is the cumulative distribution function (CDF) of $f(x)$ (see Fig. 1). The entropy in general is (Shangari and Chen 2012, Eq. 2.1)

$$\begin{aligned} & \mathcal{H}\big(f(\cdot; [l, u])\big) = \\ & \quad - \frac{1}{Z_{[l,u]}} \int_l^u f(x) \log f(x) \mathrm{d}x + \log Z_{[l,u]}, \end{aligned} \tag{8}$$

which, for Gaussian distributions $\mathcal{N}(\mu, \sigma^2)$, has the closed from (Michalowicz, Nichols, and Bucholtz 2013, Sec. 4.26)

$$\begin{aligned} & \mathcal{H}\big(f(\cdot; [l, u])\big) = \\ & \quad \frac{1}{2} \log(2\pi e \sigma^2) + \log(Z_{[l,u]}) - \frac{u' \, \varphi(u') - l' \, \varphi(l')}{2 Z_{[l,u]}}, \end{aligned} \tag{9}$$

where $u' = \frac{u - \mu}{\sigma}$, $l' = \frac{l - \mu}{\sigma}$, and $\varphi(x)$ is the PDF of the standard normal distribution. The mode of a truncated distribution (i.e., the point at which its PDF attains the maximum value) for a Gaussian $f(x)$ is

$$\arg\max_x f(x; [l, u]) = \begin{cases} l, & \text{if } \mu \leqslant l, \\ u, & \text{if } \mu \geqslant u, \\ \mu, & \text{otherwise.} \end{cases} \tag{10}$$

2.5 Sampling

We can obtain a sample q from a univariate truncated distribution $f(\cdot;[l,u])$ by first sampling a uniform random variable $y \sim \mathcal{U}(0,1)$ and then applying the inverse CDF (Burkardt 2023, Sec. 3.4):

$$q = F^{-1}\Big(F(l) + y\big(F(u) - F(l)\big)\Big). \tag{11}$$

For a multi-variate truncated distribution, we can obtain samples analytically only in special cases, which we explore in Sec. 4.1. In the general case, we must use numerical alternatives, the most straightforward of which is rejection sampling. There, samples are drawn as $q \sim f(\cdot)$, and accepted as samples from $f(\cdot;\mathcal{W})$, if $q \in \mathcal{W}$. As an alternative, (Stolz et al. 2024) propose to use Markov chain Monte Carlo sampling to draw from $f(\cdot;\mathcal{W})$, specifically, the random direction hit-and-run (RDHR) algorithm (Zabinsky and Smith 2013; Chalkis and Fisikopoulos 2021).

3 Problem Statement

We consider the following setting for action-constrained RL in this work. The set of action constraints $\mathcal{A}^s(s)$ is given for each state $s \in \mathcal{S}$. We directly truncate the original policy distribution $\pi_\theta(\cdot|s)$ to obtain the PDF of the satisfying policy distribution $\pi^s_\theta(\cdot|s)$ as (Stolz et al. 2024)

$$\pi^s_\theta(a|s) = \frac{\pi_\theta(a|s)\phi(a;\mathcal{A}^s)}{\int_{\tilde{a}\in\mathcal{A}^s(s)} \pi_\theta(\tilde{a}|s)\mathrm{d}\tilde{a}} = \frac{\pi_\theta(a|s)\phi(a;\mathcal{A}^s)}{Z_{\mathcal{A}^s}(\theta)}, \tag{12}$$

where $\phi(a;\mathcal{A}^s)$ is the indicator function that returns 1 if $a \in \mathcal{A}^s(s)$ and 0 otherwise. Since we want to use $\pi^s_\theta(\cdot|s)$ for stochastic policy gradient algorithms, we need to solve the following problems.

Problem 1. *Compute a differentiable expression for the log-probability of an action* $\log \pi^s_\theta(a|s)$ *with the gradient* $\nabla_\theta \log \pi^s_\theta(a|s) = \nabla_\theta \log \pi_\theta(a|s) - \nabla_\theta \int_{\tilde{a}\in\mathcal{A}^s(s)} \pi_\theta(\tilde{a}|s)\mathrm{d}\tilde{a}$.

Problem 2. *Compute the entropy* $\mathcal{H}\big(\pi^s_\theta(\cdot|s)\big)$.

Problem 3. *Compute the mode of the truncated policy distribution* $\arg\max_{a\in\mathcal{A}^s} \pi^s_\theta(a|s)$ *to evaluate the RL agent.*

Problem 4. *Sample efficiently from* $\pi^s_\theta(\cdot|s)$.

The difficulty of each problem depends on the nature of the set $\mathcal{A}^s$ and $\pi_\theta(\cdot|s)$. There are analytical solutions for a special case, which we detail in Sec. 4.1. For more general cases, we propose approximations in Sec. 4.2, and Sec. 4.3 proposes strategies for sampling from $\pi^s_\theta(\cdot|s)$.

4 Using Truncated Distributions in RL

4.1 Analytical Solution

When the policy distribution can be fully factorized into each action dimension i, i.e., $\pi_\theta(a|s) = \prod_{i=1}^n \pi_\theta^{(i)}(a_i|s)$, and $\mathcal{A}^s$ is an interval $[l,u]$, the truncated policy distribution can also be factorized as

$$\pi^s_\theta(a|s) = \prod_{i=1}^n \pi_\theta^{s\,(i)}(a_i|s) = \prod_{i=1}^n f(a_i;[l_i,u_i]), \tag{13}$$

where the i-th marginal distribution $\pi_\theta^{s\,(i)}(a_i|s)$ is a univariate, truncated distribution as defined in (7). Using the independent dimensions, we can define the entropy from the sum of the marginals (Cover and Thomas 2005, Cor. 8.6.2) as

$$\mathcal{H}\big(\pi^s_\theta(\cdot|s)\big) = \sum_{i=1}^n \mathcal{H}\big(\pi_\theta^{s\,(i)}(\cdot|s)\big). \tag{14}$$

When the original policy distribution $\pi_\theta(\cdot|s)$ is Gaussian, which is the case in most stochastic policy gradient RL algorithms, such as PPO (Schulman et al. 2017) or SAC (Haarnoja et al. 2018), we can use (9) for the marginal entropies in (14) to receive a closed-form expression for the entropy of the truncated policy distribution. Finally, samples from the truncated distribution $\pi^s_\theta(\cdot|s)$ can be obtained by using inverse transform sampling (11) with each marginal distribution $\pi_\theta^{s\,(i)}(\cdot|s)$ independently.

4.2 Approximations

To accurately compute the log-probability $\log \pi^s_\theta(a|s)$ in (12), we need to evaluate the normalizing constant $Z_{\mathcal{A}^s}(\theta) = \int_{\tilde{a}\in\mathcal{A}^s(s)} \pi_\theta(\tilde{a}|s)\mathrm{d}\tilde{a}$. The integral is generally intractable, but it can be approximated well by using numerical integration methods, such as Monte Carlo integration (Robert and Casella 2004) or cubature methods (Genz and Cools 2003). However, the first requires many samples to achieve a low error, and for the second, the number of evaluations of the functions scales exponentially with the dimension (Genz and Cools 2003). Also, both methods make it difficult to back-propagate through the estimate of $Z_{\mathcal{A}^s}(\theta)$, hence the gradient would have to be approximated as $\nabla_\theta \log \pi^s_\theta(a|s) \approx \nabla_\theta \pi_\theta(a|s)$. Therefore, we propose different methods for approximating the normalizing constant, which vary depending on the nature of the set $\mathcal{A}^s$. We first present methods for simple convex sets, followed by non-convex sets.

Convex sets In order to utilize the analytical solutions in Sec. 4.1 for general convex sets, we propose to approximate the convex set $\mathcal{A}^s$ as an interval $\mathcal{I}$, and use the metrics of the policy distribution truncated to $\mathcal{I}$ (which we denote as $\pi^s_\theta(\cdot|s)_\mathcal{I}$) as approximations for $\pi^s_\theta(\cdot|s)$:

$$\log \pi^s_\theta(a|s) \approx \log \pi_\theta(a|s) - \log Z_\mathcal{I}(\theta), \tag{15}$$

$$\mathcal{H}(\pi^s_\theta(\cdot|s)) \approx \mathcal{H}(\pi_\theta(\cdot|s)_\mathcal{I}). \tag{16}$$

We obtain the inner interval approximation $\mathcal{I}_{\text{inner}}$ of a convex set $\mathcal{S}$ by maximizing the geometric mean of its diameter:

$$\begin{aligned} \max \quad & \Big(\prod_{i=1}^d (u_i - l_i)\Big)^{\frac{1}{d}} \\ \text{subject to} \quad & \mathcal{I}_{\text{inner}} = [l,u] \subset \mathcal{S}. \end{aligned} \tag{17}$$

The outer approximation is the tightest interval enclosed of $\mathcal{A}^s$, which can generally be obtained through the support functions in direction of the standard basis vectors (Althoff and Frehse 2016, Eq. 1), or specifically for zonotopes and polytopes as in (Althoff 2010, Prop. 2.2, Prop. 2.3).

Since the intervals are nested such that $\mathcal{I}_{\text{inner}} \subseteq \mathcal{A}^s \subseteq \mathcal{I}_{\text{outer}}$, we can establish the general bounds $\square_{\mathcal{I}_{\text{inner}}} \leqslant \square \leqslant$

$\square_{\mathcal{I}_{\text{outer}}}$ for the normalizing constant $\square = Z_{\mathcal{A}^s}(\theta)$ and for the entropy $\square = \mathcal{H}(\pi^s_\theta(\cdot|s))$. This holds for the former, because the integral over a nonnegative function is monotonically increasing when enlarging the integration domain. For the latter, the entropy of a truncated distribution $f(x;[l,u])$ is shown to be an increasing function if the CDF $F(x)$ is log-concave, which is the case for Gaussian distributions (Shangari and Chen 2012, Thm. 2.3). These bounds enable us to interpolate the metric $\square$ between the inner and outer interval to achieve a tighter approximation than using either bound alone. We interpolate with

$$\square_{\mathcal{I}_{\text{combined}}} = \left(1 - \frac{1}{2^d}\right) \square_{\mathcal{I}_{\text{inner}}} + \frac{1}{2^d} \square_{\mathcal{I}_{\text{outer}}}, \tag{18}$$

because the volume of $\mathcal{A}^s$ compared to the outer interval approximation decreases exponentially with the dimension d (Matoušek 2002, Thm 13.2.1).

Finally, when $\mathcal{A}^s$ is convex and $\pi^s_\theta(\cdot|s)$ is a factorized Gaussian distribution, the mode is obtained by the action in $\mathcal{A}^s$, which minimizes the Mahalanobis distance to the mean:

$$\begin{aligned} \arg\min_a \quad & (a-\mu)^\top \operatorname{diag}(\sigma^{-2})(a-\mu) \\ \text{subject to} \quad & a \in \mathcal{A}^s. \end{aligned} \tag{19}$$

Non-convex and disjoint sets Under the very weak assumption that $\mathcal{A}^s$ is a measurable set, we can approximate it to arbitrary accuracy with a finite union of intervals, though the complexity increases exponentially with the dimension (Munkres 1991, Ch. 3). Using this, we can approximate $\pi^s_\theta(\cdot|s)$ for any bounded $\mathcal{A}^s$ by truncating the original policy distribution to the union of k intervals $\mathcal{U}_\mathcal{I} = \bigcup_{i=1}^k \mathcal{I}^{(i)}$.

Proposition 1. *Given the distribution truncated to the i-th interval $f(x;\mathcal{I}^{(i)})$, the PDF of the distribution $f(x)$ truncated to the union of k non-overlapping intervals $\mathcal{U}_\mathcal{I}$ is*

$$f(x;\mathcal{U}_\mathcal{I}) = \frac{f(x)\cdot\phi(x;\mathcal{U}_\mathcal{I})}{Z_\mathcal{U}} = \sum_{i=1}^k w_i f(x;\mathcal{I}^{(i)}), \tag{20}$$

where $Z_\mathcal{U}$ is the normalizing constant of $f(x;\mathcal{U}_\mathcal{I})$, and $w_i = \frac{Z_{\mathcal{I}^{(i)}}}{Z_\mathcal{U}}$ is the relative probability mass of the i-th interval.

Proof. The result is proven in Appendix A. □

Proposition 2. *The entropy of a PDF $f(x;\mathcal{U}_\mathcal{I})$ truncated to the union of k non-overlapping intervals $\mathcal{U}_\mathcal{I}$ is*

$$\mathcal{H}(f(x;\mathcal{U}_\mathcal{I})) = -\sum_{i=1}^k w_i \log w_i + \sum_{i=1}^k \mathcal{H}(f(x;\mathcal{I}^{(i)})). \tag{21}$$

Proof. The result is proven in Appendix B. □

The accuracy of the approximation $\pi^s_\theta(a|s) \approx \pi_\theta(a|s)_{\mathcal{U}_\mathcal{I}}$ and $\mathcal{H}(\pi^s_\theta(\cdot|s)) \approx \mathcal{H}(\pi^s_\theta(\cdot|s)_{\mathcal{U}_\mathcal{I}})$ depends on the method used for approximating $\mathcal{A}^s$ with $\mathcal{U}_\mathcal{I}$. A good approximation can, for instance, be obtained with the approach from (Brimkov, Andres, and Barneva 2000).

The mode $\arg\max_{a\in\mathcal{A}^s} \pi^s_\theta(a|s)$ can be obtained using (19), although the non-convex set $\mathcal{A}^s$ makes it a non-convex optimization problem, which is generally NP-hard (Murty and Kabadi 1987). If $\mathcal{A}^s$ can be inner-approximated by a convex set, an under-approximation of the mode is obtained by solving (19) for the inner-approximating set.

4.3 Sampling

Efficient sample generation Rejection sampling as described in Sec. 2.5 is often efficient, but suffers from the major drawback of a potentially very low acceptance rate when $Z_{\mathcal{A}^s}(\theta)$ is small; in extreme cases, the sampling process can even become stuck. At the same time, RDHR exhibits high computational costs per sample, but guaranteed convergence in polynomial time $\mathcal{O}(n^3)$ for well-conditioned sets (i.e., the ratio of the radii of its minimum enclosing ball and maximum contained ball is bounded by $\mathcal{O}(\sqrt{n})$) (Lovász and Vempala 2006).

To combine the strengths of both approaches, we propose to first execute rejection sampling for a maximum of M attempts, and switch to the RDHR algorithm when no sample has been accepted. This leverages the potentially fast rejection sampling, while ensuring sample generation via RDHR in $\mathcal{O}(n^3)$ even when $Z_{\mathcal{A}^s}(\theta)$ is very small.

Differentiating through sampling In SAC, the policy gradient in (3) contains $\nabla_a Q_\phi(s,a)\nabla_\theta a|_{a\sim\pi_\theta}$, which requires differentiating through $a \sim \pi^s_\theta(\cdot|s)$ (Haarnoja et al. 2018). To achieve this, the reparameterization trick (Kingma and Welling 2014) is employed, which expresses the sample as a differentiable function of an independent random variable $\varepsilon \in \mathbb{R}^d$ as $a = f_\theta(\varepsilon, s)$. This allows us to write the gradient as $\nabla_a Q_\phi\big(s, f_\theta(\varepsilon,s)\big)\nabla_\theta f_\theta(\varepsilon,s)$. More specifically, for Gaussian distributions $\mathcal{N}(\mu,\Sigma)$, where $\theta = \{\mu, L\}$ and $\Sigma = LL^T$, $\varepsilon \sim \mathcal{N}(0,I)$ is sampled from the standard Gaussian distribution and transformed as $a = \mu + L\,\varepsilon$.

Accordingly, rejection sampling can simply be made differentiable by proposing the samples with $a = f_\theta(\varepsilon, s)$. Also, the inverse transform sampling proposed in Sec. 4.1 is directly differentiable (Kingma and Welling 2014). However, this is not the case for RDHR sampling, hence we derive the differentiable reparameterization for RDHR.

Proposition 3. *Let $\pi^s_\theta(\cdot|s)$ be a multivariate Gaussian distribution $\mathcal{N}_{\mathcal{A}^s}(\mu,\Sigma)$ truncated to the convex set $\mathcal{A}^s$, with the Cholesky decomposition $\Sigma = LL^T$. Samples $a^s \sim \pi^s_\theta(\cdot|s)$ can be obtained using $a^s = f_\theta(\tilde{\varepsilon}, s) = \mu + L\tilde{\varepsilon}$, where the random variable $\tilde{\varepsilon}$ is obtained by sampling from the standard Gaussian distribution $\mathcal{N}_{\tilde{\mathcal{A}}^s}(0,I)$ truncated to the set $\tilde{\mathcal{A}}^s = L^{-1}\big(\mathcal{A}^s \oplus (-\mu)\big)$. The operator $\oplus$ denotes the Minkowski sum.*

Proof. The result is proven in Appendix C and illustrated in Fig. 2. □

Sampling from non-convex sets When $\mathcal{A}^s$ is the union of disjoint convex sets, we draw samples from $\pi^s_\theta(\cdot|s)$ by first selecting a set $\mathcal{A}^{s(i)}$ with a probability proportional to its relative probability mass w_i (see Prop. 1), then sampling a^s from the chosen convex set with a method from Sec. 4.3. For general non-convex $\mathcal{A}^s$, geometric random walks are not applicable, leaving rejection sampling as the primary option. In this case, it is advisable to still cap the number of rejection attempts at M, and, if possible, resort to a fallback action guaranteed to lie in $\mathcal{A}^s$, although this introduces a bias.

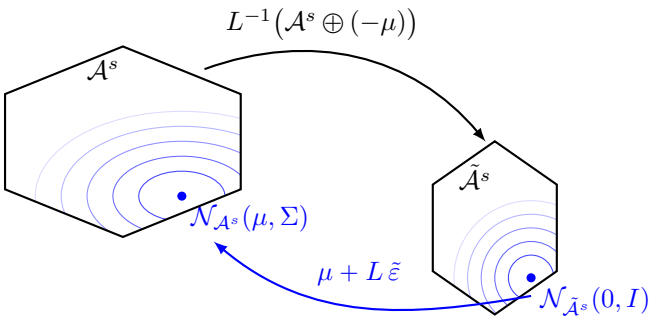


Figure 2: The reparameterization trick: We sample the independent random variable $\tilde{\varepsilon}$ from $\mathcal{N}_{\tilde{\mathcal{A}}^s}(0,1)$ truncated to $\tilde{\mathcal{A}}^s$, and then apply the affine transformation $a^s = \mu + L\,\tilde{\varepsilon}$ to obtain samples from $\pi^s_\theta(\cdot|s) = \mathcal{N}_{\mathcal{A}^s}(\mu, \Sigma)$, where $\Sigma = LL^T$.

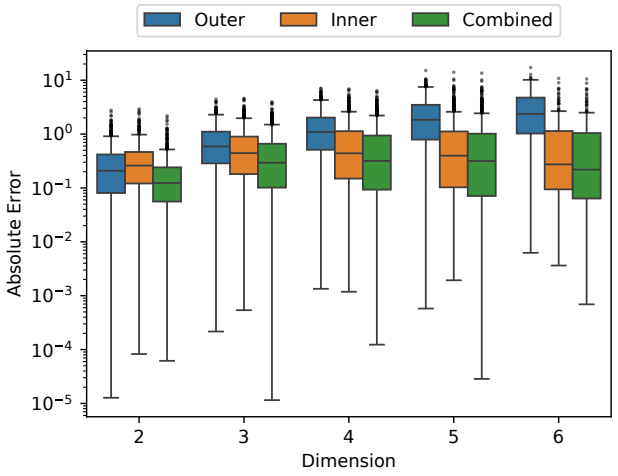


Figure 3: Errors for the integral approximation using the outer, inner, and combined approximation of the polytope.

5 Experiments

5.1 Numerical Experiments

We generate a dataset of 6000 factorized Gaussian distributions truncated to polytopes, with 1000 random instances for each dimension $d = 2, \ldots, 6$. The samples are generated to achieve a balanced distribution of probability mass inside $\mathcal{A}^s$. Each polytope $\mathcal{P}$ is formed by intersecting the unit box $[-1, 1]^d$ with $n_P \sim \mathcal{U}(d, 4d)$ random halfspaces. Each halfspace uses a random unit vector $a_j \in \mathbb{R}^d$ and offset $b_j = a_j^\top x_0 + \delta_j$, where $x_0 \sim \mathcal{U}(-0.8, 0.8)^d$ and $\delta_j \sim \mathcal{U}(0.1, 1.0)$. For the Gaussian distribution, we set $\mu = x_1 + c$, with c as the center of $\mathcal{P}$, and obtain $x_1 \sim \mathcal{U}(0, 0.5)^d$, and $\sigma \sim \mathcal{U}(0.1, 1)^d$.

Integral approximation We compare the accuracy of our interval-based approximations for the integral $\int_{\tilde{a} \in \mathcal{A}^s(s)} \pi_\theta(\tilde{a}|s) \mathrm{d}\tilde{a}$ as defined in Sec. 4.2. The ground truth is estimated with a precision of $1e-5$ using cubature methods (Genz and Cools 2003) (we use the implementation from the package PySimplicialCubature). Fig. 3 shows the absolute errors for $d = 2, \ldots, 6$, which are normalized to the average integral size in each dimension. The outer approximation shows the highest error for $d > 2$, which is expected, since volume in higher dimensions is concentrated near the boundary of geometries (Matoušek 2002). The combined method exhibits the lowest median and quartile range errors in every dimension, as expected from the derived bounds for the integral in Sec. 4.2.

Sampling We assess sampling efficiency (see Sec. 4.3) by averaging the time to draw 10 samples per truncated distribution in the dataset. Fig. 4 presents the sampling times for RDHR (geometric random walk), rejection sampling, and our combined sampling strategy with rejection limit $M = 100$. As expected, RDHR shows significantly higher median sampling time and lower variance compared to the other methods, and pure rejection sampling exhibits large outliers towards high sampling times. The combined method effectively removes these outliers, as intended, and reduces the median sampling time in lower dimensions.

5.2 Benchmarks

We evaluate our methods using environments with action constraints that guarantee safety. Computing the safe action sets $\mathcal{A}^s(s)$ for these environments requires concepts from reachability analysis that are beyond the scope of this paper; the details are provided in Appendix D. All environments are implemented in Gymnasium (Towers et al. 2024), and we use the implementations of SAC and PPO from Tianshou (Weng et al. 2022).

Seeker Our first two environments are based on the Seeker Reach-Avoid environment from (Stolz et al. 2024, Sec. 5.1.1, A.3.3). This environment represents a prototypical 2D reach-avoid task in which an agent must reach a goal area while avoiding a single obstacle. We generalize this environment to 2D (*Seeker-2D*) and 3D (*Seeker-3D*) with multiple obstacles and compute feasible action sets as polytopes and intervals instead of zonotopes (see Appendix D.2). We use the slightly modified reward function:

$$r(a, s) = \begin{cases} 100, & \text{if the goal is reached} \\ -100, & \text{if a collision occurs} \\ l_{\text{prev}} - l_{\text{curr}} - 1, & \text{otherwise,} \end{cases} \quad (22)$$

where $l = \|p_{\text{agent}} - p_{\text{goal}}\|_2$, and l_{prev} and l_{curr} are the distances to the goal in the previous and current time step, respectively.

Quadrotor Our third environment (*Quadrotor*) is a 2D quadrotor stabilization task from (Stolz et al. 2024, Sec. 5.1.2, A.3) in which the agent must stabilize a quadrotor at a goal state s^* while compensating noise. The feasible action sets are computed based on the system dynamics and a desired state set, as detailed in Appendix D.1. We use the slightly modified reward function

$$r(a, s) = \exp\left(-\|s - s^*\|_2 - 0.005 \sum_{i=1}^{2} a_{i,\text{c}}\right) - 1, \quad (23)$$

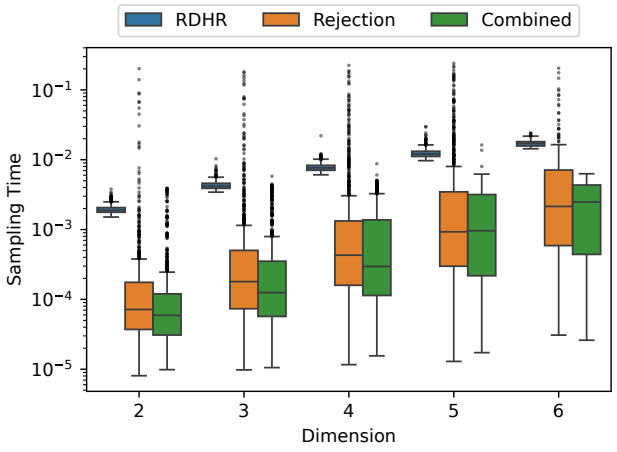


Figure 4: Comparison of the sampling times using the RDHR, rejection sampling and the combined sampling method with rejection limit $M = 100$.

where $a_{i,\mathrm{c}} = \frac{a_i - a_{\mathrm{i,min}}}{a_{\mathrm{i,range}}}$ is the normalized action cost, and $a_{\mathrm{i,min}}$ is the minimal action, and $a_{\mathrm{i,range}}$ is the absolute range of actions in dimension i.

Benchmark results We optimize hyperparameters for the base RL algorithms SAC and PPO, then use these for training (10 runs per algorithm) with truncated distributions, considering both interval and polytope representations of $\mathcal{A}^s$. Our hypothesis is that accurately estimating $\log \pi^s_\theta(\cdot|s)$ and $\mathcal{H}\big(\pi^s_\theta(\cdot|s)\big)$ improves performance over using the original metrics from $\pi_\theta(\cdot|s)$, which we test with the following setup: For polytopes, we compare policies using original metrics (*Og-Poly*) to our outer, inner, and combined approximations (*App-Poly-Out*, *App-Poly-Inn*, *App-Poly-Comb*; see Sec.4.2). For intervals, we compare exact analytic metrics (*Exact-Int*; see Sec.4.1) to original values (*Og-Int*). Fig. 5 shows mean episode returns and 95% confidence intervals for the three environments.

In *Seeker-2D*, both SAC and PPO, perform best with *App-Poly-Out*, followed by the inner and combined approximations. *Exact-Int* also achieves high returns, while policies using the original values both struggle to learn the task.

The difference between *App-Poly-Out*, and the *App-Poly-Inn* and *App-Poly-Comb* is substantially larger in the *Seeker-3D* environments. The overall returns are also lower than in the 2D case, suggesting significantly higher task complexity. Again, both algorithms fail to learn using the original values, while SAC with *App-Poly-Out* achieves the highest returns.

On the *Quadrotor* environment with PPO, *Exact-Int* achieves the highest returns, whereas *Og-Int* fails again after initial progress. However, *Og-Poly* achieves results that are slightly better than the approximations. With SAC, *Exact-Int* and *App-Poly-Inn* perform best. *Og-Poly* learns initially, but fails to improve further, and *Og-Int* does not learn at all.

5.3 Discussion and Limitations

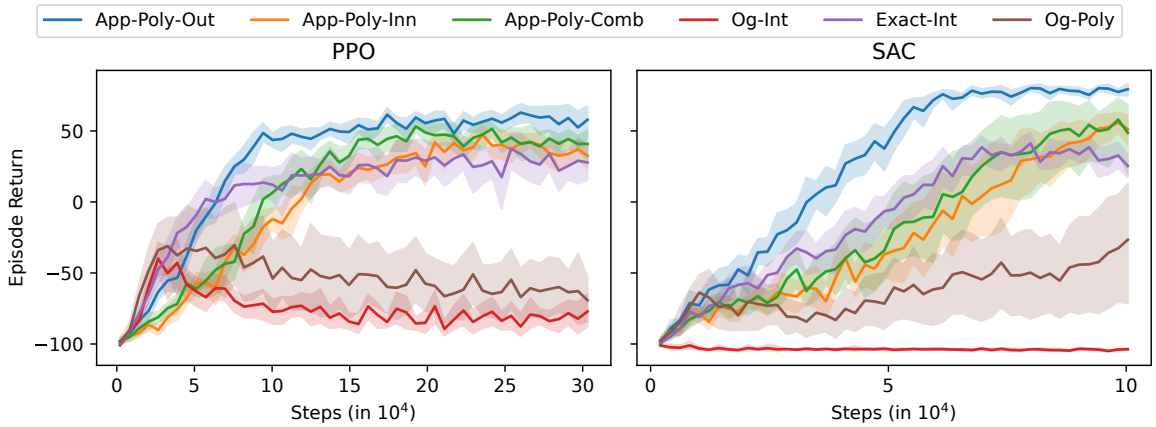
Runtime Three factors dominate the training time of the algorithms: 1) Computing $\mathcal{A}^s$ (outside the scope of this work), 2) sampling from $\pi^s_\theta(\cdot|s)$, and 3) estimating the metrics of $\pi^s_\theta(\cdot|s)$, which is dominated by estimating the normalizing constant $Z_{\mathcal{A}^s}(\theta)$.

The isolated sampling experiment shows that our combined sampling method is faster than pure rejection sampling and RDHR, making it the recommended approach when $\mathcal{A}^s$ is convex. The overall sampling time depends on the set representation of $\mathcal{A}^s$ and the dimensionality of the action space. The containment check in rejection sampling for polytopes with m halfspaces requires evaluating the halfspace condition m times, and for zonotopes requires solving a linear program (Kulmburg and Althoff 2021). Geometric random walks compute two boundary points in each step, the complexity of which also depends on the representation of $\mathcal{A}^s$. Additionally, the sampling time depends on the usual probability mass inside $\mathcal{A}^s$ during training. When it is low, the algorithm often falls back to geometric random walks, which can result in slow training.

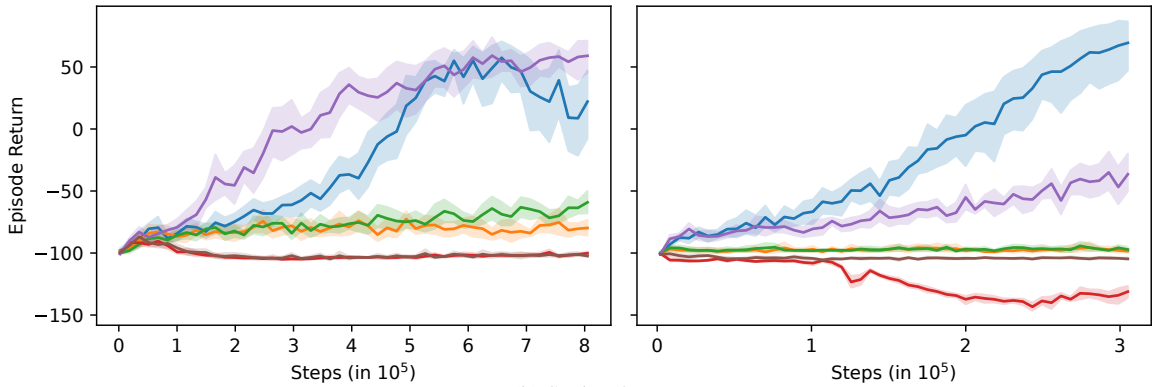
Regarding 3), our estimates of the metrics of $\pi^s_\theta(\cdot|s)$ introduce overhead compared to simply using the non-truncated distribution $\pi_\theta(\cdot|s)$ as in (Stolz et al. 2024). The overhead again depends on the representation of $\mathcal{A}^s$. For intervals, the overhead using the analytic solutions from Sec. 4.1 is minimal. However, the approximations for general convex sets introduce more overhead. For instance, with polytopes in halfspace representation, the outer approximation with support functions requires solving $2d$ linear programs, while the inner approximation in (17) solves one second-order cone program with polynomial complexity (Boyd and Vandenberghe 2004, Sec. 2.5).

RL performance The benchmark results in Fig. 3 clearly indicate that using policy metrics from the original distribution $\pi_\theta(\cdot|s)$ is ineffective. The strong performance of *Exact-Int* (with analytic solutions) compared to *Og-Int* highlights the benefit of accurate estimates for $\log \pi^s_\theta(a|s)$ and $\mathcal{H}\big(\pi^s_\theta(\cdot|s)\big)$. Two effects explain the bad performance when using $\pi^s_\theta(a|s) \approx \pi_\theta(a|s)$ (as in *Og-Int* and *Og-Poly* do). First, the gradient $\nabla_\theta \pi^s_\theta(a|s) \approx \nabla_\theta \pi_\theta(a|s)$ is now independent of the normalizing constant $Z_{\mathcal{A}^s}(\theta)$, hence it might point in the wrong direction. Second, through truncation, we often sample actions with very low probability under the original, non-truncated distribution, which can result in a small policy gradient, leading to inefficient learning.

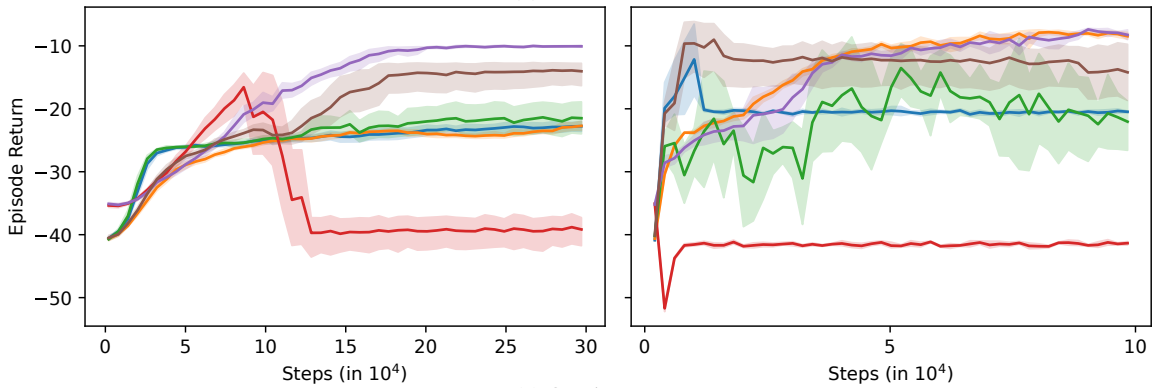
The outer approximation of the polytope with an interval exhibits higher errors compared to the inner or combined approximation in the numerical comparison (see Fig. 3). Despite this, the *App-Poly-Out* often achieves higher returns than the other two in the RL training. We attribute this to small $\mathcal{A}^s$ often causing numerical instabilities in the estimation of $Z_{\mathcal{A}^s}(\theta)$ due to very low probability mass, making the slight positive bias of the outer approximation advantageous. Future work should further investigate the potential benefit of this bias, and evaluate more precise approximations of $\mathcal{A}^s$ using the union of intervals (as proposed in Sec. 4.2) numerically, and in RL benchmarks.



(a) Seeker-2D



(b) Seeker-3D



(c) Quadrotor

Figure 5: The mean returns and 95% confidence intervals during RL training. *App-Poly-Out*, *App-Poly-Inn*, and *App-Poly-Comb*, refer to the outer, inner, and combined approximations, while *Og-Int* and *Og-Poly* refer to the interval and polytope policies using the original metrics of $\pi_\theta(\cdot|s)$. *Exact-Int* is the interval policy that computes exact analytic metrics.

6 Conclusion

Achieving effective policy updates, computational efficiency, and a predictable runtime are key challenges in action-constrained RL. Previous work proposes truncated distributions, but uses inaccurate metrics and inefficient sampling. We improve this by developing more accurate numerical estimates for the truncated policy metrics as well as an efficient hybrid sampling approach, and derive a differentiable version of the sampling mechanism, which enables the use of truncated distributions with SAC. The experiments demonstrate that our approach leads to a significantly improved performance across three benchmark environments, underlining the importance of estimating accurate policy metrics in action-constrained RL. We believe that accurate estimations of truncated distributions enable applications beyond those commonly considered in the action-constrained RL literature. Future work could, for example, investigate its use in physics-informed algorithms, curriculum learning, or other areas using probability distributions in the learning process, such as representation learning.

A Proof of Proposition 1

Since the k intervals $\mathcal{I}^{(i)}$ are non-overlapping, the indicator function for the union of intervals $\mathcal{U}_\mathcal{I}$ can be decomposed into the sum of the individual indicator functions:

$$\phi(x;\mathcal{U}_\mathcal{I}) = \sum_{i=1}^{k} \phi(x;\mathcal{I}^{(i)}).$$

We can substitute this into the expression of the distribution truncated to $\mathcal{U}_\mathcal{I}$ as

$$\begin{aligned} f(x;\mathcal{U}_\mathcal{I}) &\overset{(6)}{=} \frac{f(x)\sum_{i=1}^{k}\phi(x;\mathcal{I}^{(i)})}{Z_\mathcal{U}} \\ &= \sum_{i=1}^{k} \frac{f(x)\phi(x;\mathcal{I}^{(i)})}{Z_\mathcal{U}} = \sum_{i=1}^{k} \frac{Z_{\mathcal{I}^{(i)}} f(x)\phi(x;\mathcal{I}^{(i)})}{Z_{\mathcal{I}^{(i)}} Z_\mathcal{U}} \\ &\overset{(6)}{=} \sum_{i=1}^{k} \frac{Z_{\mathcal{I}^{(i)}}}{Z_\mathcal{U}} f(x;\mathcal{I}^{(i)}) = \sum_{i=1}^{k} w_i f(x;\mathcal{I}^{(i)}), \end{aligned}$$

where $w_i = \frac{Z_{\mathcal{I}^{(i)}}}{Z_\mathcal{U}}$. This concludes the proof.

B Proof of Proposition 2

Since the intervals $\mathcal{I}^{(i)}$ are disjoint, we have that

$$\int_{x\in\mathcal{U}_\mathcal{I}} \sum_{i=1}^{k} f(x;\mathcal{I}^{(i)})\mathrm{d}x = \sum_{i=1}^{k} \int_{x\in\mathcal{I}^{(i)}} f(x;\mathcal{I}^{(i)})\mathrm{d}x. \quad (24)$$

Using the general expression for the entropy (Shangari and Chen 2012, Eq. 1.1), we can write

$$\begin{aligned} \mathcal{H}\big(f(x;\mathcal{U}_\mathcal{I})\big) &= -\int_{x\in\mathcal{U}_\mathcal{I}} f(x;\mathcal{U}_\mathcal{I}) \log f(x;\mathcal{U}_\mathcal{I})\mathrm{d}x \\ &\overset{Prop.\,1,\,(20)}{=} -\int_{x\in\mathcal{I}^{(i)}} \sum_{i=1}^{k} w_i f(x;\mathcal{I}^{(i)}) \log\big(w_i f(x;\mathcal{I}^{(i)})\big)\mathrm{d}x \\ &\overset{(24)}{=} -\sum_{i=1}^{k} \int_{x\in\mathcal{I}^{(i)}} w_i f(x;\mathcal{I}^{(i)}) \log\Big(w_i f(x;\mathcal{I}^{(i)})\Big)\,\mathrm{d}x \\ &= -\sum_{i=1}^{k} w_i \int_{x\in\mathcal{I}^{(i)}} f(x;\mathcal{I}^{(i)})\big(\log w_i + \log f(x;\mathcal{I}^{(i)})\big)\mathrm{d}x \\ &= -\sum_{i=1}^{k} w_i \log w_i \underbrace{\int_{x\in\mathcal{I}^{(i)}} f(x;\mathcal{I}^{(i)})\mathrm{d}x}_{1} \\ &\quad -\sum_{i=1}^{k} \underbrace{\int_{x\in\mathcal{I}^{(i)}} f(x;\mathcal{I}^{(i)}) \log f(x;\mathcal{I}^{(i)})\mathrm{d}x}_{-\mathcal{H}\left(f(x;\mathcal{I}^{(i)})\right)} \\ &= -\sum_{i=1}^{k} w_i \log w_i + \sum_{i=1}^{k} \mathcal{H}\big(f(x;\mathcal{I}^{(i)})\big), \end{aligned}$$

which is equivalent to (21), thus concluding the proof.

C Proof of Proposition 3

For the proposition to be true, we need to show for the reparameterization $a^s = f(\tilde{\varepsilon}, s) = \mu + L\,\tilde{\varepsilon}$ that 1) applying it to the samples of the standard Gaussian distribution $\mathcal{N}(0, I)$ results in the Gaussian distribution $\mathcal{N}(\mu, \Sigma)$, with the Cholesky decomposition $\Sigma = LL^T$, and 2), applying it to all elements in $\tilde{\mathcal{A}}^s$ results in $\mathcal{A}^s$.

For 1), $f(\tilde{\varepsilon}, s)$ is simply an affine transformation of the standard Gaussian distribution $\mathcal{N}(0, I)$, which results in the Gaussian distribution (Tong 1990, Thm. 3.3.3)

$$\mathcal{N}(L0 + \mu, LIL^T) = \mathcal{N}(\mu, \Sigma). \quad (25)$$

For 2), we can write $\mathcal{A}^s = \{\mu + L\,\tilde{\varepsilon} \mid \tilde{\varepsilon} \in \tilde{\mathcal{A}}^s\}$ as

$$\begin{aligned} \mathcal{A}^s &= L\tilde{\mathcal{A}}^s \oplus \mu \\ \mathcal{A}^s \oplus (-\mu) &= L\tilde{\mathcal{A}}^s \\ L^{-1}\big(\mathcal{A}^s \oplus (-\mu)\big) &= \tilde{\mathcal{A}}^s, \end{aligned}$$

which shows that applying the transformation to all elements in $\tilde{\mathcal{A}}^s$ results in $\mathcal{A}^s$, thus concluding the proof.

D Computation of the Feasible Action Set

The computation of the feasible action set using reachability analysis follows a similar principle as detailed in (Stolz et al. 2024, Sec. A.3), with the modification of $\mathcal{A}^s$ being represented as a polytope instead of a zonotope. We also assume the existence of a robust control invariant set $\mathcal{S}^r$, which guarantees that there is always an action that keeps the system within its boundaries given $s_0 \in \mathcal{S}^r$. We summarize the concept and highlight the differences to the zonotope-based computation in the following sections. Interested readers are referred to the original source (Stolz et al. 2024).

D.1 Quadrotor

We introduce a zonotope as $\mathcal{Z} = \{c + G\beta \mid \|\beta\|_\infty \leqslant 1\} = \langle c, G\rangle_\mathcal{Z}$ with support function for direction $l \in \mathbb{R}^d$ as $\rho_\mathcal{Z}(\ell) = \ell^\top c + \|\ell^\top G\|_1$ (Althoff and Frehse 2016, Lemma 1). The discrete-time linearized system dynamics of the quadrotor from (Stolz et al. 2024, Eq. 30) are:

$$s_{k+1} = As_k + Ba_k + w'_k, \quad (26)$$

where $A \in \mathbb{R}^{m\times m}$ and $B \in \mathbb{R}^{m\times d}$ are the system matrices, and $w'_k \in \mathcal{W} = \langle c^\mathcal{W}, G^\mathcal{W}\rangle_\mathcal{Z} \subset \mathbb{R}^m$ is the disturbance at time step k.

Starting from states $\mathcal{S}_0$, and using the feasible inputs $\mathcal{A}^s$, the set of reachable states of the system at time step Δt is $\mathcal{R}_{\Delta t}(\mathcal{S}_0, \mathcal{A}^s)$. The robust control invariant set is a zonotope $\mathcal{S}^r = \langle c^\mathcal{S}, G^\mathcal{S}\rangle_\mathcal{Z} \subseteq \mathbb{R}^m$. To ensure $\mathcal{R}_{\Delta t}(\mathcal{S}_0, \mathcal{A}_\mathcal{P}) \subseteq \mathcal{S}^r$, we utilize support functions. For a given direction ℓ, the containment constraint becomes (Althoff and Frehse 2016, Eq. 1)

$$\ell^\top(As_k + Ba_k) \leqslant \rho_{\mathcal{S}^r}(\ell) - \rho_\mathcal{W}(\ell). \quad (27)$$

To account for $\mathcal{W}$ and $\mathcal{S}^r$, we apply the constraint for each direction defined by the generators of the zonotopes, i.e., $L = [G^\mathcal{W}, G^\mathcal{S}]^\top$, where $L \in \mathbb{R}^{n_G\times m}$ and n_G is the total number of generators in $\mathcal{W}$ and $\mathcal{S}^r$. Further, we represent

the action space $\mathcal{A}$ with the constraints $P_{\mathcal{A}} a \leqslant p_{\mathcal{A}}$. From this and (27), we can define the feasible action set as

$$\mathcal{A}^s_{\mathcal{P}}(s_k) = \left\{ a \in \mathbb{R}^2 : \begin{bmatrix} P_{\mathcal{A}} \\ L^T B \end{bmatrix} a \leqslant \begin{bmatrix} p_{\mathcal{A}} \\ \rho_1 - l_1^T A s_k \\ \vdots \\ \rho_{n_G} - l_{n_G}^T A s_k \end{bmatrix} \right\}, \tag{28}$$

where $\rho_i = \rho_{\mathcal{S}^r}(l_i) - \rho_{\mathcal{W}}(l_i)$ to simplify notation.

D.2 Seeker

We directly construct the feasible action as a polytope in halfspace representation $\mathcal{A}^s_{\mathcal{P}}$ from the optimization problem in (Stolz et al. 2024, Eq. 35). The following formulation is generalized to be viable in any dimension d, and work for multiple obstacles instead of just one. Due to the simple dynamics of the seeker environment, $s_{k+1} = s_k + a_k$, the boundary collision constraints $s_k + a \in [-10, 10]^d$ are

$$P_b = \begin{bmatrix} I_d \\ -I_d \end{bmatrix}, \quad p_b = \begin{bmatrix} 10 \cdot \mathbf{1}_d - s_k \\ 10 \cdot \mathbf{1}_d + s_k \end{bmatrix}, \tag{29}$$

where I_d is the identity matrix and $\mathbf{1}_d$ is a vector of ones in d dimensions, respectively. To avoid the m obstacles $\mathcal{O}_i$ with center o_i and radius r_i, the constraints for $i = 1, \ldots, m$ are

$$P_o = \begin{bmatrix} n_1^\top \\ \vdots \\ n_n^\top \end{bmatrix}, \quad p_o = \begin{bmatrix} b_1 - n_1^\top s_k \\ \vdots \\ b_n - n_n^\top s_k \end{bmatrix}, \tag{30}$$

where $n_i = \frac{o_i - s_k}{\|o_i - s_k\|}$ and $b_i = n_i^\top o_i - r_i$ following the halfspace approximation from (Stolz et al. 2024, App. A.3.3). The feasible action set is then

$$\mathcal{A}^s_{\mathcal{P}}(s_k) = \left\{ a \in \mathbb{R}^d : \begin{bmatrix} P_{\mathcal{A}} \\ P_b \\ P_o \end{bmatrix} a \leqslant \begin{bmatrix} p_{\mathcal{A}} \\ p_b \\ p_o \end{bmatrix} \right\}. \tag{31}$$

E Hyperparameters

The following ranges were used for Bayesian hyperparameter optimization with PPO:

- **batch_size:** $\{64, 128, 256\}$,
- **n_neurons:** $\{64, 128, 256\}$,
- **lr:** log-uniform, $[1 \times 10^{-5}, 1 \times 10^{-3}]$,
- **eps_clip:** uniform, $[0.1, 0.3]$,
- **repeat_per_collect:** integer uniform, $[1, 5]$,
- **ent_coef:** log-uniform, $[5 \times 10^{-4}, 5 \times 10^{-2}]$,
- **vf_coef:** uniform, $[0.25, 1.0]$,

and with SAC:

- **batch_size:** $\{64, 128, 256\}$,
- **n_neurons:** $\{64, 128, 256\}$,
- **actor_lr:** log-uniform, $[1 \times 10^{-5}, 1 \times 10^{-3}]$,
- **critic_lr:** log-uniform, $[1 \times 10^{-5}, 1 \times 10^{-3}]$,
- **update_per_step:** uniform, $[0.5, 1.5]$,
- **gamma:** uniform, $[0.95, 0.999]$,
- **tau:** log-uniform, $[1 \times 10^{-3}, 1 \times 10^{-1}]$,
- **alpha:** uniform, $[0.05, 0.5]$.

The hyperparameters obtained over 100 optimizations for the three environments are noted in Tab. 1 and Tab. 2.

Hyperparam.	Quadrotor	Seeker-2d	Seeker-3d
batch Size	64	128	64
ent_coef	0.0224	0.0013	0.0045
eps_clip	0.1031	0.2019	0.2233
learning rate	8.35e-4	4.77e-4	7.90e-4
n_neurons	256	64	256
repeat_p_coll.	5	4	1
vf_coef	0.8413	0.8958	0.2525

Table 1: The PPO hyperparameters for the Tianshou implementation. The parameter repeat_p_coll. refers to repeat_per_collect.

Hyperparam.	Quadrotor	Seeker-2d	Seeker-3d
batch bize	64	256	64
actor_lr	9.41e-4	6.23e-5	4.02e-5
critic_lr	4.32e-5	6.23e-4	3.32e-4
alpha	0.0509	0.1499	0.3643
gamma	0.9931	0.9911	0.9963
n_neurons	256	64	128
tau	0.0143	0.0563	0.0525
update_p_step	1.05	1.28	1.37

Table 2: The SAC hyperparameters for the Tianshou implementation. The parameter update_p_step refers to update_per_step.

Acknowledgements

We thank Paul Moritz Koebe for contributing to the implementation of the Seeker environment. We gratefully acknowledge that this project was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – SFB 1608 – 501798263; and DFG 458030766.

References

Althoff, M. 2010. *Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars*. Ph.D. thesis, Technische Universität München.

Althoff, M.; and Frehse, G. 2016. Combining zonotopes and support functions for efficient reachability analysis of linear systems. In *IEEE Conference on Decision and Control*, 7439–7446.

Boyd, S.; and Vandenberghe, L. 2004. *Convex optimization*. Cambridge university press.

Brahmanage, J.; Ling, J.; and Kumar, A. 2023. FlowPG: Action-constrained Policy Gradient with Normalizing Flows. In *Advances in Neural Information Processing Systems*, volume 36, 20118–20132.

Brahmanage, J. C.; Ling, J.; and Kumar, A. 2025. Leveraging Constraint Violation Signals for Action Constrained Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 15614–15621.

Brimkov, V. E.; Andres, E.; and Barneva, R. P. 2000. Object Discretization in Higher Dimensions. In *Discrete Geometry for Computer Imagery*, 210–221. Springer.

Burkardt, J. 2023. The Truncated Normal Distribution. Department of Scientific Computing Website, Florida State University, Tallahassee. Online resource.

Chalkis, A.; and Fisikopoulos, V. 2021. volesti: Volume Approximation and Sampling for Convex Polytopes in R. *The R Journal*, 13: 642–660.

Cover, T. M.; and Thomas, J. A. 2005. Differential Entropy. In *Elements of Information Theory*, 243–259. John Wiley & Sons, Ltd.

Genz, A.; and Cools, R. 2003. An adaptive numerical cubature algorithm for simplices. *ACM Transactions on Mathematical Software*, 29(3): 297–308.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning*, 1861–1870.

Hung, W.; Sun, S.-H.; and Hsieh, P.-C. 2025. Efficient Action-Constrained Reinforcement Learning via Acceptance-Rejection Method and Augmented MDPs. In *The Thirteenth International Conference on Learning Representations*.

Johnson, N. L.; Kotz, S.; and Balakrishnan, N. 1994. *Continuous Univariate Distributions, Volume 1*. John Wiley & Sons, Ltd, 2nd edition.

Kasaura, K.; Miura, S.; Kozuno, T.; Yonetani, R.; Hoshino, K.; and Hosoe, Y. 2023. Benchmarking Actor-Critic Deep Reinforcement Learning Algorithms for Robotics Control With Action Constraints. *IEEE Robotics and Automation Letters*, 8(8): 4449–4456.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In Bengio, Y.; and LeCun, Y., eds., *The Second International Conference on Learning Representations*.

Krasowski, H.; Thumm, J.; Müller, M.; Schäfer, L.; Wang, X.; and Althoff, M. 2023. Provably Safe Reinforcement Learning: Conceptual Analysis, Survey, and Benchmarking. *Transactions on Machine Learning Research*.

Kulmburg, A.; and Althoff, M. 2021. On the co-NP-completeness of the zonotope containment problem. *European Journal of Control*, 62: 84–91.

Lin, J. L.; Hung, W.; Yang, S. H.; Hsieh, P. C.; and Liu, X. 2021. Escaping from Zero Gradient: Revisiting Action-Constrained Reinforcement Learning via Frank-Wolfe Policy Optimization. *Proceedings of Machine Learning Research*, 161: 397–407.

Lovász, L.; and Vempala, S. 2006. Hit-and-Run from a Corner. *SIAM Journal on Computing*, 35(4): 985–1005.

Matoušek, J. 2002. Volumes in High Dimension. In *Lectures on Discrete Geometry*, 311–328. Springer.

Michalowicz, J. V.; Nichols, J. M.; and Bucholtz, F. 2013. *Handbook of Differential Entropy*. Chapman & Hall/CRC.

Munkres, J. R. 1991. *Analysis on Manifolds*. Addison–Wesley Publishing Company.

Murty, K. G.; and Kabadi, S. N. 1987. Some NP-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39(2): 117–129.

Robert, C. P.; and Casella, G. 2004. Monte Carlo Integration. In *Monte Carlo Statistical Methods*, 79–122. Springer.

Sanket, S.; Sinha, A.; Varakantham, P.; Andrew, P.; and Tambe, M. 2020. Solving Online Threat Screening Games using Constrained Action Space Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02): 2226–2235.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347.

Shangari, D.; and Chen, J. 2012. Partial monotonicity of entropy measures. *Statistics & Probability Letters*, 82(11): 1935–1940.

Stolz, R.; Krasowski, H.; Thumm, J.; Eichelbeck, M.; Gassert, P.; and Althoff, M. 2024. Excluding the Irrelevant: Focusing Reinforcement Learning through Continuous Action Masking. In *Advances in Neural Information Processing Systems*, volume 37, 95067–95094.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition.

Sutton, R. S.; McAllester, D.; Singh, S.; and Mansour, Y. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems*, volume 12.

Theile, M.; Bernardini, D.; Trumpp, R.; Piazza, C.; Caccamo, M.; and Sangiovanni-Vincentelli, A. L. 2024. Learning to Generate All Feasible Actions. *IEEE Access*, 12: 40668–40681.

Tong, Y. L. 1990. *Fundamental Properties and Sampling Distributions of the Multivariate Normal Distribution*, 23–61. Springer.

Towers, M.; Kwiatkowski, A.; Terry, J.; Balis, J. U.; De Cola, G.; Deleu, T.; Goulão, M.; Kallinteris, A.; Krimmel, M.; KG, A.; et al. 2024. Gymnasium: A Standard Interface for Reinforcement Learning Environments. *arXiv preprint arXiv:2407.17032*.

Weng, J.; Chen, H.; Yan, D.; You, K.; Duburcq, A.; Zhang, M.; Su, Y.; Su, H.; and Zhu, J. 2022. Tianshou: A Highly Modularized Deep Reinforcement Learning Library. *Journal of Machine Learning Research*, 23(267): 1–6.

Zabinsky, Z. B.; and Smith, R. L. 2013. Hit-and-Run Methods. In *Encyclopedia of Operations Research and Management Science*, 721–729. Boston, MA: Springer.