A Policy Search Method For Temporal Logic Specified Reinforcement Learning Tasks

Xiao Li, Yao Ma and Calin Belta

***Abstract*— Reward engineering is an important aspect of reinforcement learning. Whether or not the users' intentions can be correctly encapsulated in the reward function can significantly impact the learning outcome. Current methods rely on manually crafted reward functions that often requires parameter tuning to obtain the desired behavior. This operation can be expensive when exploration requires systems to interact with the physical world. In this paper, we explore the use of *temporal logic* (TL) to specify tasks in reinforcement learning. TL formula can be translated to a real-valued function that measures its level of satisfaction against a trajectory. We take advantage of this function and propose *temporal logic policy search* (TLPS), a model-free learning technique that finds a policy that satisfies the TL specification. A set of simulated experiments are conducted to evaluate the proposed approach.**

I. Introduction

Reinforcement learning (RL) has enjoyed groundbreaking success in recent years ranging from playing Atari games at super-human level [1], playing competitively with world champions in the game of Go [2] to generating visuomotor control policies for robots [3], [4]. Despite much effort being put into developing sample efficient algorithms, an important aspect of RL remains less explored. The reward function is the window for designers to specify the desired behavior and impose important constraints for the system. While most reward functions used in the current RL literature have been based on heuristics for relatively simple tasks, real world applications typically involve tasks that are logically more complex.

Commonly adopted reward functions take the form of a linear combination of basis functions (often quadratic) [5]. This type of reward function has limited expressibility and is semantically ambiguous because of its dependence on a set of weights. Reward functions of this form have been used to successfully learn high dimensional control tasks such as humanoid walking [6] and multiple household tasks (e.g. placing coat-hangers, twisting bottle caps, etc) [3]. However, parameter tuning of the reward function is required and this iteration is expensive for robotic tasks. Moreover, these tasks are logically straightforward in that there is little logical interactions between sub-tasks (such as sequencing, conjunction/disjunction, implication, etc).

Consider the harder task of learning to use an oven. The agent is required to perform a series of sub-tasks in the correct sequence (set temperature and timer $\rightarrow$ preheat $\rightarrow$ open oven door $\rightarrow$ place item in oven $\rightarrow$ close oven door). In addition, the agent has to make the simple decision of when to open the oven door and place the item (i.e. preheat finished *implies* open oven door). Tasks like this are commonly found in household environments (using the microwave, refrigerator or even a drawer) and a function that correctly maps the desired behavior to a real-valued reward can be difficult to design. If the semantics of the reward function can not be guaranteed, then an increase in the expected return will not necessarily represent better satisfaction of the task specification. This is referred to as reward hacking by [7].

Reward engineering has been briefly explored in the reinforcement learning literature. Authors of [8] and [9] provide general formalisms for reward engineering and discuss its significance. Authors of [10] proposed potential-based reward shaping and proved policy invariance under this type of reward transformation. Another line of work aims to infer a reward function from demonstration. This idea is called inverse reinforcement learning and is explored by [11] and [12].

In this paper, we adopt the expressive power of temporal logic and use it as a task specification language for reinforcement learning in continuous state and action spaces. Its quantitative semantics (also referred to as robustness degree or simply robustness) translate a TL formula to a real-valued function that can be used as the reward. By definition of the quantitative semantics, a robustness value of greater than zero *guarantees* satisfaction of the temporal logic specification.

Temporal logic (TL) has been adopted as the specification language for a wide variety of control tasks. Authors of [13] use linear temporal logic (LTL) to specify a persistent surveillance task carried out by aerial robots. Similarly, [14] and [15] applied LTL in traffic network control. Application of TL in reinforcement learning has been less investigated. [16] combined signal temporal logic (STL) with Q-learning while also adopting the log-sum-exp approximation of robustness. However, their focus is in the discrete state and action spaces, and ensured satisfiability by expanding the state space to a history dependent state space. This does not scale well for large or continuous state-action spaces which is often the case for control tasks.

Our main contributions in this paper are as follows:

- we present a model-free policy search algorithm, which we call temporal logic policy search (TLPS), that takes advantage of the robustness function to facilitate learning. We show that an optimal parameterized policy that maximizes the robustness could be obtained by solving a constrained optimization,
- a smoothing approximation of the robustness degree is proposed which is necessary for obtaining the gradients

of the objective and constraints. We prove that using the smoothed robustness as reward provides similar semantic guarantees to the original robustness definition while providing significant speedup in learning,
- finally, we demonstrate the performance of the proposed approach using simulated navigation tasks.

II. Preliminaries

A. Truncated Linear Temporal Logic (TLTL)

In this section, we provide definitions for TLTL (refer to our previous work [17] for a more elaborate discussion of TLTL). A TLTL formula is defined over predicates of form $f(s) < c$, where $f : \mathbb{R}^n \to \mathbb{R}$ is a function of state and c is a constant. We express the task as a TLTL formula with the following syntax:

$$\begin{aligned}\phi := \top \mid f(s) < c \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \\ \Diamond\phi \mid \Box\phi \mid \phi \,\mathcal{U}\, \psi \mid \phi \,\mathcal{T}\, \psi \mid \bigcirc \phi \mid \phi \Rightarrow \psi,\end{aligned} \tag{1}$$

where $\top$ is the boolean constant true, $f(s) < c$ is a predicate, $\neg$ (negation/not), $\wedge$ (conjunction/and), and $\vee$ (disjunction/or) are Boolean connectives, and $\Diamond$ (eventually), $\Box$ (always), $\mathcal{U}$ (until), $\mathcal{T}$ (then), $\bigcirc$ (next), are temporal operators. Implication is denoted by $\Rightarrow$ (implication). TLTL formulas are evaluated against finite time sequences of states $\{s_0, s_1, \ldots, s_T\}$.

We denote $s_t \in S$ to be the state at time t, and $s_{t:t+k}$ to be a sequence of states (state trajectory) from time t to $t+k$, i.e., $s_{t:t+k} = (s_t, s_{t+1}, ..., s_{t+k})$. The Boolean semantics of TLTL is defined as:

$$\begin{aligned}
s_{t:t+k} \models f(s) < c \quad &\Leftrightarrow \quad f(s_t) < c, \\
s_{t:t+k} \models \neg\phi \quad &\Leftrightarrow \quad \neg(s_{t:t+k} \models \phi), \\
s_{t:t+k} \models \phi \Rightarrow \psi \quad &\Leftrightarrow \quad (s_{t:t+k} \models \phi) \Rightarrow (s_{t:t+k} \models \psi), \\
s_{t:t+k} \models \phi \wedge \psi \quad &\Leftrightarrow \quad (s_{t:t+k} \models \phi) \wedge (s_{t:t+k} \models \psi), \\
s_{t:t+k} \models \phi \vee \psi \quad &\Leftrightarrow \quad (s_{t:t+k} \models \phi) \vee (s_{t:t+k} \models \psi), \\
s_{t:t+k} \models \bigcirc\phi \quad &\Leftrightarrow \quad (s_{t+1:t+k} \models \phi) \wedge (k > 0), \\
s_{t:t+k} \models \Box\phi \quad &\Leftrightarrow \quad \forall t' \in [t, t+k)\ s_{t':t+k} \models \phi, \\
s_{t:t+k} \models \Diamond\phi \quad &\Leftrightarrow \quad \exists t' \in [t, t+k)\ s_{t':t+k} \models \phi, \\
s_{t:t+k} \models \phi \,\mathcal{U}\, \psi \quad &\Leftrightarrow \quad \exists t' \in [t, t+k)\ s.t.\ s_{t':t+k} \models \psi \\
&\qquad \wedge (\forall t'' \in [t, t')\ s_{t'':t'} \models \phi), \\
s_{t:t+k} \models \phi \,\mathcal{T}\, \psi \quad &\Leftrightarrow \quad \exists t' \in [t, t+k)\ s.t.\ s_{t':t+k} \models \psi \\
&\qquad \wedge (\exists t'' \in [t, t')\ s_{t'':t'} \models \phi).
\end{aligned}$$

Intuitively, state trajectory $s_{t:t+k} \models \Box\phi$ (reads $s_{t:t+k}$ satisfies $\Box\phi$) if the specification defined by ϕ is satisfied for every subtrajectory $s_{t':t+k}$, $t' \in [t, t+k)$. Similarly, $s_{t:t+k} \models \Diamond\phi$ if ϕ is satisfied for at least one subtrajectory $s_{t':t+k}$, $t' \in [t, t+k)$. $s_{t:t+k} \models \phi \,\mathcal{U}\, \psi$ if ϕ is satisfied at every time step before ψ is satisfied, and ψ is satisfied at a time between t and $t+k$. $s_{t:t+k} \models \phi \,\mathcal{T}\, \psi$ if ϕ is satisfied at least once before ψ is satisfied between t and $t+k$. A trajectory s of duration k is said to satisfy formula ϕ if $s_{0:k} \models \phi$.

TLTL is equipped with quantitative semantics (robustness degree) , i.e., a real-valued function $\rho(s_{t:t+k}, \phi)$ that indicates how far $s_{t:t+k}$ is from satisfying or violating the specification ϕ. We define the task satisfaction measurement $\rho(\tau, \phi)$, which is recursively expressed as:

$$\begin{aligned}
\rho(s_{t:t+k}, \top) &= \rho_{max}, \\
\rho(s_{t:t+k}, f(s_t) < c) &= c - f(s_t), \\
\rho(s_{t:t+k}, \neg\phi) &= -\rho(s_{t:t+k}, \phi), \\
\rho(s_{t:t+k}, \phi \Rightarrow \psi) &= \max(-\rho(s_{t:t+k}, \phi), \rho(s_{t:t+k}, \psi)) \\
\rho(s_{t:t+k}, \phi_1 \wedge \phi_2) &= \min(\rho(s_{t:t+k}, \phi_1), \rho(s_{t:t+k}, \phi_2)), \\
\rho(s_{t:t+k}, \phi_1 \vee \phi_2) &= \max(\rho(s_{t:t+k}, \phi_1), \rho(s_{t:t+k}, \phi_2)), \\
\rho(s_{t:t+k}, \bigcirc\phi) &= \rho(s_{t+1:t+k}, \phi)\ (k > 0), \\
\rho(s_{t:t+k}, \Box\phi) &= \min_{t' \in [t, t+k)} (\rho(s_{t':t+k}, \phi)), \\
\rho(s_{t:t+k}, \Diamond\phi) &= \max_{t' \in [t, t+k)} (\rho(s_{t':t+k}, \phi)), \\
\rho(s_{t:t+k}, \phi \,\mathcal{U}\, \psi) &= \max_{t' \in [t, t+k)} (\min(\rho(s_{t':t+k}, \psi), \\
&\qquad \min_{t'' \in [t, t')} \rho(s_{t'':t'}, \phi))), \\
\rho(s_{t:t+k}, \phi \,\mathcal{T}\, \psi) &= \max_{t' \in [t, t+k)} (\min(\rho(s_{t':t+k}, \psi), \\
&\qquad \max_{t'' \in [t, t')} \rho(s_{t'':t'}, \phi))),
\end{aligned}$$

where ρ_{max} represents the maximum robustness value. Moreover, $\rho(s_{t:t+k}, \phi) > 0 \Rightarrow s_{t:t+k} \models \phi$ and $\rho(s_{t:t+k}, \phi) < 0 \Rightarrow s_{t:t+k} \not\models \phi$, which implies that the robustness degree can substitute Boolean semantics in order to enforce the specification ϕ.

Example 1: Consider specification $\phi = \Diamond(s > 5 \wedge s < 10)$, where s is a one dimensional state. Intuitively, this formula specifies that s eventually reaches region $(5, 10)$ for at least one time step. Suppose we have a state trajectory $s_{0:3} = s_0 s_1 s_2 = [11, 6, 7]$ of horizon 3. The robustness is $\rho(s_{0:3}, \phi) = \max_{t \in [0,3)} \Big(\min(10 - s_t, s_t - 5) \Big) = \max(-1, 1, 2) = 2$. Since $\rho(s_t, \phi) > 0$, $s_{0:1} \models \phi$ and the value $\rho(s_t, \phi) = 2$ is a measure of the satisfaction margin. Note that both states s_1 and s_2 stayed within the specified region, but s_2 "more" satisfies the predicate $(s > 5 \wedge s < 10)$ by being closer to the center of the region and thereby achieving a higher robustness value than s_1.

B. Markov Decision Process

In this section, we introduce the finite horizon infinite Markov decision process (MDP) and the semantics of a TLTL formula over an MDP. We start with the following definition:

Definition 1: A finite horizon infinite MDP is defined as a tuple $\langle S, A, p(\cdot|\cdot,\cdot)\rangle$, where $S \subseteq \mathbb{R}^n$ is the continuous state space; $A \subseteq \mathbb{R}^m$ is the continuous action space; $p(s'|s, a)$ is the conditional probability density of taking action $a \in A$ at state $s \in S$ and ending up in state $s' \in S$. We denote T as the horizon of MDP.

Given an MDP in Definition 1, a state trajectory of length T (denoted $\tau = s_{0:T-1} = (s_0, ..., s_{T-1})$) can be produced. The semantics of a TLTL formula ϕ over τ can be evaluated with the robustness degree $\rho(\tau, \phi)$ defined in the previous

section. $\rho(\tau, \phi) > 0$ implies that τ satisfies ϕ, i.e. $\tau \models \phi$ and vice versa. In the next section, we will take advantage of this property and propose a policy search method that aims to maximize the expected robustness degree.

III. Problem Formulation And Approach

We first formulate the problem of policy search with TLTL specification as follows:

Problem 1: Given an MDP in Definition 1 and a TLTL formula ϕ, find a stochastic policy $\pi(a|s)$ (π determines a probability of taking action a at state s) that maximizes the expected robustness degree

$$\pi^\star = \arg\max_{\pi} E_{p^\pi(\tau)} \left[\rho(\tau, \phi)\right], \tag{2}$$

where the expectation is taken over the trajectory distribution $p^\pi(\tau)$ following policy π, i.e.

$$p^\pi(\tau) = p(s_0) \prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t). \tag{3}$$

In reinforcement learning, the transition function $p(s'|s, a)$ is unknown to the agent. The solution to Problem 1 learns a stochastic time-varying policy $\pi(a_t|s_t)$ [18] which is a conditional probability density function of action a given current state s at time step t.

In this paper, policy π is a parameterized policy $\pi(a_t|s_t; \theta_t), \forall t = 1, \ldots, T$ (also written as π_θ in short, where $\theta = \{\theta_0, \theta_1, \ldots, \theta_{T-1}\}$) is used to represent the policy parameter. The objective defined in Equation (2) then becomes finding the optimal policy parameter θ^* such that

$$\theta^\star = \arg\max_{\theta} E_{p^{\pi_\theta}(\tau)} \left[\rho(\tau, \phi)\right]. \tag{4}$$

To solve Problem 1, we introduce temporal logic policy search (TLPS) - a model free RL algorithm. At each iteration, a set of sample trajectories are collected under the current policy. Each sample trajectory is updated to a new one with higher robustness degree by following the gradient of ρ while also keeping close to the sample so that dynamics is not violated. A new trajectory distribution is fitted to the set of updated trajectories. Each sample trajectory is then assigned a weight according to its probability under the updated distribution. Finally, the policy is updated with weight maximum likelihood. This process ensures that each policy update results in a trajectory distribution with higher expected robustness than the current one. Details of TLPS will be discussed in the next section.

As introduced in Section II-A, the robustness degree ρ consists of embedded $\max / \min$ functions and calculating the gradient is not possible. In Section V, we discuss the use of *log-sum-exp* to approximate the robustness function and provide proofs of some properties of the approximated robustness.

IV. Temporal Logic Policy Search (TLPS)

Given a TLTL formula ϕ over predicates of S, TLPS finds the parameters θ of a parametrized stochastic policy $\pi_\theta(a|s)$ that maximizes the following objective function.

$$J^{\pi_\theta} = E_{p^{\pi_\theta}}[\rho(\tau, \phi)], \ (T < \infty), \tag{5}$$

where $p^{\pi_\theta} = p^{\pi_\theta}(\tau)$ is defined in Equation (3).

In TLPS, we model the policy as a time-varying linear Gaussian, i.e. $\pi(a_t|s_t) = \mathcal{N}(K_t s_t + k_t, C_t)$ where K_t, k_t, C_t are the feedback gain, feed-forward gain and covariance of the policy at time t. (similar approach has been adopted in [19], [20]). And the trajectory distribution in Equation (3) is modeled as a Gaussian $p^{\pi_\theta}(\tau) = \mathcal{N}(\tau|\mu_\tau, \Sigma_\tau)$ where $\mu_\tau = (\mu_{s_0}, ..., \mu_{s_T})$ and $\Sigma_\tau = diag(\Sigma_{s_0}, ..., \Sigma_{s_T})$.

At each iteration, N sample trajectories are collected (denoted τ^i, $i \in [1, N]$). For each sample trajectory τ^i, we find an updated trajectory $\bar{\tau}^i$ by solving

$$\max_{\bar{\tau}^i} \hat{\rho}(\bar{\tau}^i, \phi), \ s.t. \ (\bar{\tau}^i - \tau^i)^T(\bar{\tau}^i - \tau^i) < \epsilon. \tag{6}$$

In the above equation, $\hat{\rho}$ is the *log-sum-exp* approximation of ρ. This is to take advantage of the many off-the-shelf nonlinear programming methods that require gradient information of the Lagrangian (sequential quadratic programming is used in our experiments). Using the log-sum-exp approximation we can show that its approximation error is bounded. In additional, the local ascending directions on the approximated surface coincide with the actual surface given mild constraints (these will be discussed in more detail in the next section). Equation (6) aims to find a new trajectory that achieves higher robustness. The constraint is to limit the deviation of the updated trajectory from the sample trajectory so the system dynamics is not violated.

After we obtain a set of updated trajectories, an updated trajectory distribution $\bar{p}(\tau) = \mathcal{N}(\tau|\bar{\mu}_\tau, \bar{\Sigma}_\tau)$ is fitted using

$$\bar{\mu}_\tau = \frac{1}{N}\sum_{i=1}^{N} \bar{\tau}^i, \ \bar{\Sigma}_\tau = \frac{1}{N}\sum_{i=1}^{N} (\bar{\tau}^i - \bar{\mu}_\tau)(\bar{\tau}^i - \bar{\mu}_\tau)^T, \tag{7}$$

The last step is to update the policy. We will only be updating the feed-forward terms k_t and the covariance C_t. The feedback terms K_t is kept constant (the policy parameters are $\theta_t = (k_t, C_t)$, $t \in [0, T)$). This significantly reduces the number of parameters to be updated and increases the learning speed. For each sample trajectory, we obtain its probability under $\bar{p}(\tau)$

$$p(\tau^i) = \mathcal{N}(\tau^i|\bar{\mu}_\tau, \bar{\Sigma}_\tau) \tag{8}$$

($p(\tau^i)$ is also written in short as p^i) where $i \in [1, N]$ is the sample index. Using these probabilities, a normalized weighting for each sample trajectory is calculated using the softmax function $w^i = e^{\alpha p^i} / \sum_{i=1}^{N} e^{\alpha p^i}$ ($\alpha > 0$ is a

parameter to be tuned). Finally, similar to [19], the policy is updated using weighted maximum likelihood by

$$k_t' = \sum_{i=1}^{N} w^i k_t^i$$
$$C_t' = \sum_{i=1}^{N} w^i (k_t^i - k_t')(k_t^i - k_t')^T. \tag{9}$$

According to [21], such update strategy will result in convergence. The complete algorithm is described in Algorithm 1.

Algorithm 1 Temporal Logic Policy Search

Inputs: Episode horizon T, batch size N, KL constraint parameter ϵ, smoothed robustness function $\hat{\rho}(s_{0:T}, \phi)$, softmax parameter $\alpha > 0$
Initialize policy $\pi \leftarrow (K_t, k_t, C_t)$
Initialize trajectory buffer $\mathcal{B} \leftarrow \emptyset$
for $m = 1$ to *number of training episodes* **do**
  τ_m = SampleTrajectories(π, T)
  Store τ_m in $\mathcal{B}$
  if Size($\mathcal{B}$) $\geq N$ **then**
    $\bar{\tau}^i \leftarrow$ GetUpdatedTrajectories(τ^i) **for** $i = 1$ to N **end for** ▷ Using Equation (6)
    $\bar{\mu}_\tau, \bar{\Sigma}_\tau \leftarrow$ FitTrajectoryDistribution($\{\tau_1, ..., \tau_N\}$) ▷ Using Equation (7)
    for i=1 to N **do**
      $p^i \leftarrow \mathcal{N}(\tau^i | \bar{\mu}_\tau, \bar{\Sigma}_\tau)$
      $w^i = \frac{e^{\alpha p^i}}{\sum_{i=1}^{N} e^{\alpha p^i}}$
    end for
    for t = 0 to T-1 **do**
      $k_t' \leftarrow \sum_i^N w^i k_t^i$
      $C_t' \leftarrow \sum_i^N w^i (k_t^i - k_t')(k_t^i - k_t')^T$
    end for
    Clear buffer $\mathcal{B} \leftarrow \emptyset$
  end if
end for

V. Robustness Smoothing

In the TLPS algorithm introduced in the previous section, one of the steps requires solving a constrained optimization problem that maximizes the robustness (Equation (6)). The original robustness definition in Section II-A is non-differentiable and thus rules out many efficient gradient-based methods. In this section we adopt a smooth approximation of the robustness function using *log-sum-exp*. Specifically

$$\max(x_1, ..., x_n) \approx \frac{1}{\beta} \log \sum_i^n \exp(\beta x_i)$$
$$\min(x_1, ..., x_n) \approx -\frac{1}{\beta} \log \sum_i^n \exp(-\beta x_i), \tag{10}$$

where $\beta > 0$ is a smoothness parameter. We denote an iterative max-min function as

$$M(x) = \text{mami}_i f_i(x),$$

where $f_i(x) = \text{mami}_j f_j(x)$. mami denotes a function as mami $\in \{\max, \min, \mathcal{I}\}$ where $\mathcal{I}$ is a operator such that $\mathcal{I} f_j(x) = f_j(x)$. i and j are index of the functions in mami and can be any positive integer. As we showed in Section II-A, any robustness function could be expressed as an iterative max-min function.

Following the *log-sum-exp* approximation, any iterative max-min function (i.e., the robustness of any TL formula) can be approximated as follows

$$\hat{M}(x) = \frac{1}{\beta} \log \left(\sum_i \exp\left(\beta f_i(x)\right) \right),$$

where $\beta_i > 0$ if $\text{mami}_i = \max_i$ and $\beta_i < 0$ if $\text{mami}_i = \min_i$. In the reminder of this section, we provide three lemmas that show the following:

- the approximation error between $M(x)$ and $\hat{M}(x)$ approaches zero as $\beta_i \to \infty$. This error is always bounded by the log of the number of $f(x)$ which is determined by the number of predicates in the TL formulae and the horizon of the problem. Tuning β_i trades off between differentiability of the robustness function and approximation error.
- despite the error introduced by the approximation, the optimal points remain invariant (*i.e.* $\text{argmax}_x M(x) = \text{argmax}_x \hat{M}(x)$). This result provides guarantee that the optimal policy is unchanged when using the approximated TL reward,
- even though the *log-sum-exp* approximation smooths the robustness function. Locally the ascending directions of $M(x)$ and $\hat{M}(x)$ can be tuned to coincide with small error and the deviation is controlled by the parameter β. As many policy search methods are local methods that improve the policy near samples, it is important to ensure that the ascending direction of the approximated TL reward does not oppose that of the real one.

Due to space constraints, we will only provide sketches of the proofs for the lemmas.

Lemma 1: Let N_i be the number of terms of mami_i, M and $\hat{M}$ satisfy

$$M - \sum_{i \in S_{min}} \frac{1}{|\beta_i|} \log N_i \leq \hat{M} \leq M + \sum_{i \in S_{max}} \frac{1}{\beta_i} \log N_i$$

where $S_{min} = \{i : \text{mami}_i = \min_i\}$ and $S_{max} = \{i : \text{mami}_i = \max_i\}$.

Proof: For simplicity and without loss of generality, we illustrate the proof of Lemma 1 by constructing an approximation for a finite *max-min-max* problem

$$\Phi(x) = \max_{i \in I} \min_{j \in J} \max_{k \in K} f_{i,j,k}(x).$$

Let $M_I = |I|$, $M_J = |J|$, $M_K = |K|$, and $\beta_I > 0$, $\beta_J < 0$, $\beta_K > 0$. Firstly, we define $\Phi_j(x) = \max_{k \in K} f_{i,j,k}(x)$.

Straightforward algebraic manipulation reveals that

$$\log\left(\sum_{j\in J}\exp(\beta_J\Phi_j)\right)+\frac{\beta_J}{\beta_K}\log(M_K) \tag{11}$$
$$\leq \log\left(\sum_{j\in J}\left[\sum_{k\in K}\exp(\beta_K f_{i,j,k}(x))\right]^{\frac{\beta_J}{\beta_K}}\right)$$
$$\leq \log\left(\sum_{j\in J}\exp(\beta_J\Phi_j)\right).$$

Furthermore, let us define $\Phi_i = \min_{j\in J}\Phi_j$, we have

$$\beta_J\Phi_i \leq \log\left(\sum_{j\in J}\exp(\beta_J\Phi_j)\right) \leq \log(M_J)+\beta_J\Phi_i.$$

By substituting into Equation (11), we obtain

$$\beta_J\Phi_i+\log(M_J) \geq \log\left(\sum_{j\in J}\exp(\beta_J\Phi_j)\right)$$
$$\geq \beta_J\Phi_i+\frac{\beta_J}{\beta_K}\log(M_K).$$

Multiplying $\frac{1}{\beta_J}$ on both side, then

$$\log(\sum_{i\in I}\exp(\beta_I\Phi_i))+\frac{\beta_I}{\beta_J}\log(M_J)$$
$$\leq \log\left(\sum_{i\in I}\left[\sum_{j\in J}\left(\sum_{k\in K}\exp(\beta_K f_{i,j,k}(x))\right)^{\frac{\beta_J}{\beta_K}}\right]^{\frac{\beta_I}{\beta_J}}\right)$$
$$\leq \log(\sum_{i\in I}\exp(\beta_I\Phi_i))+\frac{\beta_I}{\beta_K}\log(M_K).$$

Finally, let $\Phi = \max_{i\in I}\Phi_i$, then we have

$$\exp(\beta_I\Phi) \leq \sum_{i\in I}\exp(\beta_I\Phi_i) \leq M_I\exp(\beta_I\Phi)$$
$$\beta_I\Phi \leq \log(\sum_{i\in I}\exp(\beta_I\Phi_i)) \leq \log(M_I)+\beta_I\Phi \tag{12}$$

Substitute into Equation (12)

$$\beta_I\Phi+\frac{\beta_I}{\beta_J}\log(M_J)$$
$$\leq \log\left(\sum_{i\in I}\left[\sum_{j\in J}\left(\sum_{k\in K}\exp(\beta_K f_{i,j,k}(x))\right)^{\frac{\beta_J}{\beta_K}}\right]^{\frac{\beta_I}{\beta_J}}\right)$$
$$\leq \beta_I\Phi+\log(M_I)+\frac{\beta_I}{\beta_K}\log(M_K).$$

Then we conclude the proof. ■

Lemma 2: Suppose $X^* = \{x^* : x^* \in \text{argmax}_x M(x)\}$, there exist a positive constant B such that for all $|\beta| \geq B$ x^* is also one of the maximum point of $\hat{M}(x)$ for any x^*, *i.e.*

$$x^* \in \text{argmax}_x \hat{M}(x).$$

Proof: We start by considering M as a maximum function, i.e. $M(x) = \max_i f_i(x)$.let us denote $I_{max} = \text{argmax}_i f_i(x^*)$, then $x^* \in \text{argmax}_x \hat{M}(x)$ when

$$\sum_{i\neq I_{max}}\exp(\beta f_i(x^*)) - \sum_{i\neq I_{max}}\exp(\beta f_i(x))$$
$$\leq \exp(\beta f_{Imax}(x^*)) - \exp(\beta f_{Imax}(x)).$$

There always exists a positive constant B, such that for all $\beta > B$ the above statement holds. Lemma 2 can be obtained by using the above proof for the mami function in general. ■

Lemma 3: Let us denote the sub-gradient of M as $\frac{\partial M}{\partial x} = \{\frac{\partial M}{\partial x_1},\ldots,\frac{\partial M}{\partial x_N}\}$ and the gradient of $\hat{M}$ as $\frac{\partial \hat{M}}{\partial x} = \{\frac{\partial \hat{M}}{\partial x_1},\ldots,\frac{\partial \hat{M}}{\partial x_N}\}$. There exists a positive constant B such that for all $|\beta| \geq B$, $\frac{\partial M}{\partial x}$ and $\frac{\partial \hat{M}}{\partial x}$ satisfy

$$\langle\frac{\partial M}{\partial x},\frac{\partial \hat{M}}{\partial x}\rangle \geq 0,$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product.

Proof: Here we will only provide the proof when M is a point-wise maximum of convex functions. One can generalize it to any iterative max-min function using the chain rule. Supposing $M(x) = \max_i f_i(x)$, the sub-gradient of $M(x)$ is

$$\frac{\partial M}{\partial x} = \partial f_i(x), i \in I(x),$$

where $I(x) = \{i|f_i(x) = f(x)\}$ is the set of "active" functions. The corresponding $\hat{M}$ is defined as

$$\hat{M} = \frac{1}{\beta}\log\left(\sum_i \exp(\beta f_i(x))\right),$$

where its first order derivative is

$$\frac{\partial \hat{M}}{\partial x} = \sum_i \frac{\exp(\beta f_i(x))\partial f_i(x)}{\sum_k \exp(\beta f_k(x))}.$$

$\langle\frac{\partial M}{\partial x},\frac{\partial \hat{M}}{\partial x}\rangle > 0$ if

$$\frac{\exp(\beta f_i(x))}{\sum_k \exp(\beta f_k(x))}f_i(x)$$
$$\geq \sum_{j\notin I(x)}\frac{\exp(\beta f_j(x))}{\sum_k \exp(\beta f_k(x))}f_j(x), \forall i \in I(x).$$

Therefore, there always exists a positive constant B, such that $\langle\frac{\partial M}{\partial x},\frac{\partial \hat{M}}{\partial x}\rangle > 0$ holds for all $\beta > B$. ■

VI. Case Studies

In this section, we apply TLPS on a vehicle navigation example. As shown in Figure 1, the vehicle navigates in a 2D environment. It has a 6 dimensional continuous state feature space $s = [x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}]$ where (x, y) is the position of its center and θ is the angle its heading makes with the x-axis. Its 2 dimensional action space $a = [a_v, a_\Phi]$ consists

of the forward driving speed and the steering angle of its front wheels. The car moves according to dynamics

$$\begin{aligned}\dot{x} &= a_v \cos\theta \\ \dot{y} &= a_v \sin\theta \\ \dot{\theta} &= \frac{a_v}{L}\tan a_\Phi\end{aligned} \tag{13}$$

with added Gaussian noise (L is the distance between the front and rear axles). However the learning agent is not provided with this model and needs to learn the desired control policy through trial-and-error.

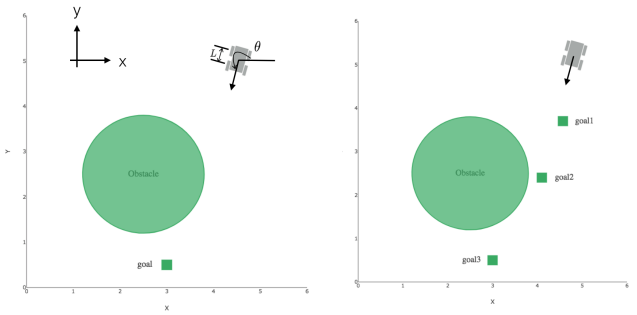

Fig. 1 : Vehicle navigation task using TLTL specifications. The vehicle is shown in brown, the obstacle is shown as the green circle and the goals are shown as the green squares. *left*: Task 1 is to reach the goal while avoiding the obstacle. *right*: Task 2 is to visit goals 1,2,3 in this order while avoiding the obstacle

We test TLPS on two tasks with increasing difficulty. In the first task, the vehicle is required to reach the goal g while avoiding the obstacle o. We express this task as a TLTL specification

$$\begin{aligned}\phi_1 =& \Diamond(x > x_g^l \wedge x < x_g^u \wedge y > y_g^l \wedge y < y_g^u) \wedge \\ & \Box(d_o > r_o).\end{aligned} \tag{14}$$

In Equation (14), $(x_g^l, x_g^u, y_g^l, y_g^u)$ defines the square shaped goal region, d_o is the Euclidean distance between the vehicle's center and the center of the obstacle, r_o is the radius of the obstacle. In English, ϕ_1 describes the task of "*eventually* reach goal ***g*** *and always* stay away from the obstacle". Using the quantitative semantics described in Section II-A , the robustness of ϕ_i is

$$\begin{aligned}&\rho_1(\phi_1, (x,y)_{0:T}) = \\ &\min\Bigg(\max_{t\in[0,T)}\Big(\min\Big(x_t - x_g^l, x_g^u - x_t, y_t - y_g^l, y_g^u - y_t\Big)\Big), \\ &\min_{t\in[0,T)}\Big(d_o^t - r_o\Big)\Bigg),\end{aligned} \tag{15}$$

where (x_t, y_t) and d_o^t are the vehicle position and distance to obstacle center at time t. Using the *log-sum-exp*, approximation for $\rho_1(\phi_1, (x,y)_{0:T})$ can be obtained as

$$\begin{aligned}&\hat{\rho}_1(\phi_1, (x,y)_{0:T}) = \\ &-\frac{1}{\beta}\log\sum_{t=0}^{T}\Big(\exp[-\beta(x_t - x_g^l)] + \exp[-\beta(x_g^u - x_t)] + \\ &\exp[-\beta(y_t - y_g^l)] + \exp[-\beta(y_g^u - y_t)] + \exp[-\beta(d_o^t - r_o)]\Big).\end{aligned} \tag{16}$$

Because we used the same β throughout the approximation, intermediate $\log$ and $\exp$ cancel and we end up with Equation (16). $\hat{\rho}_1(\phi_1, (x,y)_{0:T})$ is used in the optimization problem defined in Equation (6).

In task 2, the vehicle is required to visit goals 1, 2, 3 in this specific order while avoiding the obstacle. Expressed in TLTL results in the specification

$$\begin{aligned}\phi_2 =& (\psi_{g_1} \mathcal{T} \psi_{g_2} \mathcal{T} \psi_{g_3}) \wedge (\neg(\psi_{g_2} \vee \psi_{g_3}) \mathcal{U} \psi_{g_1}) \wedge \\ & (\neg(\psi_{g_3}) \mathcal{U} \psi_{g_2}) \wedge (\bigwedge_{i=1,2,3} \Box(\psi_{g_i} \Rightarrow \bigcirc\Box\neg\psi_{g_i})) \wedge \\ & \Box(d_o > r_o),\end{aligned} \tag{17}$$

where $\bigwedge$ is a shorthand for a sequence of conjunction, $\psi_{g_i} : x > x_{g_i}^l \wedge x < x_{g_i}^u \wedge y > y_{g_i}^l \wedge y < y_{g_i}^u$ are the predicates for goal g_i. In English, ϕ_2 states "visit g_1 *then* g_2 *then* g_3, *and don't* visit g_2 *or* g_3 *until* visiting g_1, *and don't* visit g_3 *until* visiting g_2, *and always* if visited g_i *implies next always don't* visit g_i (don't revisit goals), *and always* avoid the obstacle" . Due to space constraints the robustness of ϕ_2 and its approximation will not be explicitly presented, but it will take a similar form of nested $\min()/\max()$ functions that can be generated from the quantitative semantics of TLTL.

During training time, the obstacle is considered "penetrable" in that the car can surpass its boundary with a negative reward granted according to the penetrated depth. In practice we find that this facilitates learning compared to a single negative reward given at contact with the obstacle and restarting the episode.

Each episode has a horizon $T = 200$ time-steps. 40 episodes of sample trajectories are collected and used for each update iteration. The policy parameters are initialized randomly in a given region (the policy covariances should be initialized to relatively high values to encourage exploration). Each task is trained for 50 iterations and the results are presented in Figures 2 and 3. Figure 2 shows sample trajectory distributions for selected iterations. Trajectory distributions are illustrated as shaded regions with width equal to 2 standard deviations. Lighter shade indicates earlier time in the training process. We used $\beta = 9$ for this set of results. We can see from Figure 2 that the trajectory distribution is able to converge and satisfy the specification. Satisfaction occurs much sooner for task 1 (around 30 iterations) compared with task 2 (around 50 iterations).

Figure 3 compares the average robustness (of 40 sample trajectories) per iteration for TLPS with different values

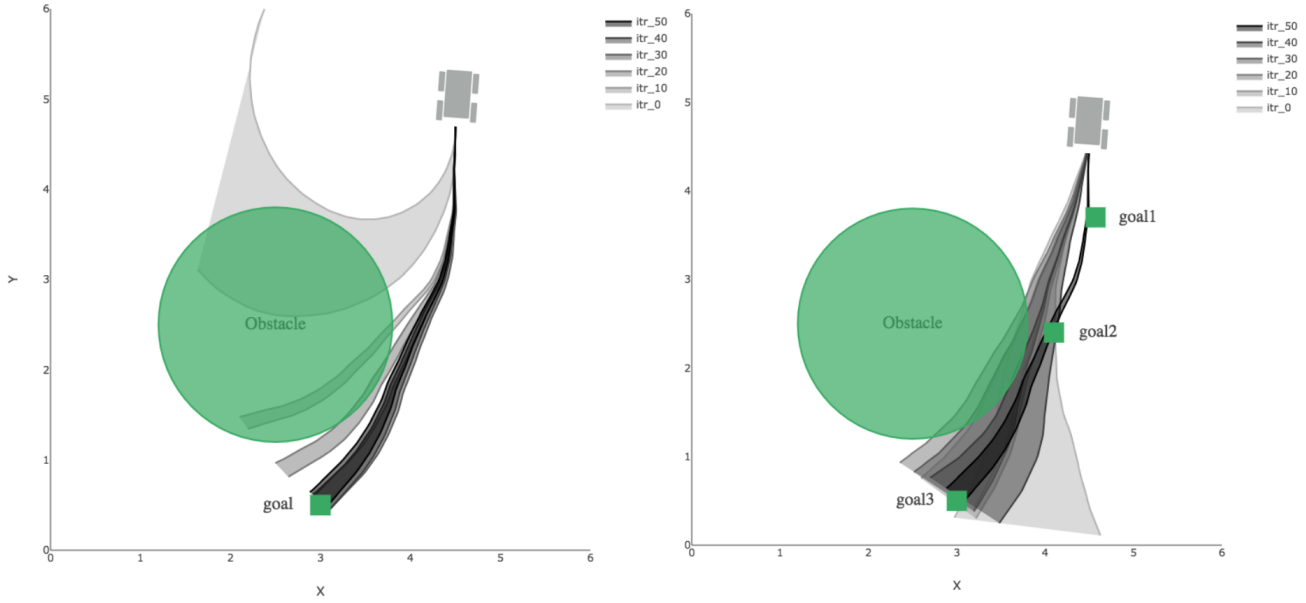

Fig. 2 : Sample trajectory distributions for selected iterations for *left*: task 1, *right*: task 2. Each iteration consists of 40 sample trajectories each having a horizon of 200 time-steps. The width of each distribution is 2 standard deviations and color represent recency in the training process (lighter color indicates earlier time in training).

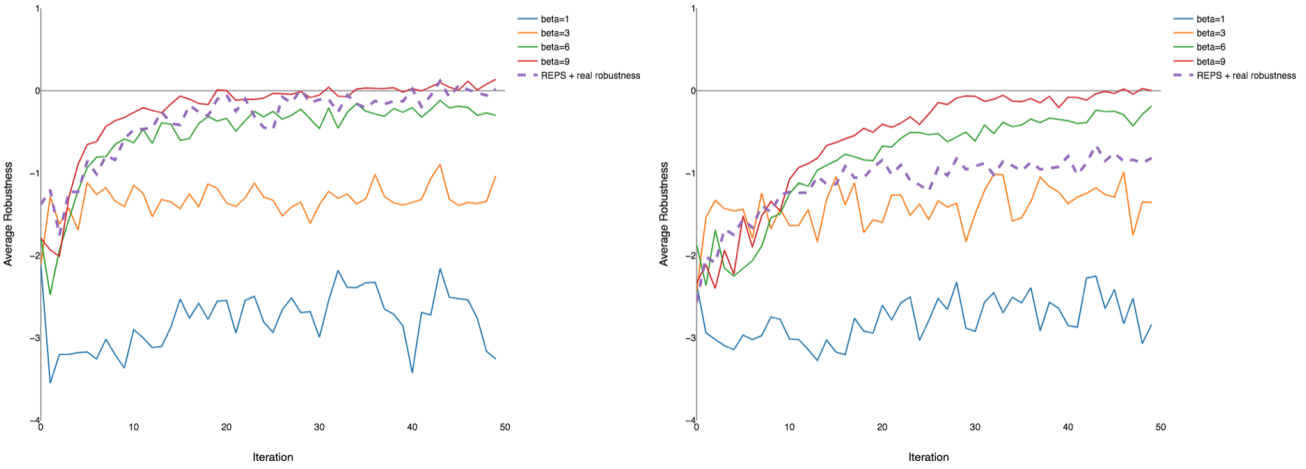

Fig. 3 : Average return vs training iteration for *left*: task 1, *right*: task2. The average return is represented as the original robustness value calculated from sample trajectories. TLPS is compared with varying β. REPS with the original robustness as terminal reward is used as a baseline.

of the approximation parameters β in (10). As a baseline, we also compare TLPS with episode-based relative entropy policy search (REPS) [18]. The original robustness function is used as the terminal reward for REPS and our previous work [17] has shown that this combination outperforms heuristic reward designed for the same robotic control task. The magnitude of robustness value changes with varying β. Therefore, in order for the comparison to be meaningful (putting average returns on the same scale), sample trajectories collected for each comparison case are used to calculate their original robustness values against the TLTL formula and plotted in Figure 3 (a similar approach taken in [17]). The original robustness is chosen as the comparison measure for its semantic integrity (value greater than zero indicates satisfaction).

Results in Figure 3 shows that larger β results in faster convergence and higher average return. This is consistent with the results of Section V since larger β indicates lower approximation error. However, this advantage diminishes as β increases due to the approximated robustness function losing differentiability. For the relatively easy task 1, TLPS performed comparatively with REPS. However, for the harder task 2, TLPS exhibits a clear advantage both in terms of rate of convergence and quality of the learned policy.

TLPS is a local policy search method that offers gradual policy improvement, controllable policy space exploration and smooth trajectories. These characteristics are desirable for learning control policies for systems that involve physical interactions with the environment. S (likewise for other local RL methods). Results in Figure 3 show a rapid exploration decay in the first 10 iterations and little improvement is seen after the 40^{th} iteration. During experiments, the authors find that adding a policy covariance damping schedule can help with initial exploration and final convergence. A principled exploration strategy is possible future work.

Similar to many policy search methods, TLPS is a local method. Therefore, policy initialization is a critical aspect of the algorithm (compared with value-based methods such as Q-learning). In addition, because the trajectory update step in Equation (6) does not consider the system dynamics and relies on being close to sample trajectories, divergence can occur with a small β or a large learning rate. Making the algorithm more robust to hyperparameter changes is also an important future direction.

VII. Conclusion

As reinforcement learning research advance and more general RL agents are developed, it becomes increasingly important that we are able to correctly communicate our intentions to the learning agent. A well designed RL agent will be proficient at finding a policy that maximizes its returns, which means it will exploit any flaws in the reward function that can help it achieve this goal. Human intervention can sometimes help alleviate this problem by providing additional feedback. However, as discussed in [8], if the communication link between human and the agent is unstable (space exploration missions) or the agent operates on a timescale difficult for human to respond to (financial trading agent), it is critical that we are confident about what the agent will learn.

In this paper, we applied temporal logic as the task specification language for reinforcement learning. The quantitative semantics of TL is adopted for accurate expression of logical relationships in an RL task. We explored robustness smoothing as a means to transform the TL robustness to a differentiable function and provided theoretical results on the properties of the smoothed robustness. We proposed temporal logic policy search (TLPS), a model-free method that utilizes the smoothed robustness and operates in continuous state and action spaces. Simulation experiments are conducted to show that TLPS is able to effectively find control policies that satisfy given TL specifications.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. a. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, and K. Kavukcuoglu, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7585, pp. 484–489, 2016. [Online]. Available: http://dx.doi.org/10.1038/nature16961

[3] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-End Training of Deep Visuomotor Policies," *Arxiv*, p. 6922, 2015. [Online]. Available: http://arxiv.org/abs/1504.00702

[4] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection," *arXiv*, 2016. [Online]. Available: http://arxiv.org/abs/1603.02199v1

[5] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," *arXiv preprint arXiv:1610.00633*, 2016.

[6] J. Peters, S. Vijayakumar, and S. Schaal, "Reinforcement learning for humanoid robotics," in *Proceedings of the third IEEE-RAS international conference on humanoid robots*, 2003, pp. 1–20.

[7] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete Problems in AI Safety," pp. 1–29, 2016. [Online]. Available: http://arxiv.org/abs/1606.06565

[8] D. Dewey, "Reinforcement learning and the reward engineering principle," in *2014 AAAI Spring Symposium Series*, 2014.

[9] I. Arel, "The threat of a reward-driven adversarial artificial general intelligence," in *Singularity Hypotheses*. Springer, 2012, pp. 43–60.

[10] "Policy invariance under reward transformations : Theory and application to reward shaping," *Sixteenth International Conference on Machine Learning*, vol. 3, pp. 278–287, 1999.

[11] A. Ng and S. Russell, "Algorithms for inverse reinforcement learning," *Proceedings of the Seventeenth International Conference on Machine Learning*, vol. 0, pp. 663–670, 2000. [Online]. Available: http://www-cs.stanford.edu/people/ang/papers/icml00-irl.pdf

[12] P. Sermanet, K. Xu, and S. Levine, "Unsupervised perceptual rewards for imitation learning," *arXiv preprint arXiv:1612.06699*, 2016.

[13] K. Leahy, D. Zhou, C.-I. Vasile, K. Oikonomopoulos, M. Schwager, and C. Belta, "Persistent surveillance for unmanned aerial vehicles subject to charging and temporal logic constraints," *Autonomous Robots*, vol. 40, no. 8, pp. 1363–1378, 2016.

[14] S. Sadraddini and C. Belta, "A provably correct mpc approach to safety control of urban traffic networks," in *American Control Conference (ACC), 2016*. IEEE, 2016, pp. 1679–1684.

[15] S. Coogan, E. A. Gol, M. Arcak, and C. Belta, "Traffic network control from temporal logic specifications," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 2, pp. 162–172, 2016.

[16] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-learning for robust satisfaction of signal temporal logic specifications," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, 2016, pp. 6565–6570.

[17] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," *IEEE International Conference on Intelligent Robots and Systems*, 2017.

[18] M. P. Deisenroth, "A Survey on Policy Search for Robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1, pp. 1–142, 2011. [Online]. Available: http://www.nowpublishers.com/articles/foundations-and-trends-in-robotics/ROB-021

[19] Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine, "Path integral guided policy search," *arXiv preprint arXiv:1610.00529*, 2016.

[20] W. H. Montgomery and S. Levine, "Guided policy search via approximate mirror descent," in *Advances in Neural Information Processing Systems*, 2016, pp. 4008–4016.

[21] F. Stulp and O. Sigaud, "Path integral policy improvement with covariance matrix adaptation," *ICML*, 2012.