PAC-Bayes Control: Synthesizing Controllers that Provably Generalize to Novel Environments

Anirudha Majumdar
Department of Mechanical and Aerospace Engineering
Princeton University
ani.majumdar@princeton.edu

Maxwell Goldstein
Department of Mathematics
Princeton University
mag4@princeton.edu

Abstract: Our goal is to synthesize controllers for robots that provably generalize well to novel environments given a dataset of example environments. The key technical idea behind our approach is to leverage tools from *generalization theory* in machine learning by exploiting a precise analogy (which we present in the form of a reduction) between robustness of controllers to novel environments and generalization of hypotheses in supervised learning. In particular, we utilize the *Probably Approximately Correct (PAC)-Bayes* framework, which allows us to obtain upper bounds (that hold with high probability) on the expected cost of (stochastic) controllers across novel environments. We propose control synthesis algorithms that explicitly seek to minimize this upper bound. The corresponding optimization problem can be solved efficiently using *convex* optimization (Relative Entropy Programming in particular) in the setting where we are optimizing over a finite control policy space. In the more general setting of continuously parameterized controllers, we minimize this upper bound using stochastic gradient descent. We present examples of our approach in the context of obstacle avoidance control with depth measurements. Our simulated examples demonstrate the potential of our approach to provide strong generalization guarantees on controllers for robotic systems with continuous state and action spaces, nonlinear dynamics, and partially observable state via sensor measurements.

1 Introduction

Imagine an unmanned aerial vehicle that successfully navigates a thousand different obstacle environments or a robotic manipulator that successfully grasps a million objects in our dataset. How likely are these systems to succeed on a novel (i.e., previously unseen) environment or object? How can we explicitly synthesize controllers that provably generalize well to such environments or objects? Current approaches for designing controllers for robotic systems either do not provide such guarantees on generalization or provide guarantees only under extremely restrictive assumptions (e.g., strong assumptions on the geometry of a novel environment [1, 2, 3, 4]).

The goal of this paper is to develop an approach for synthesizing controllers for robotic systems that provably generalize well with high probability to novel environments given a dataset of example environments. The key conceptual idea for enabling this is to exploit a precise analogy between robustness of controllers to novel environments and generalization in supervised learning. This analogy allows us to translate techniques for learning hypotheses with generalization guarantees in the supervised learning setting into techniques for synthesizing control policies for robot tasks with performance guarantees on novel environments. In particular, here we leverage PAC-Bayes theory (Probably Approximately Correct Bayes) [5], which provides some of the tightest known generalization bounds for classical supervised learning approaches [6, 7] and has also very recently been used to promote and explain generalization of deep neural nets [8, 9, 10].

Statement of Contributions: To our knowledge, the results in this paper constitute the first attempt to provide generalization guarantees for learning-based controllers for robotic systems with continuous state and action spaces, nonlinear dynamics, and partially observable state operating in novel environments. To this end, this paper makes three primary contributions. First, we provide a *reduction* that allows us to translate generalization bounds from supervised learning to generalization bounds for controllers. We apply this reduction to translate PAC-Bayes bounds to the control setting we consider here (Section 4). Second, we propose solution algorithms for minimizing the regularized cost

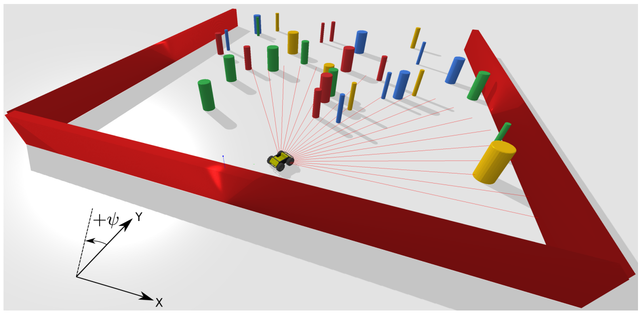

Figure 1: We demonstrate our approach on the problem of synthesizing reactive obstacle avoidance controllers for a differential-drive robot model equipped with a depth sensor. Our approach provides strong guarantees on the performance of the synthesized controllers on novel environments even with a small number of training environments (e.g., guaranteed expected collision-free traversal rate of 82.2% with 100 training environments).

functions specified by PAC-Bayes theory in order to synthesize controllers with generalization guarantees (Section 5). In the setting where we are optimizing over a finite policy space (Section 5.1), we show how to solve the corresponding optimization problem using *convex* optimization (*Relative Entropy Programs (REPs)* in particular). In the more general setting of continuously parameterized controllers (Section 5.2), we rely on stochastic gradient descent to perform the optimization. Third, we demonstrate our approach (Section 6) on the problem of synthesizing reactive obstacle avoidance controllers for a differential-drive robot model with 20-dimensional depth measurements (Figure 1). Our simulation results demonstrate that we are able to obtain strong performance guarantees even with a relatively small number of training environments. We compare the bounds obtained from PAC-Bayes theory with exhaustive sampling to illustrate the tightness of the bounds.

1.1 Related Work

One approach for synthesizing controllers with guaranteed performance is to leverage robust control techniques (e.g., H-infinity control [11] or chance-constrained programming [12, 13, 14, 15]). However, such techniques typically require an explicit description of the uncertainty affecting the system. While uncertainty models for the robot's dynamics (or measurement model) can often be obtained via system identification, assuming an uncertainty model for the environment (e.g., a distribution over all possible environment geometries) is unrealistic. One way to address this is to assume that a novel environment satisfies conditions that allow a real-time planner to *always* succeed. For example, in the context of navigation, this constraint could be satisfied by hand-coding emergency maneuvers (e.g., stopping maneuvers or loiter circles) [1, 2, 3] or assuming that the environment satisfies certain geometric conditions (e.g., large separation between obstacles) that allow for safe navigation [4]. However, such conditions are rarely satisfied by real-world environments. Moreover, such conditions are domain specific; it is not clear how one would specify such constraints for other applications (e.g., grasping).

Another conceptually appealing approach for synthesizing controllers with guaranteed performance on a priori unknown environments is to model the problem as a Partially Observable Markov Decision Process (POMDP) [16], where the environment is part of the (partially observed) state of the system [17]. Computational considerations aside, such an approach is made infeasible by the need to specify a distribution over environments the robot might encounter. Unfortunately, specifying such a distribution over real-world environments is extremely challenging. Thus, many approaches (including ours) assume that we only have *indirect* access to the true underlying distribution over environments in the form of examples. For example, Richter et al. [17, 18] propose an approximation to the POMDP framework in the context of navigation by learning to predict future collision probabilities from past data. The work on deep-learning based approaches for manipulation represents another prominent set of techniques where interactions with example environments (objects in this case) are used to learn control policies [19, 20, 21, 22, 23]. While the approaches mentioned above have led to impressive empirical demonstrations, it is very challenging to guarantee that such methods will perform well on novel environments (especially when a limited number of training examples are available, as is often the case for robotics applications).

The primary theoretical framework we utilize in this paper is PAC-Bayes generalization theory [5], which provides some of the tightest known generalization bounds for classical supervised learning problems [6, 7] and has recently been applied to explain and promote generalization in deep learning [8, 9, 10]. PAC-Bayes theory has also been applied to learn control policies for Markov Decision Processes (MDPs) with provable sample complexity bounds [24, 25]. However, we note that the focus of our work is quite different from the work on PAC-Bayes MDP bounds (and the more general framework of PAC MDP bounds [26, 27, 28]), which consider the standard reinforcement learning setup where a control policy must be learned through multiple interactions with a given MDP (with unknown transition dynamics and/or rewards). In contrast, here we focus on *zero-shot* generalization to a novel environment. In other words, a controller learned from examples of different environments must immediately perform well on a new one (i.e., without further exploratory interactions with the new environment). We further note that [24] considers finite state and action spaces along with policies that depend on full state feedback while [25] relaxes the assumption on finite state spaces but retains the other modeling assumptions. In contrast, we target systems with continuous state and action spaces and synthesize control policies that rely on rich sensory inputs.

On the algorithmic front, we make significant use of Relative Entropy Programs (REPs) [29], which constitute a rich class of efficiently solvable convex optimization problems (containing linear and second-order cone programs [30] as special cases). REPs are problems in which a linear functional of the decision variables is minimized subject to linear constraints and conic constraints given by a *relative entropy cone* (see [29] for a more thorough introduction). Importantly for us, REPs can handle constraints of the form $\mathrm{KL}(p\|q) \leq c$, where p and q are decision vectors and $\mathrm{KL}(\cdot\|\|\cdot)$ represents the Kullback-Leibler divergence. This will allow us to use REPs to synthesize controllers using the PAC-Bayes framework in the setting where we are optimizing over a finite set of policies.

2 Problem Formulation

Notation: We use $v[i]$ to refer to the i-th component of a vector $v \in \mathbb{R}^n$. We denote by $\mathbb{R}^n_+$ the set of elementwise nonnegative vectors in $\mathbb{R}^n$ and use $\odot$ to denote element-wise multiplication.

We assume that the robot's dynamics are described by a discrete-time system $x(t+1) = f(x(t), u(t); E)$, where $t \in \mathbb{Z}_+$ is the time index, $x(t) \in \mathcal{X}$ is the state, $u(t) \in \mathcal{U}$ is the control input, and E is the environment that the robot operates in. We use the term "environment" here broadly to refer to any factors that are external to the robot (e.g., an obstacle field that a mobile robot is navigating, external disturbances, or an object that a manipulator is attempting to grasp).

Let $\mathcal{E}$ denote the space of all possible environments. We assume that there is an underlying distribution $\mathcal{D}$ over $\mathcal{E}$ from which environments are drawn. Importantly, we *do not* assume that we have explicit descriptions of $\mathcal{E}$ or $\mathcal{D}$. Instead, we only assume indirect access to $\mathcal{D}$ in the form of a data set $S = \{E_1, \ldots, E_N\}$ of N training environments drawn i.i.d. from $\mathcal{D}$.

Let $g : \mathcal{X} \times \mathcal{E} \rightarrow \mathcal{Y}$ denote the robot's sensor mapping from a state x and an environment E to an observation $y = g(x; E) \in \mathcal{Y}$. Let $\pi : \mathcal{Y} \rightarrow \mathcal{U}$ denote a control policy that maps sensor measurements to control inputs. Note that this is a very general model that captures policies that depend on *histories* of sensor measurements (by augmenting x to keep track of histories of states and letting $\mathcal{Y}$ denote the space of histories of measurements).

We assume that the robot's desired behavior is encoded through a cost function. In particular, let $r_\pi : \mathcal{E} \rightarrow (\mathcal{X} \times \mathcal{U})^T$ denote the function that "rolls out" the system with control policy π, i.e., r_π maps an environment E to the state-control trajectory obtained by applying π (up to a time horizon T). We will assume that the environment captures all sources of stochasticity (including random initial conditions) and the rollout function for a *particular* environment is thus deterministic. We then let $C(r_\pi; E)$ denote the cost incurred by control policy π when operating in environment E over a time horizon T. We assume that the cost $C(r_\pi; E)$ is bounded and will assume (without further loss of generality) that $C(r_\pi; E) \in [0, 1]$.

The primary assumption we make in this work is the following.

Assumption 1. *Given any control policy π, we can compute the cost $C(r_\pi; E_i)$ for the training environments $E_1, \ldots, E_N$.*

This assumption is satisfied if one can simulate the robot's operation in the environments $E_1, \ldots, E_N$. We note that computational considerations aside, we do not make any restrictions on the dynamics f or the sensor mapping g beyond the ability to simulate them. The models that our approach can handle are thus extremely rich in principle (e.g., nonlinear or hybrid dynamics, sensor models involving raycasting or simulated vision, etc.). Another possibility for satisfying Assumption 1 is to run the controller π on the hardware system itself in the given environments. This may be feasible for problems such as grasping, which are not safety-critical in nature. In such cases, our approach does not require models of the dynamics, sensor mapping, or the rollout function.

Goal: Our goal is to design a control policy that minimizes the expected value of the cost C across environments:

$$\min_{\pi \in \Pi} \; C_{\mathcal{D}}(\pi) := \underset{E \sim \mathcal{D}}{\mathbb{E}} [C(r_\pi; E)]. \tag{1}$$

In this work, it will be useful to consider a more general setting where we choose a *distribution* P over the control policy space Π (since the PAC-Bayes bounds we will use will assume this setting). Our goal is then to solve the following optimization problem, which we refer to as $\mathcal{OPT}$:

$$C^\star := \min_{P \in \mathcal{P}} C_{\mathcal{D}}(P) := \min_{P \in \mathcal{P}} \mathop{\mathbb{E}}_{E \sim \mathcal{D}} \mathop{\mathbb{E}}_{\pi \sim P} [C(r_\pi; E)], \qquad (\mathcal{OPT})$$

where $\mathcal{P}$ denotes the space of probability distributions over Π. Note that the outer expectation here is taken with respect to the *unknown* distribution $\mathcal{D}$. This is the primary challenge for tackling $\mathcal{OPT}$.

3 Background: PAC-Bayes Theory in Supervised Learning

The primary technical framework we leverage in this paper is PAC-Bayes theory. Here we provide a brief overview of the key results from PAC-Bayes theory in the context of supervised learning. Let $\mathcal{Z}$ be an input space and $\mathcal{Z}'$ be a set of labels. Let $\mathcal{D}$ be the (unknown) true distribution on $\mathcal{Z}$. Let $\mathcal{H}$ be a hypothesis class consisting of functions $h_w : \mathcal{Z} \to \mathcal{Z}'$ parameterized by $w \in \mathbb{R}^d$ (e.g., neural networks parameterized by weights w). Let $l : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}$ be a loss function[1]. We will denote by $\mathcal{P}$ the space of probability distributions on the parameter space $\mathbb{R}^d$. Informally, we will refer to distributions on $\mathcal{H}$ when we mean distributions over the underlying parameter space.

PAC-Bayes theory then applies to learning algorithms that output a *distribution* over hypotheses. Specifically, the PAC-Bayes framework applies to learning algorithms with the following structure: (1) Choose a *"prior" distribution* $P_0 \in \mathcal{P}$ *before* observing any data; (2) observe training data $S = \{z_i\}_{i=1}^N$ and choose a *"posterior" distribution* $P \in \mathcal{P}$. It is important to note that the posterior distribution P *need not* be the Bayesian posterior. PAC-Bayes theory applies to *any* distribution P.

Let us denote the training loss associated with the posterior distribution P as $l_S(P) := \frac{1}{N} \sum_{z \in S} \mathop{\mathbb{E}}_{w \sim P} [l(h_w; z)]$ and the true expected loss as $l_{\mathcal{D}}(P) := \mathop{\mathbb{E}}_{z \sim \mathcal{D}} \mathop{\mathbb{E}}_{w \sim P} [l(h_w; z)]$. The following theorem is the primary result from PAC-Bayes theory[2].

Theorem 1 (PAC-Bayes Bound for Supervised Learning [5, 31])**.** *For any* $\delta \in (0, 1)$*, with probability at least* $1 - \delta$ *over samples* $S \sim \mathcal{D}^N$*, the following inequalities hold:*

$$\underbrace{l_{\mathcal{D}}(P)}_{\textit{True loss}} \leq \mathrm{KL}^{-1}\Big(l_S(P) \| \frac{\mathrm{KL}(P \| P_0) + \log(\frac{2\sqrt{N}}{\delta})}{N}\Big) \leq \underbrace{l_S(P)}_{\textit{Training loss}} + \underbrace{\sqrt{\frac{\mathrm{KL}(P \| P_0) + \log(\frac{2\sqrt{N}}{\delta})}{2N}}}_{\textit{"Regularizer"}}. \qquad (2)$$

Here, the KL inverse is defined as: $\mathrm{KL}^{-1}(p \| c) := \sup\{q \in [0, 1] \mid \mathrm{KL}(p \| q) \leq c\}$ (see Section 1 of the Appendix for more details). Theorem 1 provides two upper bounds on the true expected loss $l_{\mathcal{D}}(P)$. The second follows from the first by applying the well-known upper bound for the KL inverse: $\mathrm{KL}^{-1}(p \| c) \leq p + \sqrt{c/2}$ and is easier to work with for the purpose of optimization. It provides a strategy for choosing a distribution P over hypotheses by minimizing a combination of the training loss and a "regularizer".

4 PAC-Bayes Controllers

We now describe our approach for adapting the PAC-Bayes framework in order to tackle the control synthesis problem $\mathcal{OPT}$ and synthesize (stochastic) control policies with guaranteed expected performance across novel environments. Our key idea for doing this is to exploit a precise analogy between the supervised learning setting from Section 3 and the control synthesis setting described in Section 2. Table 1 presents this relationship.

One can think of the relationship in Table 1 as providing a *reduction* [32] from the control synthesis problem $\mathcal{OPT}$ to a supervised learning problem. We are provided input data in the form of a data set of example environments. Choosing a "hypothesis" corresponds to choosing a control policy π (since the rollout function r_π is determined by π). A "hypothesis" maps an environment E to a "label", corresponding to the state-control trajectory obtained by applying π on E. This "label" incurs a loss $C(r_\pi; E)$.

[1]Note that we are considering a slightly restricted form of the supervised learning problem (which is sufficient for our needs here) where each input $z \in \mathcal{Z}$ has only one correct label $z' \in \mathcal{Z}'$.

[2]The bound we state here is a well-known bound due to Maurer [31] that improves slightly upon the original PAC-Bayes bounds [5]. This bound holds when costs are bounded within $[0, 1]$ (as assumed here) and $N \geq 8$.

Supervised Learning			Control Synthesis	
Input data	$z \in \mathcal{Z}$		Environment	$E \in \mathcal{E}$
Hypothesis	$h_w : \mathcal{Z} \to \mathcal{Z}'$	$\longleftarrow$	Rollout function	$r_\pi : \mathcal{E} \to (\mathcal{X} \times \mathcal{U})^H$
Loss	$l(h_w; z)$		Cost	$C(r_\pi; E)$

Table 1: A reduction from the control synthesis problem we consider here to supervised learning.

We can use this reduction to translate the PAC-Bayes theorem for supervised learning (Theorem 1) to the control setting. We assume that the space Π of control policies is parameterized by $w \in \mathbb{R}^d$. This in turn parameterizes rollout functions. With a slight abuse of notation, we will refer to rollout functions r_w instead of r_π (with the understanding that w is the parameter corresponding to π).

Let P_0 be a "prior" distribution over the parameter space $\mathbb{R}^d$. The prior can be used to encode domain knowledge, but need not be "true" in any Bayesian sense (i.e., bounds will hold for any prior). Let P be a (possibly data-dependent) "posterior". Following the notation from Section 2, we denote the true expected cost across environments by $C_\mathcal{D}(P)$. We denote the cost on the training environments as

$$C_S(P) := \frac{1}{N} \sum_{E \in S} \mathop{\mathbb{E}}_{w \sim P}[C(r_w; E)]. \tag{3}$$

The following theorem then allows us to upper bound $C_\mathcal{D}(P)$.

Theorem 2 (PAC-Bayes Bound for Control Policies)**.** *For any $\delta \in (0,1)$, with probability at least $1-\delta$ over sampled environments $S \sim \mathcal{D}^N$, the following inequality holds:*

$$\underbrace{C_\mathcal{D}(P)}_{\textit{True expected cost}} \leq \mathrm{KL}^{-1}\Big(C_S(P) \| \frac{\mathrm{KL}(P\|P_0) + \log(\frac{2\sqrt{N}}{\delta})}{N}\Big) \leq \underbrace{C_S(P)}_{\textit{Training cost}} + \underbrace{\sqrt{\frac{\mathrm{KL}(P\|P_0) + \log(\frac{2\sqrt{N}}{\delta})}{2N}}}_{\textit{"Regularizer"}}. \tag{4}$$

Proof. The proof follows immediately from Theorem 1 given the reduction in Table 1. □

The left hand side of inequality (4) is the cost function $C_\mathcal{D}(P)$ of the optimization problem $\mathcal{OPT}$. Theorem 2 thus provides an upper bound (that holds with probability $1-\delta$) on the true expected performance across environments of any controller distribution P in terms of the loss on the training environments in $S = \{E_i\}_{i=1}^N$ and a "regularizer". Our approach for choosing P is to minimize this upper bound. Algorithm 1 outlines the steps involved in our approach. We note that while P is chosen by optimizing $C_{\text{PAC}}(P)$, the final upper bound $C^\star_{\text{bound}}$ on $C_\mathcal{D}(P)$ is not computed as $C_{\text{PAC}}(P^\star_{\text{PAC}})$. While this is a valid bound, a tighter bound is provided by the KL inverse term in (4). The Appendix (Section 1) shows how to compute the KL inverse using a simple REP.

Algorithm 1 PAC-Bayes Control Synthesis

```
1: Fix prior distribution $P_0 \in \mathcal{P}$ over controllers
2: Inputs: $S = \{E_1, \dots, E_N\}$: Training environments, $\delta$: Probability threshold
3: Outputs:
4: $P^\star_{\text{PAC}} = \operatorname*{argmin}_{P \in P} C_{\text{PAC}}(P) := \frac{1}{N}\sum_{E \in S} \mathop{\mathbb{E}}_{w \sim P}[C(r_w; E)] + \sqrt{\frac{\mathrm{KL}(P\|P_0)+\log(\frac{2\sqrt{N}}{\delta})}{2N}}$
5: $C^\star_{\text{bound}} := \mathrm{KL}^{-1}\Big(C_S(P^\star_{\text{PAC}}) \| \frac{\mathrm{KL}(P^\star_{\text{PAC}}\|P_0)+\log(\frac{2\sqrt{N}}{\delta})}{N}\Big)$
```

5 Computing PAC-Bayes Controllers

We now describe how to tackle the optimization problem in Algorithm 1 for minimizing the upper bound on the true expected cost. We first show how to perform this optimization using Relative Entropy Programming in the setting where the control policy space Π is finite (Section 5.1). We then tackle the more general setting where Π is continuously parameterized (Section 5.2).

5.1 Finite Control Policy Space

Let the space of policies be $\Pi = \{\pi_1, \dots, \pi_L\}$. Our goal is then to optimize a *discrete* probability distribution P (with corresponding probability vector p) over the space Π. Thus, $p[j]$ denotes the probability assigned to controller π_j. Define a matrix $\hat{C}$ of costs, where each element $\hat{C}[i,j] :=$

$C(r_{\pi_j}; E_i)$ corresponds to the cost incurred on environment $E_i \in S$ by controller $\pi_j \in \Pi$ (recall that Assumption 1 implies that we can compute each $\hat{C}[i, j]$). The training cost $C_S(P)$ is then:

$$\frac{1}{N} \sum_{E \in S} \underset{\pi \sim P}{\mathbb{E}}[C(r_\pi; E)] = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{L} \hat{C}[i, j] p[j] := \bar{C}p. \tag{5}$$

We note that finding a vector p that minimizes the training cost corresponds to a *Linear Program*.

Minimizing the PAC-Bayes upper bound $C_{\text{PAC}}(P)$ corresponds to solving the following optimization problem (see Section 2 of the Appendix for a detailed derivation):

$$\min_{p \in \mathbb{R}^L, \tau, \lambda} \quad \tau \tag{6}$$

$$\text{s.t.} \quad \lambda^2 \geq \frac{\text{KL}(p \| p_0) + \log(\frac{2\sqrt{N}}{\delta})}{2N}, \ \lambda = \tau - \bar{C}p, \ \lambda \geq 0, \ 0 \leq p \leq 1, \ \sum_j p[j] = 1.$$

Our key observation here is that for a *fixed* $\lambda = \lambda_0$, the above problem is a Relative Entropy Program (REP) since it consists of minimizing a linear cost function subject to linear equality and inequality constraints and an additional inequality constraint of the form $\text{KL}(p \| p_0) \leq$ constant. We can thus solve this problem very efficiently (see [29, 33] for details on computational complexity).

We note that $\lambda \in [0, 1]$ since $\lambda = \tau - \bar{C}p$, where $\tau \in [0, 1]$ (because τ upper bounds the true expected cost) and $\bar{C}p \in [0, 1]$ (recall that we assumed that costs are bounded in $[0, 1]$). In order to solve problem (6) to global optimality, we can thus simply search over $\lambda \in [0, 1]$ (e.g., using a bisection search) to find the λ that leads to the lowest optimal value for the corresponding REP.

5.2 Continuously-Parameterized Control Policy Space

We now consider policies π_w parameterized by the vector $w \in \mathbb{R}^d$ (e.g., neural net weights). We will consider stochastic policies defined by Gaussian distributions $w \sim \mathcal{N}(\mu, \Sigma)$ with diagonal covariance $\Sigma = \text{diag}(s)$ (with $s \in \mathbb{R}^d_+$). We use the shorthand $\mathcal{N}_{\mu,s} := \mathcal{N}(\mu, \text{diag}(s))$. Using Gaussians makes computations easier since we can express the KL divergence between Gaussians in closed form. We can then apply Algorithm 1 and choose μ, s to minimize the PAC-Bayes bound $C_{\text{PAC}}(\mathcal{N}_{\mu,s})$. For this to be a practical algorithm, there are two primary issues we need to address.

First, in order to minimize the bound $C_{\text{PAC}}(\mathcal{N}_{\mu,s})$, one would like to apply gradient-based methods. However, the cost function may not be a differentiable (or even continuous) function of the parameters w. For example, in the case of designing obstacle avoidance controllers, a natural (but discontinuous) cost function is the one that assigns a cost of 1 if the robot collides (and 0 otherwise). To tackle this issue, we employ a differentiable surrogate for the cost function during optimization (note that the final bound is still evaluated for the original cost function).

The second challenge is the fact that computing the training cost $C_S(\mathcal{N}_{\mu,s})$ requires computing $\underset{w \sim \mathcal{N}_{\mu,s}}{\mathbb{E}}[C(r_w; E)]$. For most realistic settings, this expectation cannot be computed in closed form. We address this issue in a manner similar to [8]. In particular, in order to optimize μ and s using gradient descent, we use the following unbiased estimator of $C_S(\mathcal{N}_{\mu,s})$:

$$\frac{1}{N} \sum_{E \in S} C(r_{\mu + \sqrt{s} \odot \xi}; E), \quad \xi \sim \mathcal{N}_{0, I_d}. \tag{7}$$

In other words, in each gradient step we use an i.i.d. sample of ξ and compute gradients of (7) with respect to μ and s. At the end of the optimization procedure, we fix the optimal $\mu^\star$ and $s^\star$ and estimate the training cost $C_S(P) = C_S(\mathcal{N}_{\mu^\star, s^\star})$ by producing a large number of samples $w_1, \ldots, w_L$ drawn from $\mathcal{N}_{\mu^\star, s^\star}$: $\hat{C}_S(\mathcal{N}_{\mu^\star, s^\star}) := \frac{1}{NL} \sum_{E \in S} \sum_{i=1}^{L} C(r_{w_i}; E)$. We can then use a sample convergence bound (see [34]) to bound the error between $\hat{C}_S(\mathcal{N}_{\mu^\star, s^\star})$ and $C_S(\mathcal{N}_{\mu^\star, s^\star})$. In particular, the following bound is an application of the relative entropy version of the Chernoff bound for random variables (i.e., costs) bounded in $[0, 1]$ and holds with probability $1 - \delta'$:

$$C_S(\mathcal{N}_{\mu^\star, s^\star}) \leq \bar{C}_S(\mathcal{N}_{\mu^\star, s^\star}; L, \delta') := \text{KL}^{-1}(\hat{C}_S(\mathcal{N}_{\mu^\star, s^\star}) \| \frac{1}{L} \log(\frac{2}{\delta'})). \tag{8}$$

Combining inequalities (4) and (8) using the union bound, we see that the following bound holds with probability at least $1 - \delta - \delta'$:

$$C_{\mathcal{D}}(\mathcal{N}_{\mu^\star, s^\star}) \leq C^\star_{\text{bound}} := \text{KL}^{-1}\left(\bar{C}_S(\mathcal{N}_{\mu^\star, s^\star}; L, \delta') \| \frac{\text{KL}(\mathcal{N}_{\mu^\star, s^\star} \| P_0) + \log(\frac{2\sqrt{N}}{\delta})}{N}\right). \tag{9}$$

This is the final version of our bound. Our approach is summarized in Algorithm 1 in the Appendix.

6 Example: Reactive Obstacle Avoidance Control

In this section, we demonstrate our approach on the problem of synthesizing reactive obstacle avoidance controllers for a ground vehicle model equipped with a depth sensor. We first consider a finite policy space Π and leverage the REP-based framework described in Section 5.1 to provide guarantees on the performance of the controller across novel obstacle environments. We then consider continuously parameterized policies and apply the approach from Section 5.2.

Dynamics. A pictorial depiction of the robot is provided in Figure 1. The state is $[x, y, \psi]$, where x and y are the x and y positions of the vehicle respectively, and ψ is the yaw angle. We model the system using the (nonlinear) equations of motion of a differential drive vehicle with two control inputs u_l and u_r corresponding to the left and right wheel speeds respectively (see Section 4 of Appendix for details). We set $u_l = u_0 - u_{\text{diff}}, \quad u_r = u_0 + u_{\text{diff}}$ with u_0 chosen to ensure that the robot has a fixed speed $v_0 = 2.5$m/s. We limit the turning rate by constraining $u_{\text{diff}} \in [-u_0/2, u_0/2]$. The system is simulated as a discrete-time system with time-step $\Delta t = 0.05$s.

Obstacle environments. A typical obstacle environment is shown in Figure 1 and consists of N_{obs} cylinders of varying radii along with three walls that bound the environment between $x \in [-5, 5]$m and $y \in [0, 10]$m. Environments are generated by first sampling the integer N_{obs} uniformly between 20 and 40, and then independently sampling the x-y positions of the cylinders from a uniform distribution over the ranges $x \in [-5, 5]$m and $y \in [2, 10]$m. The radius of each obstacle is sampled uniformly from the range $[0.05, 0.2]$m. The robot's state is always initialized at $[x, y, \psi] = [0, 1, 0]$.

Obstacle Avoidance Controllers. We assume that the robot is equipped with a depth sensor that provides distances $y[i]$ along 20 rays in the range $\theta[i] \in [-\pi/3, \pi/3]$ radians (+ve is clockwise) up to a sensing horizon of 5m (as shown in Figure 1). A given sensor measurement y thus belongs to the space $\mathcal{Y} = \mathbb{R}^{20}$. Let $\hat{y} = 1/y \in \mathbb{R}^{20}$ be the inverse distance vector computed by taking an element-wise reciprocal of y. We then choose u_{diff} as the following dot product:

$$u_{\text{diff}} = K \cdot \hat{y}. \tag{10}$$

An example of $K \in \mathbb{R}^{20}$ is (see Figure 1 in the Appendix for a plot):

$$K[i] = \begin{cases} (y_0/x_0)(x_0 - \theta[i]) & \text{if } \theta[i] \geq 0, \\ (y_0/x_0)(-x_0 - \theta[i]) & \text{if } \theta[i] < 0. \end{cases} \tag{11}$$

For $\theta[i] > 0$, $K[i]$ is a linear function of $\theta[i]$ with x- and y-intercepts equal to x_0 and y_0. This is reflected about the origin for $\theta[i] < 0$. Intuitively, this corresponds to a simple reactive controller that computes a weighted combination of inverse distances in order to turn away from obstacles that are close. Simple reactive controllers of this kind have been shown to be quite effective in practice [35, 36, 37, 38], but can often be challenging to tune by hand in order to achieve good expected performance across *all* environments. We tackle this challenge by applying the PAC-Bayes control framework proposed here.

Results (finite policy space). To obtain a finite policy space, we choose $L = 50$ different K's of the form (11) by choosing different x and y intercepts x_0 and y_0. In particular, (x_0, y_0) is chosen by discretizing the space $[0.1, 5.0] \times [0, 10.0]$ into 5 values for x_0 and 10 values for y_0. Our policy space is thus $\Pi = \{\pi_1, \dots, \pi_L\}$, where each controller π_i corresponds to a particular choice of K.

We consider a time horizon of $T = 100$ and assign a cost of 1 if the robot collides with an obstacle during this period (and 0 otherwise). We choose a uniform prior over the policy space Π and apply the REP framework from Section 5.1. The PyBullet package [39] is used to simulate the dynamics and depth sensor; we use these simulations to compute the elements of the cost matrix $\bar{C}$ (ref. Section 5.1). Each simulation takes ~ 0.01s to execute in our implementation (note that the computation of the different elements of $\bar{C}$ can be entirely parallelized). Given the matrix $\bar{C}$ with 100 sampled environments, each REP (corresponding to a fixed value of λ in Problem (6)) takes ~ 0.05s to solve using the CVXPY package [40] and the SCS solver [41]. We discretize the interval $[0, 1]$ into 100 values to find the optimal λ. Complete code for this implementation is available online (see Section 6 of Appendix).

Table 2 presents the upper bound $C^{\star}_{\text{bound}}$ on the true expected cost of the PAC-Bayes controller $P^{\star}_{\text{PAC}}$ (ref. Algorithm 1) for different sample sizes N with $\delta = 0.01$. The table also presents an estimate of the true expected cost $C_{\mathcal{D}}(P^{\star}_{\text{PAC}})$ obtained by sampling 10^5 environments. As the table illustrates, the PAC-Bayes bound provides strong guarantees even for relatively small sample sizes. For example,

N (# of training environments)	100	500	1000	10000
PAC-Bayes bound ($C^\star_{\text{bound}}$)	0.178	0.135	0.121	0.096
True expected cost (estimate)	0.087	0.084	0.088	0.083

Table 2: Comparison of PAC-Bayes bound with the true expected cost (estimated by sampling 10^5 obstacle environments). Using only 100 samples, with probability 0.99 over samples, the PAC-Bayes controller is guaranteed to have an expected success rate of 82.2%. The true expected success rate is approximately 91.3%.

using only 100 training environments, the PAC-Bayes controller is guaranteed (with probability $1-\delta = 0.99$) to have an expected success rate of 82.2% (i.e., an expected cost of 0.178). Exhaustive sampling indicates that the expected success rate for the PAC-Bayes controller is approximately 91.3% for this case. Videos of representative trials can be found online (see Appendix, Section 6).

Results (continuous policy space). Next, we consider a continuously parameterized policy space and apply the approach from Section 5.2. We parameterize our policies using the matrix $K \in \mathbb{R}^{20}$ in equation (10) while ensuring symmetry of the control law, i.e., $K[i] = -K[j]$ for $\theta[i] = -\theta[j]$ (note that K is no longer constrained to be of the form (11)). The dimensionality of the parameter space is thus $d = 10$. For the purpose of optimization, we employ a continuous surrogate cost function in place of the discontinuous 0-1 cost. We choose this to be the negative of the minimum distance to an obstacle along a trajectory (appropriately scaled to lie within $[0, 1]$). Note that we employ this surrogate cost only for optimization; all results are presented for the 0-1 cost. Numerically estimated gradients are used to perform gradient descent. We choose a prior $P_0 = \mathcal{N}_{\mu_0,s_0}$ with $s_0 = 0.01$; the mean μ_0 is given by a vector K of the form (11) with x-intercept 2.5 and y-intercept 10.0.

We use $N = 100$ training environments and choose confidence parameters $\delta = 0.009$, $\delta' = 0.001$, and $L = 30,000$ samples to evaluate the sample convergence bound in equation (8). The obtained PAC-Bayes bound $C^\star_{\text{bound}}$ is 0.224. Thus, with probability 0.99 over sampled training data, the optimized PAC-Bayes controller is guaranteed to have an expected success rate of 77.6%. Exhaustive sampling with 10^5 environments indicates that the expected success rate is approximately 92.5%. Videos of representative trials can be found online (see Appendix, Section 6).

7 Discussion and Conclusions

We have presented an approach for synthesizing controllers that provably generalize well to novel environments given a dataset of example environments. Our approach leverages PAC-Bayes theory to obtain upper bounds on the expected cost of (stochastic) controllers on novel environments and can be applied to robotic systems with continuous state and action spaces, nonlinear dynamics, and partially observable state. We synthesize controllers by explicitly minimizing this upper bound using convex optimization in the case of a finite policy space and using stochastic gradient descent in the case of continuously parameterized policies. We demonstrated our approach by synthesizing depth sensor-based obstacle avoidance controllers with guarantees on collision-free navigation in novel environments. Our simulation results compared the generalization guarantees provided by our technique with exhaustive numerical evaluations in order to demonstrate that our approach is able to provide strong bounds even with relatively few training environments.

Challenges and future work: On the practical front, our future work will focus on applying the presented approach to provide guarantees on neural-network based controllers for vision-based tasks such as navigation and grasping by leveraging existing datasets such as the Stanford 3D Indoor Spaces (S3DIS) dataset [42] and DexNet [22]. On the theoretical front, our approach inherits the challenges associated with generalization theory in supervised learning. For example, here we assumed that training and test environments are drawn independently from the same underlying distribution. There has been significant progress towards relaxing these assumptions in the supervised learning context (e.g., domain adaptation techniques [43] and PAC-Bayes bounds that do not assume i.i.d. data [44, 45]). An important feature of the reduction-based perspective we presented in Section 4 is the ability to immediately port over such improvements from the supervised learning setting to the control setting. Another exciting future direction is to combine our approach with *meta-learning* techniques in order to achieve provably data-efficient control on novel tasks. Specifically, we will investigate using a PAC-Bayes bound as part of the objective of a meta-learning algorithm such as MAML [46] to achieve improved generalization performance and few-shot learning.

We believe that the approach presented here along with the indicated future directions represent an important step towards synthesizing controllers with provable guarantees for challenging robotic platforms with rich sensory inputs operating in novel environments.

Acknowledgments

This work was partially supported by a Google Faculty Research Award.

References

[1] T. Schouwenaars, J. How, and E. Feron. Receding horizon path planning with implicit safety guarantees. In *Proceedings of the IEEE American Control Conference (ACC)*, volume 6, pages 5576–5581. IEEE, 2004.

[2] T. Fraichard. A short paper about motion safety. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1140–1145. IEEE, 2007.

[3] D. Althoff, M. Althoff, and S. Scherer. Online safety verification of trajectories for unmanned flight with offline computed robust invariant sets. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3470–3477. IEEE, 2015.

[4] A. Majumdar and R. Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research (IJRR)*, 36(8):947–982, July 2017.

[5] D. A. McAllester. Some PAC-Bayesian theorems. *Machine Learning*, 37(3):355–363, 1999.

[6] J. Langford and J. Shawe-Taylor. PAC-Bayes & margins. In *Advances in Neural Information Processing Systems*, pages 439–446, 2003.

[7] M. Seeger. PAC-Bayesian generalisation error bounds for gaussian process classification. *Journal of machine learning research*, 3(Oct):233–269, 2002.

[8] G. K. Dziugaite and D. M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

[9] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks. *preprint arXiv:1707.09564*, 2017.

[10] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5949–5958, 2017.

[11] B. A. Francis. *A course in H-infinity control theory*. Berlin; New York: Springer-Verlag, 1987.

[12] A. Charnes and W. W. Cooper. Chance-constrained programming. *Management science*, 6(1):73–79, 1959.

[13] L. Blackmore, H. Li, and B. Williams. A probabilistic approach to optimal robust path planning with obstacles. In *Proceedings of the IEEE American Control Conference (ACC)*. IEEE, 2006.

[14] M. P. Vitus and C. J. Tomlin. Closed-loop belief space planning for linear, Gaussian systems. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2152–2159. IEEE, 2011.

[15] M. Ono, M. Pavone, Y. Kuwata, and J. Balaram. Chance-constrained dynamic programming with application to risk-aware robotic space exploration. *Autonomous Robots*, 39(4):555–571, 2015.

[16] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[17] C. Richter, W. Vega-Brown, and N. Roy. Bayesian learning for safe high-speed navigation in unknown environments. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2015.

[18] C. Richter and N. Roy. Safe visual navigation via deep learning and novelty detection. In *Proceedings of Robotics: Science and Systems (RSS)*, 2017.

[19] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[20] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[21] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, pages 5074–5082, 2016.

[22] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

[23] J. Tobin, W. Zaremba, and P. Abbeel. Domain randomization and generative models for robotic grasping. *arXiv preprint arXiv:1710.06425*, 2017.

[24] M. M. Fard and J. Pineau. PAC-Bayesian model selection for reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1624–1632, 2010.

[25] M. M. Fard, J. Pineau, and C. Szepesvári. PAC-Bayesian policy evaluation for reinforcement learning. *arXiv preprint arXiv:1202.3717*, 2012.

[26] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49 (2-3):209–232, 2002.

[27] R. I. Brafman and M. Tennenholtz. R-max – A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

[28] J. Fu and U. Topcu. Probably approximately correct MDP learning and control with temporal logic constraints. *arXiv preprint arXiv:1404.7073*, 2014.

[29] V. Chandrasekaran and P. Shah. Relative entropy optimization and its applications. *Mathematical Programming*, 161(1-2):1–32, 2017.

[30] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[31] A. Maurer. A note on the PAC Bayesian theorem. *arXiv preprint cs/0411099*, 2004.

[32] A. Beygelzimer, V. Dani, T. Hayes, J. Langford, and B. Zadrozny. Error limiting reductions between classification tasks. In *Proceedings of the 22nd International Conference on Machine learning*, pages 49–56. ACM, 2005.

[33] Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM, 1994.

[34] J. Langford and R. Caruana. (not) bounding the true error. In *Advances in Neural Information Processing Systems*, pages 809–816, 2002.

[35] R. C. Arkin. *Behavior-based robotics*. MIT press, 1998.

[36] A. Beyeler, J.-C. Zufferey, and D. Floreano. Vision-based control of near-obstacle flight. *Autonomous robots*, 27(3):201, 2009.

[37] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert. Learning monocular reactive uav control in cluttered natural environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1765–1772. IEEE, 2013.

[38] J. Conroy, G. Gremillion, B. Ranganathan, and J. S. Humbert. Implementation of wide-field integration of optic flow for autonomous quadrotor navigation. *Autonomous robots*, 27(3):189, 2009.

[39] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning, 2018.

[40] S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[41] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. SCS: Splitting conic solver, version 2.0.2. `https://github.com/cvxgrp/scs`, Nov. 2017.

[42] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3D semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016.

[43] P. Germain, A. Habrard, F. Laviolette, and E. Morvant. PAC-Bayes and domain adaptation. *arXiv preprint arXiv:1707.05712*, 2017.

[44] L. Ralaivola, M. Szafranski, and G. Stempfel. Chromatic PAC-Bayes bounds for non-iid data: Applications to ranking and stationary β-mixing processes. *Journal of Machine Learning Research*, 11(Jul): 1927–1956, 2010.

[45] P. Alquier and B. Guedj. Simpler pac-bayesian bounds for hostile data. *Machine Learning*, 107(5): 887–902, 2018.

[46] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

Appendix

Anirudha Majumdar
Department of Mechanical and Aerospace Engineering
Princeton University
`ani.majumdar@princeton.edu`

Maxwell Goldstein
Department of Mathematics
Princeton University
`mag4@princeton.edu`

1 Computing KL inverse using Relative Entropy Programming

PAC-Bayes bounds are typically expressed as bounds on a quantity $q^\star \in [0,1]$ of the form $\mathrm{KL}(p\|q^\star) \leq c$ (for some $p \in [0,1]$ and $c \geq 0$). These bounds can then be used to upper bound $q^\star$ by the *KL inverse* as follows:

$$q^\star \leq \mathrm{KL}^{-1}(p\|c) := \sup\{q \in [0,1] \mid \mathrm{KL}(p\|q) \leq c\}. \tag{1}$$

In prior work on PAC-Bayes theory [1, 2], the KL inverse was numerically approximated using local root-finding techniques such as Newton's method, which do not have a priori guarantees on convergence to a global solution. Here we observe that the KL inverse is readily expressed as the optimal value to a simple Relative Entropy Program. In particular, the expression for the KL inverse in (1) corresponds to an optimization problem with a (scalar) decision variable q, a linear cost function (i.e., $-q$), linear inequality constraints (i.e., $0 \leq q \leq 1$), and a constraint on the KL divergence between the decision variable q and the constant p. We can thus compute the KL inverse exactly (up to numerical tolerances) using convex optimization (e.g., interior point methods [3]).

2 Derivation of Optimization Problem for Finite Policy Spaces

Minimizing the PAC-Bayes upper bound $C_{\text{PAC}}(P)$ directly corresponds to solving the following optimization problem:

$$\begin{aligned} \min_{p\in\mathbb{R}^L} \quad & \bar{C}p + \sqrt{\frac{\mathrm{KL}(p\|p_0) + \log(\frac{2\sqrt{N}}{\delta})}{2N}} \\ \text{s.t.} \quad & 0 \leq p \leq 1,\ \sum_j p[j] = 1. \end{aligned} \tag{2}$$

This optimization problem can be *equivalently* reformulated via an *epigraph constraint* [4] as:

$$\begin{aligned} \min_{p\in\mathbb{R}^L,\tau} \quad & \tau \\ \text{s.t.} \quad & \tau \geq \bar{C}p + \sqrt{\frac{\mathrm{KL}(p\|p_0) + \log(\frac{2\sqrt{N}}{\delta})}{2N}} \\ & 0 \leq p \leq 1,\ \sum_j p[j] = 1. \end{aligned}$$

We can then further equivalently rewrite this problem as:

$$\begin{aligned} \min_{p\in\mathbb{R}^L,\tau,\lambda} \quad & \tau \\ \text{s.t.} \quad & \lambda^2 \geq \frac{\mathrm{KL}(p\|p_0) + \log(\frac{2\sqrt{N}}{\delta})}{2N} \\ & \lambda = \tau - \bar{C}p,\ \lambda \geq 0 \\ & 0 \leq p \leq 1,\ \sum_j p[j] = 1. \end{aligned} \tag{3}$$

3 PAC-Bayes Control Synthesis via Gradient Descent

Algorithm 1 summarizes our approach for optimizing PAC-Bayes controllers in the setting where we have a continuously parameterized policy space. Note that in order to ensure positivity of $s \in \mathbb{R}^d_+$, we perform the optimization with respect to $\eta := \log(s)$.

Algorithm 1 PAC-Bayes Control Synthesis via Gradient Descent

```
 1: **Inputs:**
 2: $S = \{E_1, \ldots, E_N\}$: Training environments
 3: $\delta, \delta' \in (0,1)$: Probability thresholds
 4: $P_0$: Prior over controllers
 5: $\mu, s \in \mathbb{R}^d$: Initializations for $\mu$ and $s$
 6: $\gamma$: step size for gradient descent
 7: **Outputs:**
 8: $\mu^\star, s^\star$: Optimal $\mu, s$
 9: $C^\star_{\text{bound}} := \text{KL}^{-1}\Big(\bar{C}_S(\mathcal{N}_{\mu^\star,s^\star}; L, \delta') \| \frac{\text{KL}(\mathcal{N}_{\mu^\star,s^\star}\|P_0)+\log(\frac{2\sqrt{N}}{\delta})}{N}\Big)$
10: **Procedure:**
11: $B(\mu, s, w) := \frac{1}{N}\sum_{E \in S} C(r_w; E) + \sqrt{\frac{\text{KL}(\mathcal{N}_{\mu^\star,s^\star}\|P_0)+\log(\frac{2\sqrt{N}}{\delta})}{2N}}$
12: **while** $\neg$converged **do**
13:     Sample $\xi \sim \mathcal{N}_{0,I_d}$ and set $w \leftarrow \mu + \sqrt{s} \odot \xi$
14:     $\mu \leftarrow \mu - \gamma \nabla_\mu B(\mu, \exp(\eta), w)$
15:     $\eta \leftarrow \eta - \gamma \nabla_\eta B(\mu, \exp(\eta), w)$
16:     $s \leftarrow \exp(\eta)$
17: **end while**
```

4 Dynamics of Differential Drive Robot

We model the robot in our example as a differential drive vehicle. The state of the system is given by $[x, y, \psi]$, where x and y are the x and y positions of the vehicle respectively, and ψ is the yaw angle. The nonlinear dynamics are given by:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -\frac{r}{2}(u_l + u_r)\sin(\psi) \\ \frac{r}{2}(u_l + u_r)\cos(\psi) \\ \frac{r}{L}(u_r - u_l) \end{bmatrix}, \tag{4}$$

where u_l and u_r are the control inputs (corresponding to the left and right wheel speeds respectively), $r = 0.1$m corresponds to the radius of the wheels, and $L = 0.5$m corresponds to the width of the base of the vehicle. We set:

$$u_l = u_0 - u_{\text{diff}}, \quad u_r = u_0 + u_{\text{diff}}, \tag{5}$$

where $u_0 = v_0/r$ with $v_0 = 2.5$m/s. This ensures that the robot has a fixed speed v_0. We limit the turning rate by constraining $u_{\text{diff}} \in [-u_0/2, u_0/2]$. The system is simulated as a discrete-time system with time-step $\Delta t = 0.05$s.

5 Example of a Simple Reactive Controller

An example of $K[i]$ as a function of $\theta[i]$ is shown in Figure 1.

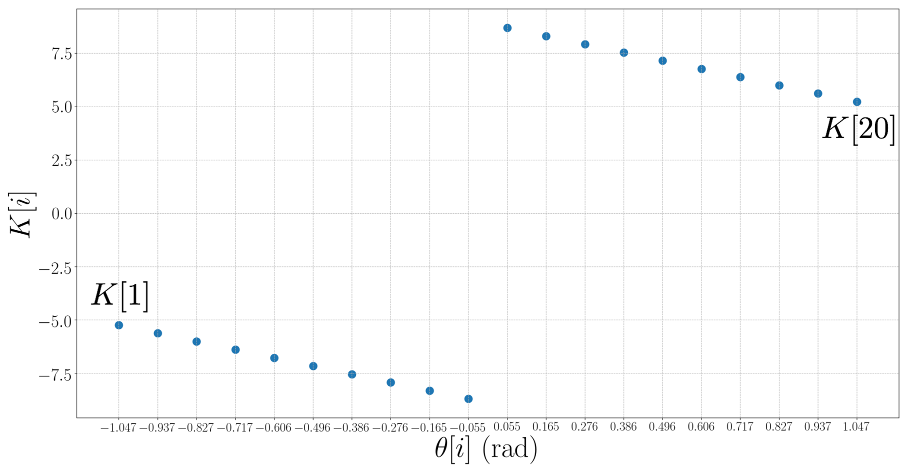

Figure 1: Example of $K[i]$ as a function of $\theta[i]$.

6 Code and Video

A complete implementation of our REP-based approach is available online at:
https://github.com/irom-lab/PAC-Bayes-Control

Videos of results are available online at:
https://youtu.be/zu_O-lW5X_8

References

[1] G. K. Dziugaite and D. M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

[2] G. K. Dziugaite and D. M. Roy. Entropy-sgd optimizes the prior of a pac-bayes bound: Data-dependent pac-bayes priors via differential privacy. *arXiv preprint arXiv:1712.09376*, 2017.

[3] V. Chandrasekaran and P. Shah. Relative entropy optimization and its applications. *Mathematical Programming*, 161(1-2):1–32, 2017.

[4] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.