Robust Policy Search with Applications to Safe Vehicle Navigation

Matthew Sheckells[1], Gowtham Garimella[1], and Marin Kobilarov[1]

***Abstract*— This work studies the design of reliable control laws of robotic systems operating in uncertain environments. We introduce a new approach to stochastic policy optimization based on probably approximately correct (PAC) bounds on the expected performance of control policies. An algorithm is constructed which directly minimizes an upper confidence bound on the expected cost of trajectories instead of employing a standard approach based on the expected cost itself. This algorithm thus has built-in robustness to uncertainty, since the bound can be regarded as a certificate for guaranteed future performance. The approach is evaluated on two challenging robot control scenarios in simulation: a car with side slip and a quadrotor navigating through obstacle-ridden environments. We show that the bound accurately predicts future performance and results in improved robustness measured by lower average cost and lower probability of collision. The performance of the technique is studied empirically and compared to several existing policy search algorithms.**

I. Introduction

Consider an autonomous vehicle executing a task, such as safely reaching a desired goal location while avoiding obstacles in an uncertain environment. The vehicle is guided using a control policy, i.e. a function mapping from the perceived robot and environment state to the vehicle control inputs. Complete knowledge of the vehicle model and sources of uncertainty might not be available. This work is motivated by the question: can such a policy be computed to satisfy high-confidence guarantees for successful completion of the task? Such guarantees correspond to statements such as "with 99% confidence the robot will reach the goal within the next 60 seconds," or "with 99% chance the robot will not collide with obstacles." Deriving such guarantees before actually executing the policy effectively certifies the vehicle performance and safety with respect to the uncertainty in the world, and could prove essential for deploying truly reliable robotic systems.

This work introduces a stochastic policy search method that is based on direct optimization of such high-confidence bounds. At the core of the approach lies a recently derived sample complexity bound for stochastic optimization [1] which provides guarantees on the performance of a policy based on previous executions of (possibly) different policies. Using the probably approximately correct (PAC) learning framework first introduced by [2], the bound certifies the performance of the policy with some probability $1-\delta$, where $\delta \in (0,1)$ denotes the confidence level. In this work, we propose a new policy search algorithm which directly minimizes this PAC bound. The algorithm offers several advantages over existing methods. Since it minimizes the bound on the cost instead of the standard average cost itself, the algorithm is expected to have some degree of "built-in" robustness. The bound provides a certificate which could prove crucial for deploying the algorithm on real-world systems operating in performance and safety-critical settings. Furthermore, this technique does not require any parameter tuning since all parameters are computed directly from the optimization and are automatically refined as the vehicle gains new experience. Finally, the approach does not make any simplifying assumptions about the vehicle model or form of uncertainty, such as linearity or Gaussianity, and is applicable to any general stochastic optimization problem, beyond policy optimization in robotics.

Problem Formulation: A common way to pose a policy search problem is through the optimization

$$\xi^* = \arg\min_{\xi} \mathbb{E}_{\tau \sim p(\cdot|\xi)}[J(\tau)], \tag{1}$$

where ξ is a vector of decision variables, τ represents the system response through the probability density $p(\tau|\xi)$, and $J(\tau)$ defines a positive cost function. It is assumed that $p(\tau|\xi)$ is known or can be sampled from (e.g. from a real system or from a high-quality physics simulator). Instead of directly optimizing ξ, a typical approach in stochastic optimization and policy search (e.g. see [3], [4] for an overview) is to construct a stochastic policy, for instance encoded as $\xi \sim \pi(\cdot|\nu)$, where the vector ν denotes the stochastic policy parameters. Policy search then takes the form

$$\min_{\nu} \hat{\mathcal{J}}(\nu) + \alpha D(\nu, \nu_0), \tag{2}$$

where $\hat{\mathcal{J}}$ is an empirical average of the cost (or regret) over N past experiences, $D(\nu,\nu_0)$ is a regularizing distance to a prior policy ν_0, and $\alpha > 0$ is a manually chosen mixing factor (or in some cases corresponds to a Lagrangian multiplier for a constraint $D(\nu,\nu_0) \leq \epsilon$ with manually chosen ϵ). In contrast, the proposed algorithm performs an optimization of the form

$$\min_{\nu,\alpha} \hat{\mathcal{J}}_\alpha(\nu) + \alpha d(\nu, \nu_0) + \phi(\alpha, N, \delta), \tag{3}$$

where $\hat{\mathcal{J}}_\alpha$ is a *robust empirical estimate* of $\mathcal{J} \triangleq \mathbb{E}[J]$, with $d(\cdot,\cdot)$ denoting a distance between distributions different than $D(\cdot,\cdot)$ defined in (2) and to be given in more detail later. The additional *concentration-of-measure* term ϕ reflects the discrepancy between the empirical cost $\hat{\mathcal{J}}_\alpha$ and the true mean cost $\mathcal{J}$. The expression in (3) (denoted $\mathcal{J}^+$) is in fact a high-confidence bound on the expected cost, i.e. with probability

[1]M. Sheckells, G. Garimella and M. Kobilarov are with the Department of Computer Science and the Department of Mechanical Engineering, Johns Hopkins University, 3400 N Charles Str, Baltimore, MD 21218, USA msheckells|ggarime1|marin@jhu.edu

$1-\delta$ it holds that $\mathcal{J} \leq \mathcal{J}^+$. This paper implements a policy search algorithm minimizing bounds of the form (3) and investigates the resulting performance in comparison with related standard methods.

Related Work: The proposed algorithm falls in the category of direct policy search techniques closely related to several prior methods. Reward-Weighted Regression (RWR) is an iterative policy search algorithm that was first introduced in [5] to perform system identification for operational space control and has since been used for policy search. Relative Entropy Policy Search (REPS) is an extension of RWR, which uses an information-theoretic approach to bound how much the policy is allowed to change between iteratons [6]. Policy learning by Weighting Exploration with Returns (PoWER) [7] uses step-based weighted maximum likelihood estimates to perform policy updates. We compare the proposed algorithm to both RWR and REPS in §IV.

Policy gradient algorithms use gradient descent to minimize the expected cost of a policy [8]. As it is nontrivial to compute the gradient of the expected cost with respect to the policy, research in this area focuses on techniques for estimating it. REINFORCE [9] estimates the policy gradient using the likelihood ratio trick and subtracts a constant baseline from the rewards used in the gradient computation to reduce variance in the estimate. The Least-Squares Finite Difference (LSFD) gradient is a simple method to approximate the policy gradient and has been used with success to tune controllers for bipedal robots [10]. The Policy Gradients with Parameter based Exploration (PGPE) method samples directly in policy parameter space and uses an optimal baseline for the trajectory cost which reduces variance in the gradient estimate [11]. It has been used for learning parameters for robust standing with a humanoid robot [12]. We compare our algorithm to both LSFD and PGPE in §IV.

Natural gradient methods use the Fisher information matrix of the distribution to limit the KL divergence between trajectory distributions before and after the policy update to avoid overaggressive steps and premature convergence. The strongest theoretical advantage of this technique is that its performance is agnostic to the parameterization of the policy and, therefore, is safe to be used with arbitrary policies. This approach originated in supervised learning [13] and was first introduced to reinforcement learning by Kakade [14]. It was then extended to a natural covariant algorithm that considers the induced path-distribution manifold of the policy [15]. Natural gradient methods have been used in the context of humanoid robots [16], in actor-critic methods [17], and in the autonomous learning of motor skills [18]. These techniques, however, require that the KL divergence constraint be chosen by the user, whereas in our approach an appropriate weight on the information distance penalty is chosen automatically by the optimization procedure.

Conservative Policy Iteration (CPI) computes an updated policy as a weighted sum of a prior policy and a greedy approximator, where the weighting factor is chosen to maximize a lower bound on the policy improvement [19]. Schulman et. al. extend the policy update scheme to include more general stochastic policy classes and use it for optimizing neural network controllers for swimming, hopping, and walking tasks [20]. As noted by the authors, this approach is similar to natural gradient techniques, except that the exact KL divergence is used (instead of a quadratic approximate) and the KL divergence constraint is strictly enforced. As such, it still has the drawback that the information distance constraint must be manually tuned by the user.

Taking a dynamic programming approach, [21] uses samples to approximate a value function and then chooses a policy which maximizes the value while keeping the trajectory close to past samples. Guided Policy Search (GPS) uses model regression and trajectory optimization to guide the search away from local minima and is especially effective at learning policies with a large number of parameters [22]. This method has been used with success to learn neural network policies in systems with unknown dynamics [23]. Similar to our approach, GPS makes use of importance sampling to estimate the behavior of a policy; however, it uses a heuristic regularizer to keep policies close to past samples instead of using a more theoretically founded approach based on performance bounds.

Evaluation: The performance of the algorithm is investigated by learning control policies for two robotic systems: a car with side slip and a quadrotor platform, each set in an obstacle-ridden environment. The performance is evaluated in how efficiently the robotic system is able to avoid obstacles and safely reach a goal state. To generalize the policy to arbitrary environments, the obstacles, initial state and the goal state are sampled from a random distribution and uncertainty is added to the system dynamics. We statistically compare the performance of policies learned with our method, RWR, REPS, PGPE, and LSFD for these scenarios.

II. Background

A. Iterative Stochastic Policy Optimization (ISPO)

The goal of ISPO is to generate an optimal control policy which minimizes a cost function as shown in (1). This work considers *episodic* policy search techniques, which generate a stochastic policy on the parameter space ξ. Episodic tasks are those that end after a given number of time steps, for example execution of a finite-time control policy. These methods iteratively build a surrogate stochastic model $\pi(\xi|\nu)$ from which ξ can be sampled, with parameters $\nu \in \mathcal{V}$ where $\mathcal{V}$ is a vector space. Employing probabilistic models is a also common technique in non-linear and stochastic optimization [24], [25], [26], [27], [28], [29] for adaptive exploration of the parameter space during the optimization of complex non-convex functions. The surrogate stochastic model induces a joint density $p(\tau, \xi|\nu) = p(\tau|\xi)\pi(\xi|\nu)$ which contains the natural stochasticity of the system $p(\tau|\xi)$ and artificial control-exploration stochasticity $\pi(\xi|\nu)$ due to the surrogate model. The goal is to then minimize the expected cost

$$\mathcal{J}(\nu) \triangleq \mathbb{E}_{\tau,\xi\sim p(\cdot|\nu)}\left[J(\tau)\right]$$

iteratively until convergence. This usually corresponds to $\pi(\xi|\nu)$ shrinking to a tight peak around ξ^* or around several peaks if the distribution is multi-modal. The general framework for solving the problem is described in Algorithm 1.

Algorithm 1: Iterative Stochastic Policy Optimization (ISPO)

1: Initialize hyper-distribution ν_0, $i \leftarrow 0$
2: **while** Bound on expected cost greater than threshold **do**
3: **for** $j = 1, \ldots, M$ **do**
4: Sample trajectory $(\xi_j, \tau_j) \sim p(\cdot|\nu_i)$
5: Compute a new policy ν_{i+1} using observed costs $\{J(\tau_1), \ldots, J(\tau_M)\}$, set $i = i + 1$

end

A key step in ISPO is computing the new policy based on the observed costs of previous policies. The specific implementation of Step 5 corresponds to different algorithms such as RWR, REPS, etc. This work proposes a new algorithm for updating the policy based on minimizing an upper confidence bound on its expected future cost.

B. PAC Bounds for Iterative Policy Adaptation

The proposed technique relies on a recently derived probably approximately correct (PAC) bound [1] on the performance of a stochastic policy given past observations of performance of possibly different policies. While the bound in [1] was originally introduced for the purposes of algorithm performance analysis, in this paper we employ it for synthesis of a new algorithm. A brief overview of the bound is given next due to its central importance.

Given a prior distribution $\pi(\cdot|\nu_0)$ on control parameters and M executions based on the prior, the expected cost of a new policy $\pi(\cdot|\nu)$ is given by

$$\mathcal{J}(\nu) \triangleq \mathbb{E}_{\tau,\xi\sim p(\cdot|\nu)}[J(\tau)] = \mathbb{E}_{\tau,\xi\sim p(\cdot|\nu_0)}\left[J(\tau)\frac{\pi(\xi|\nu)}{\pi(\xi|\nu_0)}\right], \tag{4}$$

which can be approximated by the empirical cost using samples $\xi_j \sim \pi(\xi|\nu_0)$ and $\tau_j \sim p(\tau|\xi_j)$, i.e. $\mathcal{J}(\nu) \approx \frac{1}{M}\sum_{j=1}^{M}\left[J(\tau_j)\frac{\pi(\xi_j|\nu)}{\pi(\xi_j|\nu_0)}\right]$. As noted in [30], the change-of-measure likelihood ratio $\frac{\pi(\xi|\nu)}{\pi(\xi|\nu_0)}$ can be unbounded, so a standard Hoeffding or Bernstein bound becomes impractical to apply. The bound derived in [1] employs a recent robust estimation technique [31] to deal with the unboundedness of the policy adaptation. Instead of estimating the expectation $m = \mathbb{E}[X]$ of a random variable $X \in [0, \infty)$ using its empirical mean $\widehat{m} = \frac{1}{M}\sum_{j=1}^{M} X_j$, a more robust estimate can be obtained by truncating its higher moments, i.e. using $\widehat{m}_\alpha \triangleq \frac{1}{\alpha M}\sum_{j=1}^{M}\psi(\alpha X_j)$ for some $\alpha > 0$ where $\psi(x) = \log(1+x+\frac{1}{2}x^2)$. As a result, as long as x has finite variance it is possible to obtain practical bounds even if x itself is unbounded.

To obtain tight bounds, it is useful to use samples created during previous iterations of ISPO, say from L previous policies $\nu_0, \nu_1, \ldots, \nu_{L-1}$ from iterations $i = 0, \ldots, L-1$. Let $z = (\tau, \xi)$ and define $\ell_i(z, \nu) \triangleq J(\tau)\frac{\pi(\xi|\nu)}{\pi(\xi|\nu_i)}$. The expected cost based on multiple iterations can now be expressed as

$$\mathcal{J}(\nu) \equiv \frac{1}{L}\sum_{i=0}^{L-1}\mathbb{E}_{z\sim p(\cdot|\nu_i)}\ell_i(z, \nu).$$

This, again, can be approximated by the empirical mean $\mathcal{J}(\nu) \approx \frac{1}{ML}\sum_{i=0}^{L-1}\sum_{j=1}^{M}[\ell_i(z_{ij}, \nu)]$. The more robust estimate is then given by

$$\widehat{\mathcal{J}}_\alpha(\nu) \triangleq \frac{1}{\alpha LM}\sum_{i=0}^{L-1}\sum_{j=1}^{M}\psi\left(\alpha\ell_i(z_{ij}, \nu)\right).$$

The main result obtained in [1] can now be stated as follows:

Theorem 1: With probability $1 - \delta$ the expected cost of executing a stochastic policy with parameters $\xi \sim \pi(\cdot|\nu)$ is bounded according to

$$\mathcal{J}(\nu) \leq \inf_{\alpha>0}\left\{\widehat{\mathcal{J}}_\alpha(\nu) + \frac{\alpha}{2L}\sum_{i=0}^{L-1} b_i^2 e^{D_2(\pi(\cdot|\nu)||\pi(\cdot|\nu_i))} + \frac{1}{\alpha LM}\log\frac{1}{\delta}\right\}, \tag{5}$$

computed after L iterations, with M samples $z_{i1}, \ldots, z_{iM} \sim p(\cdot|\nu_i)$ obtained at iterations $i = 0, \ldots, L-1$, where $D_\beta(p||q)$ denotes the Renyii divergence between p and q. The constants b_i are such that $0 \leq J(\tau) \leq b_i$ at each iteration.

III. PAC Robust Policy Search (PROPS)

We now introduce PROPS: a new policy optimization algorithm which follows the ISPO framework described in Section II-A, at the core of which lies the minimization of the bound (5) described in Section II-B. A new policy ν_{i+1} is computed using observed costs from previous iterations by optimizing the bound (5) jointly over the "annealing coefficient" α and the policy ν. That is we solve

$$\nu^* = \arg\min_{\nu}\min_{\alpha>0}\mathcal{J}_\alpha^+(\nu), \tag{6}$$

and set $\nu_{i+1} = \nu^*$ where the value of the bound $\mathcal{J}_\alpha^+(\nu)$ is defined as

$$\mathcal{J}_\alpha^+(\nu) \triangleq \widehat{\mathcal{J}}_\alpha(\nu) + \frac{\alpha}{2L}\sum_{i=0}^{L-1} b_i^2 e^{D_2(\pi(\cdot|\nu)||\pi(\cdot|\nu_i))} + \frac{1}{\alpha LM}\log\frac{1}{\delta},$$

employing information from L past iterations with a batch of M samples per iteration.

A gradient-based method is used to find a (locally) optimal solution to (6). The details of the gradients are discussed next.

A. PAC Bound Gradients

The derivative of the bound with respect to α is given by

$$\frac{d\mathcal{J}_\alpha^+(\nu)}{d\alpha} = \frac{d\widehat{\mathcal{J}}_\alpha(\nu)}{d\alpha} + \frac{1}{2L}\sum_{i=0}^{L-1} b_i^2 e^{D_2(\pi(\cdot|\nu)||\pi(\cdot|\nu_i))} - \frac{1}{\alpha^2 LM}\log\frac{1}{\delta},$$

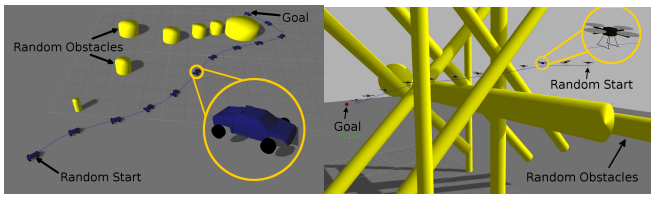


Fig. 1. The PROPS algorithm is evaluated on two challenging control tasks: a car with side-slip (left) and a quadrotor (right) navigating through an obstacle-ridden environment.

where the derivative of the robust empirical cost with respect to α is

$$\frac{d\widehat{\mathcal{J}}_\alpha(\nu)}{d\alpha} = \frac{1}{\alpha LM}\left\{\sum_{i=0}^{L-1}\sum_{j=1}^{M}\left[\psi'\left(\alpha\ell_{ij}\right)\ell_{ij} - \frac{\psi\left(\alpha\ell_{ij}\right)}{\alpha}\right]\right\},$$

using the shorthand notation $\ell_{ij} \triangleq \ell_i(z_{ij},\nu)$ and $\psi' = \frac{1+x}{1+x+\frac{1}{2}x^2}$ denoting the derivative of $\psi(x)$. The gradient of the bound with respect to the policy parameters ν is

$$\nabla_\nu \mathcal{J}_\alpha^+(\nu) = \nabla_\nu \widehat{\mathcal{J}}_\alpha(\nu) + \frac{\alpha}{2L}\sum_{i=0}^{L-1} b_i^2 e^{D_2(\pi(\cdot|\nu)||\pi(\cdot|\nu_i))}\nabla_\nu D_2\left(\pi(\cdot|\nu)||\pi(\cdot|\nu_i)\right),$$

where the gradient of the robust empirical cost with respect to ν is given by

$$\nabla_\nu \widehat{\mathcal{J}}_\alpha(\nu) = \frac{1}{LM}\sum_{i=0}^{L-1}\sum_{j=1}^{M}\psi'\left(\alpha\ell_i(z_{ij},\nu)\right)\nabla_\nu \ell_i(z_{ij},\nu)$$

and the gradient of $\ell_i(z_{ij},\nu)$ is $\nabla_\nu \ell_i(z_{ij},\nu) = \frac{J(\tau_{ij})}{\pi(\xi_{ij}|\nu_i)}\nabla_\nu \pi(\xi_{ij}|\nu)$.

IV. Empirical Evaluation

To investigate the performance of the PROPS algorithm, we apply it to two different control policy design scenarios: choosing appropriate non-linear controller gains for a car driving on a slippery surface and selecting controller gains for a quadrotor. In each scenario, the objective, quantified by the cost described in Section IV-A, is for the robot to move from a randomly generated initial state to a static goal location in the presence of process noise while minimizing control effort and avoiding obstacles placed at random locations. The stochasticity in the problem arises from the vehicle dynamics, start states, and obstacle shapes and configurations.

A. Application to Discrete-Time Stochastic Control

The classic discrete-time optimal control problem considers a discrete-time dynamical model with state $x_k \in X$, where X is an n-dimensional manifold, and control inputs $u_k \in \mathbb{R}^m$ at time $t_k \in [0,T]$ where $k = 0,\ldots,N$ indicates the time step. The system process is defined by

$$\begin{aligned} & x_{k+1} = f_k(x_k,u_k,w_k), \\ \text{subject to} \quad & g_k(x_k,u_k) \le 0, \quad g_N(x_N) \le 0, \end{aligned}$$

where f_k corresponds to the system dynamics, g_k corresponds to system constraints, like not intersecting with obstacles, and w_k is process noise. The system trajectory is thus defined as $\tau = (\{x_0,\ldots,x_N\},\{u_0,\ldots,u_{N-1}\})$. We seek an optimal control policy of the form $u_k = \Phi_k(x_k,\xi)$ parametrized using a vector ξ and depending on the current state x_k. In addition, the policy should be understood as implicitly depending on the current environment state, e.g. containing a distribution of obstacles.

Costs: The policy is computed to minimize a given cost $J(\tau)$. The cost can correspond to optimizing performance

$$J(\tau) \triangleq L_N(x_N) + \sum_{k=0}^{N-1} L_k(x_k,u_k),$$

where L_N is a terminal cost on the final state and L_k are given non-linear functions. The cost J can be arbitrary but is defined here with $L_k(x,u) = \frac{1}{2}\|u\|_R^2$ for some matrix $R > 0$ and $L_N(x,u) = \frac{1}{2}\|x - x_f\|_{Q_f}^2$ for some matrix $Q_f > 0$. The cost can also be related to safety, e.g.

$$J(\tau) = I_{g(\tau)>0},$$

where $\{g(\tau) \le 0\} \triangleq \bigwedge_{k=0}^{N-1}\{g_k(x_k,u_k) \le 0\} \wedge \{g_N(x_N) \le 0\}$.

Finally, the density of the trajectory is expressed as $p(\tau|\xi) = p(x_0)\Pi_{k=0}^{N-1}p(x_{k+1}|x_k,u_k)\delta(u_k - \Phi_k(x_k,\xi))$, where $\delta(\cdot)$ is the Dirac delta. The goal of PROPS in this setting, then, is to compute feedback control policies which minimize the PAC bound (5) on the expected value of J over this density. It should be noted that, in general, PROPS can handle any arbitrary cost function, not just the costs described for this setting.

B. Algorithm Benchmarking

We compare the performance of PROPS to that of existing ISPO algorithms mentioned in Section I, namely RWR, LSFD, PGPE, and REPS. The RWR method uses a surrogate distribution to sample different policy parameters and uses weighted maximum likelihood estimates (MLE) to compute a new surrogate distribution based on the performance of policies sampled from the previous distribution. The weights chosen for each sample are typically a soft-max function $e^{-\beta J(\tau)}$, where β is a parameter which must be tuned by the user. REPS also uses weighted MLE to iteratively update a surrogate distribution, but it chooses a specific β at each iteration which bounds the distance between the updated policy and the old policy in terms of KL divergence. LSFD samples finite-difference steps to take with respect to the parameters, and the gradient of the expected cost is then estimated using least squares. PGPE also estimates the policy gradient by sampling finite-difference steps to take with respect to the parameters, but it uses a different update step than LSFD and adaptively updates the sampling variance instead of using a variance schedule predetermined by the user.

To benchmark our algorithm against these existing methods, we perform ten training trials. In a single trial, each

algorithm is given the same samples with which to train. The samples that are used for training are randomly selected and are different between trials. After each algorithm is run until convergence for each trial, the trained policies are used on 1000 sampled contexts (i.e. starting states and obstacle positions) and a statistical test on matched pairs of costs is used to determine any performance difference between the algorithms. Collision probabilities of the policies learned with each algorithm are also compared. Parameters for LSFD, RWR, PGPE, and REPS were chosen empirically to yield the best possible result and provide a fair benchmark. All bounds are computed using $M = 300$ samples per batch, a maximum batch size of $L = 20$, and a confidence of $1 - \delta = 0.95$. In each simulation, we assume that the state of the system is perfectly measured.

C. Stochastic Policies

Gaussian distributions with diagonal covariance matrices are used for the surrogate distribution and context sampling. As such, the Renyii divergence between two Gaussian distributions will take the form

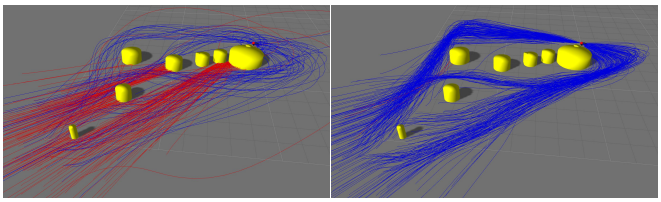
$$D_\beta\left(\mathcal{N}(\cdot|\mu_0,\Sigma_0)\|\mathcal{N}(\cdot|\mu_1,\Sigma_1)\right) = \frac{\beta}{2}\|\mu_1-\mu_0\|^2_{\Sigma_\beta^{-1}} + \frac{1}{2(1-\beta)}\log\frac{|\Sigma_\beta|}{|\Sigma_0|^{1-\beta}|\Sigma_1|^\beta},$$

where $\Sigma_\beta = (1-\beta)\Sigma_0 + \beta\Sigma_1$. Note that the divergence (and, thus, (5)) will become undefined if Σ_β is not positive-definite, which can occur if the Σ_0 has diagonal terms larger than the matching terms in Σ_1. Therefore, in the implementation of the PROPS algorithm, we impose a constraint in the optimization such that Σ_β is always positive-definite. In the other algorithms, we impose no such constraint as to not alter the algorithms, so it is possible for the PAC bound on the performance to become undefined when using these methods.

D. Car with Slip

Consider a rear-drive bicycle model with side slip operating in a planar environment among obstacles. The state of the system is defined as $x = (p_x, p_y, v, \psi, \dot{\psi}, \beta)$ where (p_x, p_y) is the position, v is the forward speed, ψ is the angle between the x-axis of the body coordinate frame and the x-axis of the world frame, β is the angle between the velocity and the x-axis of the body frame. The control inputs are $u = (\delta, u_p)$, defining the driving force u_p and steering angle δ. The car has mass m and length L. For brevity, we do not include the dynamics equations here and instead refer the reader to [32], [33] for details. We inject noise into the forces exerted on the wheels of the car to simulate terrain inconsistencies.

We employ the complex model with side slip for simulating the true vehicle response, but assume that the model is not available for control design. Instead, we construct a non-linear controller assuming a simplified car model given by $\dot{x} = \left[v\cos\psi, v\sin\psi, \frac{1}{m}u_p, v\tan\delta/L, 0, 0\right]^\top$ that does not reflect slip dynamics and other non-linear effects. Based on previous work by [34], a backstepping gyroscopic obstacle avoidance controller of the form $u = \Phi(x, \xi)$ with parameters $\xi = (k_1, k_2, k_o, k_d)$ using the simplified dynamics is employed, where Φ implicitly also depends on the randomly sampled environmental obstacles. The problem constraints $g_k(x, u) \leq 0$ include circular obstacles $\mathcal{O} \subset \mathbb{R}^2$ and control bounds defined as $\|\delta\| \leq \delta_{\max}, \|u_p\| \leq u_{p_{\max}}$. There are four gains which tune the performance of this controller. Two backstepping gains (k_1, k_2) specify the rate at which backstepping errors go to zero. The obstacle avoidance gain k_o controls the virtual velocity introduced by an obstacle, which is only detected within a radius that scales with the obstacle size based on the detection gain k_d. Figure 1 shows a visualization of the environment and Figure 2 shows resulting control policies for the driving scenario over several iterations of the optimization.



iteration #1
collision prob.=75.7%

iteration #100
collision prob.=2.0%

Fig. 2. Visualization of trajectories of the car on a static obstacle set using the PROPS policy solution at iterations 1 and 100 when starting from a random initial state. Red trajectories indicate a trajectory that collided with an obstacle while blue trajectories are collision free.

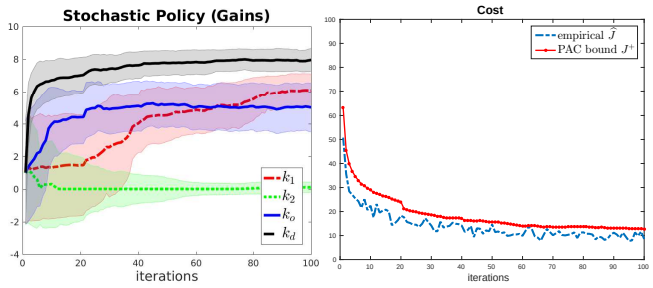
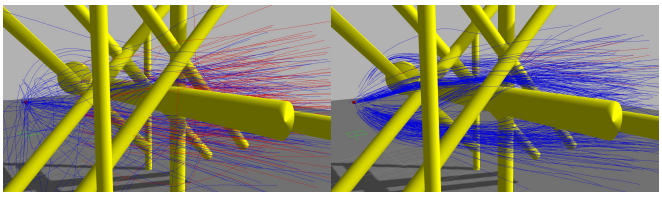


Fig. 3. Convergence of the policy and cost for the car policy learned using our algorithm. Note that the 95% confidence bound on performance (red) is tight yet not exceeded during 100 iterations of training, which indicates it is an accurate certification of the robustness of the learned policy.

E. Aerial Vehicle

Next, consider a quadrotor system operating in 3D among obstacles. The quadrotor is treated as a rigid body and takes control inputs as a thrust force u along the axial direction $e \in \mathbb{R}^3$ and torques $\tau \in \mathbb{R}^3$ in the body frame. The dynamics are a standard quadrotor model, e.g. as used in [34], with Gaussian noise injected into the dynamics as a random external force. A backstepping Lyapunov stable controller has been designed for the quadrotor to avoid obstacles and reach a goal position, with details available in [34].

In this scenario, PROPS will optimize over nine controller gains which affect the cost of trajectories taken by the quadrotor system and the probability of collision of the trajectories. Figure 4 shows a visualization of the environment and resulting control policy for the quadrotor scenario over



iteration #1
collision prob.=61.3%

iteration #100
collision prob.=5.7%

Fig. 4. Visualization of quadrotor trajectories on a static obstacle set using the PROPS solution at iterations 1 and 100 when starting from a random initial state.

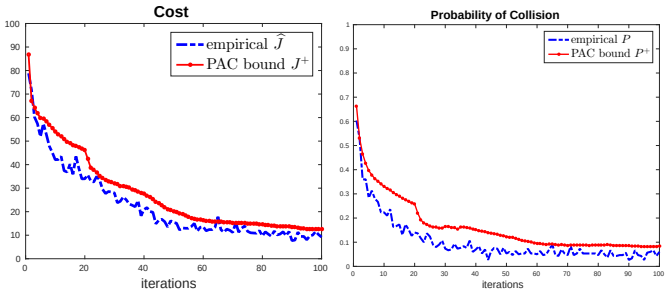


Fig. 5. Convergence of the cost and collision probability for the quadrotor policy learned using our algorithm. Note that the 95% confidence bound on performance (shown in red) is exceeded slightly only once over 100 iterations of training, which indicates it is an accurate certification of the robustness of the policy.

several iterations of the optimization.

F. Discussion

The convergence of the policies, trajectory costs, and collision probabilities when using our algorithm are shown in Figures 3 and 5 for the car and quadrotor scenarios, respectively. One can see that as the bound is minimized, the empirical cost also drops. Furthermore, the bound is a clear predictor of future performance as it is never breached by the empirical cost for the case of the car and is breached only once in the case of the quadrotor. The tightness of the bound guarantees a high performing policy with 95% confidence.

Figure 6 shows, for each algorithm, the convergence of the empirical cost, collision probability, and associated PAC bounds. PROPS directly minimizes the PAC bound on the expected cost of a policy. As a result, the costs resulting from the final distribution of parameters chosen by the algorithm has a tighter PAC bound than that of the other algorithms tested. This bound shows that future samples from the distribution have a tightly bounded cost with a high probability, certifying the policy's robustness. The probability of collision for the final distribution of each scenario is shown in Table II. One can see that PROPS consistently has a small probability of collision across all scenarios with no parameter tuning involved.

The performance of the PROPS algorithm is also compared with the other algorithms using a *matched two sample t-test*. For each of the ten training trials, we use a sample size of 1000 with matched uncertainty to compare the mean empirical cost of trajectories obtained from the final parameter distributions of each algorithm. The 95% confidence bounds on the mean difference between the empirical cost of trajectories obtained from the final converged distribution of PROPS and that of the other algorithms is shown in Table I (all p-values $< 5\times10^{-7}$, except for PGPE in the car scenario where $p = .045$). One can see that under similar conditions, PROPS performs better than other algorithms in terms of trajectory costs.

Compared to other algorithms, the performance of PROPS is not sensitive to any user-selected parameters. One need not specify the learning rate or the maximum distance between the iterating parameter distributions.

V. Conclusion

This work introduced a new algorithm for iterative stochastic policy optimization based on minimizing an upper confidence bound on the performance of a policy. We demonstrated the effectiveness of the algorithm on two challenging robot control problems and compared it to existing methods. The approach achieved small collision probabilities and tighter guarantees on expected future performance compared to other techniques. Future work will apply this technique to more general function approximators, e.g. neural networks, to learn robust general-purpose controllers instead of domain-specific non-linear controllers defined by a small set of gains. Another direction of interest is to apply the framework to more general reinforcement learning methods that exploit the recursive nature of sampled trajectories (e.g. [21], [23], [35]), for instance by establishing bounds on the quality of both policy and value function approximations.

References

[1] M. Kobilarov, "Sample complexity bounds for iterative stochastic policy optimization," in *Advances in Neural Information Processing Systems 28*, pp. 3096–3104, Curran Associates, Inc., 2015.

[2] L. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, 1984.

[3] M. P. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," pp. 388–403, 2013.

[4] S. Schaal and C. G. Atkeson, "Learning control in robotics," *Robotics & Automation Magazine, IEEE*, vol. 17, no. 2, pp. 20–29, 2010.

[5] J. Peters and S. Schaal, "Reinforcement Learning by Reward-weighted Regression for Operational Space Control," *International Conference on Machine Learning*, pp. 745–750, 2007.

[6] J. Peters, K. Mülling, and Y. Altün, "Relative Entropy Policy Search," *Artificial Intelligence*, pp. 1607–1612, 2008.

[7] J. Kober and J. Peters, *Policy Search for Motor Primitives in Robotics*, vol. 84. 2011.

[8] J. Peters and S. Schaal, "Policy Gradient Methods for Robotics," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2219–2225, 2006.

[9] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[10] N. Kohl and P. Stone, "Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion," *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 3, no. May, pp. 2619–2624, 2004.

[11] F. Sehnke, C. Osendorfer, A. Graves, and J. Peters, "Parameter-exploring Policy Gradients," *October*, no. 2005, 2009.

[12] F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, and J. Schmidhuber, "Policy Gradients with Parameter-based Exploration for Control," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5163 LNCS, no. PART 1, pp. 387–396, 2008.

[13] S.-I. Amari, "Natural gradient works efficiently in learning," *Neural computation*, vol. 10, no. 2, pp. 251–276, 1998.

	Car w/ Slip	Quadrotor
RWR	(-4.77, -4.76)	(-4.44, -4.43)
LSFD	(-6.28, -6.27)	(-27.24, -27.23)
REPS	(-5.94, -5.93)	(-2.72, -2.70)
PGPE	(-0.55, -0.54)	(-54.23, -54.22)

TABLE I

CONFIDENCE BOUNDS ON MEAN DIFFERENCE OF TRAJECTORY COSTS BETWEEN POLICIES TRAINED ON PROPS AND POLICIES TRAINED ON OTHER ALGORITHMS

	Car w/ Slip	Quadrotor
PROPS	**7.3%**	**4.9%**
RWR	14.9%	6.7%
LSFD	18.0%	13.7%
REPS	21.3%	5.5%
PGPE	10.8%	49.3%

TABLE II

EMPIRICAL COLLISION PROBABILITIES OF TRAINED POLICIES

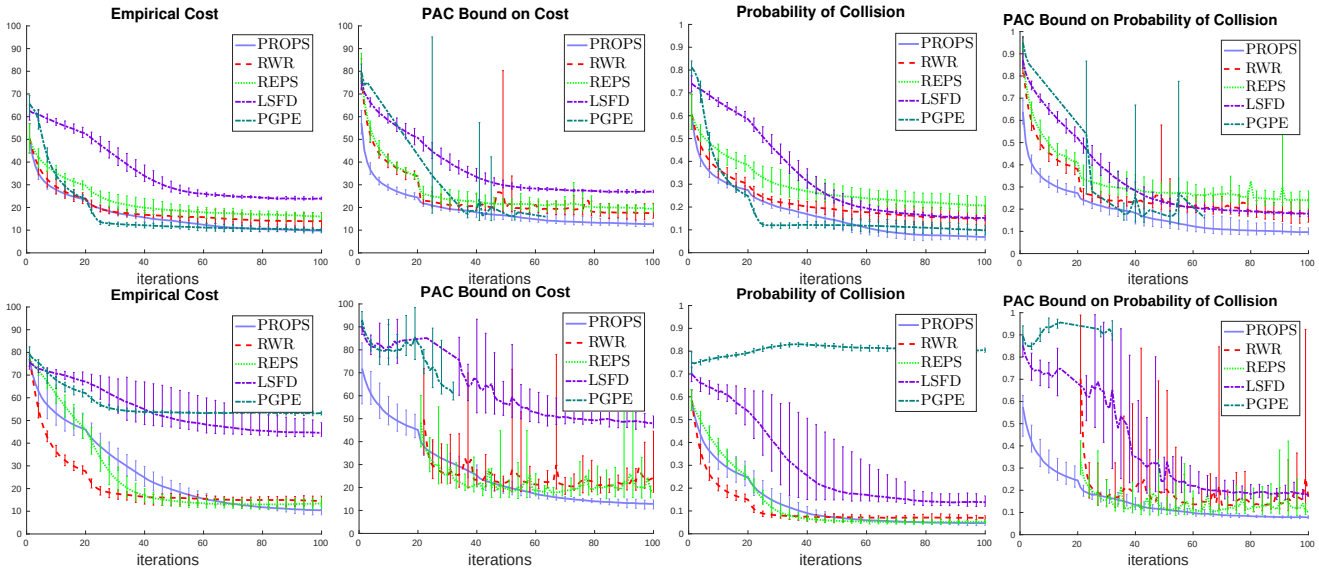


Fig. 6. Comparison of the convergence of the empirical cost, empirical collision probability, PAC bounds on the trajectory cost, and bounds on the probability of collision for the car (top) and quadrotor (bottom) scenarios during training. The data points indicate the mean performance across ten training trials, while the error bars indicate the minimum and maximum values across trials. As discussed in Section IV-C, it is possible for the PAC bounds to be undefined. When a data point and error bar is not present for a particular iteration, that indicates that the bound was undefined at that time.

[14] S. Kakade, "A natural policy gradient.," in *NIPS*, vol. 14, pp. 1531–1538, 2001.

[15] J. A. Bagnell and J. Schneider, "Covariant policy search," IJCAI, 2003.

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

[17] J. Peters, S. Vijayakumar, and S. Schaal, "Natural actor-critic," in *European Conference on Machine Learning*, pp. 280–291, Springer, 2005.

[18] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural networks*, vol. 21, no. 4, pp. 682–697, 2008.

[19] S. Kakade and J. Langford, "Approximately optimal approximate reinforcement learning," in *ICML*, vol. 2, pp. 267–274, 2002.

[20] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," *arXiv preprint arXiv:1502.05477*, 2015.

[21] R. Lioutikov, A. Paraschos, J. Peters, and G. a. Neumann, "Sample-based information-theoretic stochastic optimal control," in *Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[22] S. Levine and V. Koltun, "Guided policy search," in *Proceedings of The 30th International Conference on Machine Learning*, pp. 1–9, 2013.

[23] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Neural Information Processing Systems (NIPS)*, 2014.

[24] R. Y. Rubinstein and D. P. Kroese, *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer Science & Business Media, 2013.

[25] A. Zhigljavsky and A. Žilinskas, *Stochastic Global Optimization*, vol. 9. Springer Science & Business Media, 2007.

[26] P. Hennig and C. J. Schuler, "Entropy search for information-efficient global optimization," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 1809–1837, 2012.

[27] C. Igel, N. Hansen, and S. Roth, "Covariance matrix adaptation for multi-objective optimization," *Evolutionary computation*, vol. 15, no. 1, pp. 1–28, 2007.

[28] P. Larranaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, vol. 2. Springer Science & Business Media, 2002.

[29] M. Pelikan, D. E. Goldberg, and F. G. Lobo, "A survey of optimization by building and using probabilistic models," *Computational optimization and applications*, vol. 21, no. 1, pp. 5–20, 2002.

[30] C. Cortes, Y. Mansour, and M. Mohri, "Learning Bounds for Importance Weighting.," *Nips*, pp. 1–9, 2010.

[31] O. Catoni, "Challenging the Empirical Mean and Empirical Variance: A Deviation Study," *Annales de l'institut Henri Poincare (B) Probability and Statistics*, vol. 48, no. 4, pp. 1148–1185, 2012.

[32] Y. Yoon, J. Shin, H. J. Kim, Y. Park, and S. Sastry, "Model-predictive active steering and obstacle avoidance for autonomous ground vehicles," *Control Engineering Practice*, vol. 17, no. 7, pp. 741 – 750, 2009.

[33] A. Tahirovic and G. Magnani, *Passivity-Based Model Predictive Control for Mobile Vehicle Motion Planning*. Springer, 2013.

[34] G. Garimella, M. Sheckells, and M. Kobilarov, "A stabilizing gyroscopic obstacle avoidance controller for underactuated systems," in *IEEE International Conference on Decision and Control*, 2016.

[35] J. Fu and U. Topcu, "Probably approximately correct mdp learning and control with temporal logic constraints," in *Proceedings of Robotics: Science and Systems*, (Berkeley, USA), July 2014.