2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Robust Model-free Reinforcement Learning with Multi-objective Bayesian Optimization

Matteo Turchetta[1] Andreas Krause[1] Sebastian Trimpe[2]

Abstract— **In reinforcement learning (RL), an autonomous agent learns to perform complex tasks by maximizing an exogenous reward signal while interacting with its environment. In real world applications, test conditions may differ substantially from the training scenario and, therefore, focusing on pure reward maximization during training may lead to poor results at test time. In these cases, it is important to trade-off between performance and robustness while learning a policy. While several results exist for robust, model-based RL, the model-free case has not been widely investigated. In this paper, we cast the robust, model-free RL problem as a multi-objective optimization problem. To quantify the robustness of a policy, we use delay margin and gain margin, two robustness indicators that are common in control theory. We show how these metrics can be estimated from data in the model-free setting. We use multi-objective Bayesian optimization (MOBO) to solve efficiently this expensive-to-evaluate, multi-objective optimization problem. We show the benefits of our robust formulation both in sim-to-real and pure hardware experiments to balance a Furuta pendulum.**

I. Introduction

In reinforcement learning (RL) [1], the goal is to learn a controller to perform a desired task from the data produced by the interaction between the learning agent and its environment. In this framework, autonomous agents are trained to maximize their return. It is common to assume that such agents will be deployed in conditions that are similar, if not equal, to those they were trained in. In this case, a return-maximizing agent performs well at test time. However, in real world applications, this assumption may be violated. For example, in robotics, we can use RL to learn to fly a drone indoor. However, later on we may use the same drone to carry a payload in a windy environment. The new environmental conditions and the possible deterioration of the drone components due to their usage may result in a poor, if not catastrophic, performance of the learned controller. Another scenario where training and testing conditions differ substantially is the sim-to-real setting, i.e., when we deploy a controller trained in simulation on a real-world agent.

Considering robustness alongside performance when learning a controller can limit performance degradation due to different training and testing environments. In special cases, these goals may be aligned, and a high-performing controller can also be robust. This is the case for the Linear Quadratic Regulator (LQR), a linear state-feedback controller that is optimal for the case of linear dynamics, quadratic cost, and perfect state measurements. It is well-known that the LQR exhibits strong robustness indicators, such as gain and phase margins [2]. While performance and robustness go hand in hand for the LQR, they are often conflicting in other cases. For example, a celebrated result in control theory shows that the Linear Quadratic Gaussian (LQG) regulator - the noisy counterpart of the LQR - can be arbitrarily close to instability, despite being optimal [3]. Thus, in general, we need to trade-off between performance and robustness [4].

Contributions. While many works investigating the performance/robustness trade-off exist in both the RL and control theory literature for the model-based setting, few results are known for the model-free scenario. However, there are several real-world scenarios where models are not available, inaccurate, or too expensive to use, but robustness is fundamental. Thus, in this paper, we introduce the first data-efficient, robust, model-free RL method based on policy optimization with multi-objective Bayesian optimization (MOBO). In particular, these are our contributions:

- We formulate the robust, model-free RL as a multi-objective optimization problem.
- We propose a model-free, data-driven evaluation of delay and gain margins, two common robustness indicators from the model-based setting (where they are computed analytically).
- We solve this problem efficiently with expected hypervolume improvement (EHI).
- We introduce the first method that can learn robust controllers directly on hardware in a model-free fashion.
- We show how our approach outperforms non-robust policy optimization in evaluations on a Furuta pendulum for both a sim-to-real and a pure hardware setting.

Related work. Robustness has been widely investigated in control theory [5], and standard robust control techniques for linear systems include loop transfer recovery [6], H_∞ control, and μ synthesis [5], [7]. However, these methods typically assume the availability of a model, and none of these includes a learning component. Recently, robustness has drawn attention in data-driven settings, giving rise to the field of robust, model-based RL. Robust Markov decision processes study the RL problem when the transition model is subject to known and bounded uncertainties. For example, [8] studies the dynamic programming recursion in this setting. Other methods that consider parametric uncertainties include [9], [10]. All the previous methods are model-based.

Robustness and performance are typical objectives in

[1] Learning and Adaptive Systems Group, ETH Zurich, Zürich, Switzerland matteotu@inf.ethz.ch krausea@ethz.ch

[2] Intelligent Control Systems Group, Max Planck Institute for Intelligent Systems, Stuttgart, Germany trimpe@is.mpg.de

This research was supported in part by the Max Planck ETH Center for Learning Systems, the Max Planck Society and the Cyber Valley Initiative.

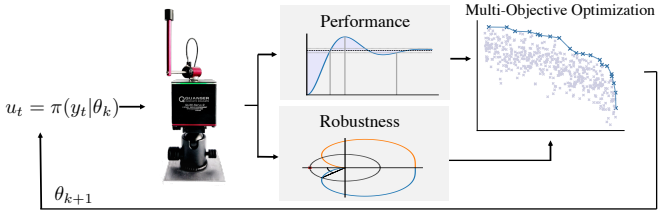

Fig. 1. The multi-objective optimization model for the robust, model-free RL problem. We choose a controller corresponding to parameters θ_k. Subsequently, we evaluate its performance and robustness by deploying it on the system. Finally, we use these observations to choose a new controller.

control design, which often conflict each other, thus requiring design trade-offs [4], [11]. In the model free literature, this trade-off is often fixed a priori and the resulting problem is solved with standard optimization methods. In [12] a weighted cost that balances performance and robustness is optimized. In [13] robust controllers are learned via gradient ascent with random multiplicative noise on the control action. In [14], [15] external, adversarial disturbances are used instead. In these works, the upper bound on the magnitude of the disturbance implicitly balances robustness and performance. However, setting this trade-off is often not intuitive and, in case the requirements are misspecified or updated, a new controller must be learned. Alternatively, robust control design methods based on multi-objective optimization explore the spectrum of such trade-offs. The work in [16] gives a review of such methods, with a focus on genetic algorithms, which, due to their low data efficiency require the model to compute the robustness indices.

Model-free RL methods are typically validated in simulations due to their high sample complexity. However, in robotics, it is crucial to test these methods on hardware. Bayesian optimization (BO) [17], [18] has been successfully applied to learn low-dimensional controllers for episodic tasks through policy search on robotic systems. For example, [19] learns a linear controller for a quadrotor hovering task, [20] learns a linear state feedback controller for a cart-pole system in a sim-to-real setting and [21], [22] tune controllers for locomotion tasks. However, none of these methods considers robustness, making ours the first one to learn robust controllers from data directly on hardware.

MOBO is the branch of BO that solves multi-objective problems. MOBO algorithms include EHI [23], ϵ-PAL [24], and PESMO [25], while [26] combines them with techniques from robust optimization. They have been applied to several tasks including trading off prediction speed and accuracy in machine learning models. However, they have rarely been applied to RL. For example, this has been done in [27], [28], where a trade-off between frontal camera movement and forward speed is found for a snake-like robot and in [29], where energy efficiency and walking speed are optimized for a micro hexapod. Robustness is not treated in these works.

II. Problem Statement

In this section, we introduce our formulation of robust model-free RL as a multi-objective optimization problem. For ease of exposition, we limit ourselves to two objectives. However, this approach naturally extends to the any number of objectives, for example, multiple robustness indicators.

We assume we have a system with *unknown* dynamics, h, and *unknown* observation model, g,

$$\mathbf{x}_{t+1} = h(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t), \qquad \mathbf{o}_t = g(\mathbf{x}_t, \mathbf{v}_t) \tag{1}$$

where $\mathbf{x}$ is the state, $\mathbf{u}$ is the control input, $\mathbf{o}$ is the observation and $\mathbf{w}$ and $\mathbf{v}$ are the process and sensor noise. An RL agent aims at learning a controller $\mathbf{u}_t = \pi(\mathbf{o}_t|\theta)$, i.e., a mapping parametrized by θ from an observation $\mathbf{o}_t$ to an action $\mathbf{u}_t$ that allows it to complete its task. Policy optimization algorithms are a class of model-free RL methods that solve this problem by optimizing the performance of a given controller for the task at hand as a function of the parameters θ. Concretely, given a performance metric $f_1 : \Theta \to \mathbb{R}$, standard, non-robust policy optimization algorithms aim to find $\theta^* \in \operatorname{argmax}\ f_1(\theta)$. In this work, we consider regulation tasks, i.e., bringing and keeping the system in a desired goal state $\overline{\mathbf{x}}$. This includes common problems like stabilization, set-point tracking, or disturbance rejection. The performance indicator f_1 encodes these objectives.

To extend this framework to the robustness-aware case, we use a second function $f_2 : \Theta \to \mathbb{R}$ that measures the robustness of a controller. Since both the dynamics h and the observation model g are unknown, we must evaluate or approximate the value of f_2 from data. In Sec. III-B, we introduce the gain and the delay margin, two alternatives for f_2 that are commonly used in model-based control and we discuss how to evaluate them in the model-free setting.

We aim at finding the best controller in terms of performance and robustness, as measured by f_1 and f_2. However, since we compare controllers based on multiple, and possibly conflicting, criteria, we cannot define a single best controller. Given a controller θ, we denote with $\mathbf{f}_\theta = [f_1(\theta), f_2(\theta)]$ the array containing its performance and robustness values. To compare two controllers θ_1 and θ_2, we use the canonical partial order over $\mathbb{R}^2$: $\mathbf{f}_{\theta_1} \succeq \mathbf{f}_{\theta_2}$ iff $f_i(\theta_1) \geq f_i(\theta_2)$ for $i = 1, 2$. This induces a relation in the controller space Θ: $\theta_1 \succeq \theta_2$ iff $\mathbf{f}_{\theta_1} \succeq \mathbf{f}_{\theta_2}$. If $\theta_1 \succeq \theta_2$, we say that θ_1 dominates θ_2. The Pareto set $\Theta^* \subseteq \Theta$ is the set of non-dominated points in the domain, i.e., $\theta^* \in \Theta^*$ iff $\exists i = 1, 2$ such that $f_i(\theta^*) > f_i(\theta)$ for all $\theta \in \Theta$. The Pareto front is the set of function values corresponding to the Pareto set. The Pareto set is optimal in the sense that, for each point θ^* in it, it is not possible to find another point in the domain that improves the value of one objective without degrading another [30]. The goal of this paper is to approximate Θ^* from data.

Fig. 1 represent our problem graphically: we suggest a controller, we evaluate its performance and robustness on the system and we select a new controller based on these observations to find an approximation of the Pareto front.

III. Learning the Performance-Robustness Trade-off

For the robust, model-free RL setting we consider, we propose to learn the Pareto front characterizing the performance-

robustness trade-off of a given system with MOBO. Here, we describe the necessary components to solve our problem in a data efficient way: MOBO and the robustness and performance indicators used in our experiments. Moreover, we discuss how to evaluate such indicators from data in a model-free fashion.

A. Multi-objective Bayesian optimization

MOBO algorithms solve multi-objective optimization problems by sequentially querying the objective at different inputs and obtaining noisy evaluations of the corresponding values. They build a statistical model of the objectives to capture the belief over them given the data available. They measure how informative a point in the domain is about the problem solution with an acquisition function. At every iteration, they evaluate the objective at the most informative point, as measured by the acquisition function. Thus, the complex multi-objective optimization problem is decomposed into a sequence of simpler scalar-valued optimization problems. In the following, we describe the surrogate model and the acquisition function used in this work.

Intrinsic Model of Coregionalization A single-output Gaussian process (GP) [31] is a probability distribution over the space of functions of the form $f : \Theta \to \mathbb{R}$, such that the joint distribution of the function values computed over any finite subset of the domain follows a multi-variate Gaussian distribution. A GP is fully specified by a mean function $\mu : \Theta \to \mathbb{R}$, which, w.l.o.g., is usually assumed to be zero, $\mu(\theta) = 0$ for all $\theta \in \Theta$, and a covariance function, or kernel, $k : \Theta \times \Theta \to \mathbb{R}$. The kernel encodes the strength of statistical correlation between two latent function values and, therefore, it expresses our prior belief about the function behavior.

Similarly, a D-output GP is a probability distribution over the space of functions of the form $\mathbf{f} : \Theta \to \mathbb{R}^D$. The difference with respect to single-output GPs is that, in this case, the kernel must capture the correlation across different output dimensions in addition to the correlation of function values at different inputs. The simplest way of doing this is by assuming that each output is independent. However, this model disregards the fundamental trade-off between robustness and performance that we are considering. For a review on kernels for multi-output GPs, see [32]. In this work, we use the intrinsic model of coregionalization (ICM), which defines the covariance between the i^{th} value of $\mathbf{f}(\theta)$ and the j^{th} value of $\mathbf{f}(\theta')$ by separating the input and the output contribution as follows, $b_{ij}k(\theta, \theta')$. In this case, we say $\mathbf{f} \sim \mathcal{GP}(\boldsymbol{\mu}(\cdot), \mathbf{K}(\cdot,\cdot) = \mathbf{B}k(\cdot,\cdot))$, where $\boldsymbol{\mu} : \Theta \to \mathbb{R}^D$ is a D-dimensional mean function, $k : \Theta \times \Theta \to \mathbb{R}$ is a scalar-valued kernel and $\mathbf{B} \in \mathbb{R}^{D\times D}$ is a matrix describing the correlation in the output space (more details on $\mathbf{B}$ in Sec. IV). Given N noisy observations of $\mathbf{f}$, $\mathcal{D} = \{(\theta_1, \mathbf{y}_1), \cdots (\theta_N, \mathbf{y}_N)\}$, with $\mathbf{y}_i = \mathbf{f}(\theta_i) + \omega_i$, where $\omega_i \sim \mathcal{N}(\mathbf{0}, \Sigma)$ is i.i.d. Gaussian noise, we can compute the posterior distribution of the function values conditioned on $\mathcal{D}$ at a target input $\theta^\star$ in closed form as $p(\mathbf{f}(\theta^\star)|\mathcal{D}, \theta^\star) \sim \mathcal{N}(\mathbf{f}^\star(\theta^\star), \mathbf{K}^\star(\theta^\star, \theta^\star)))$. We denote with $\boldsymbol{\theta}$ the inputs contained in $\mathcal{D}$ and with $\mathbf{K}(\boldsymbol{\theta}, \boldsymbol{\theta})$ the $ND \times ND$ matrix with entries $(\mathbf{K}(\theta_u, \theta_v))_{i,j}$ for $u, v = 1, \ldots, N$ and $i, j = 1, \ldots, D$, then

$$\mathbf{f}^\star(\theta^\star) = \mathbf{K}_{\theta^\star}^\top (\mathbf{K}(\boldsymbol{\theta}, \boldsymbol{\theta}) + \boldsymbol{\Sigma})^{-1} \overline{\mathbf{y}}, \tag{2}$$

$$\mathbf{K}^\star(\theta^\star, \theta^\star) = \mathbf{K}(\theta^\star, \theta^\star) - \mathbf{K}_{\theta^\star}(\mathbf{K}(\boldsymbol{\theta}, \boldsymbol{\theta}) + \boldsymbol{\Sigma})^{-1} \mathbf{K}_{\theta^\star}^\top, \tag{3}$$

where $\boldsymbol{\Sigma} = \Sigma \otimes I_N$, with $\otimes$ denoting the Kronecker product, $\mathbf{K}_{\theta^\star} \in \mathbb{R}^{D \times ND}$ has entries $(\mathbf{K}(\theta^\star, \theta_v))_{i,j}$ for $v = 1, \ldots, N$ and $i, j = 1, \ldots, D$ and $\overline{\mathbf{y}}$ is the ND-dimensional vector containing the concatenation of the observations in $\mathcal{D}$.

Expected Hypervolume Improvement EHI is an acquisition function introduced in [23], which selects inputs to evaluate based on a notion of improvement with respect to the incumbent solution. In multi-objective optimization, incumbent solutions take the form of approximations of the Pareto set, $\mathcal{X}^*$, whose quality is measured by the hypervolume indicator induced by the corresponding front, $\mathcal{Y}^*$ with respect to a reference r. Formally, the hypervolume indicator of a set of points A with respect to a reference r, $\text{HV}(A; r)$, is the Lebesgue measure of the hypervolume covered by the boxes that have an element in A as upper corner and the reference as lower corner. It quantifies the size of the portion of the output space that is Pareto-dominated by the points in A. Given an estimate of the Pareto front, $\mathcal{Y}^*$, the hypervolume improvement of $\theta \in \Theta$ is defined as the relative improvement in hypervolume obtained by adding the function value at θ, $f(\theta)$, to $\mathcal{Y}^*$, $\text{HI}(f(\theta); \mathcal{Y}^*, r) = \text{HV}(\mathcal{Y}^* \cup \mathbf{f}(\theta); r) - \text{HV}(\mathcal{Y}^*; r)$. However, we do not know $\mathbf{f}(\theta)$. Instead, we have a belief over its value expressed by the posterior distribution of the GP, which, in turn, induces a distribution over the hypervolume improvement corresponding to an input θ. The EHI acquisition function quantifies the informativeness of an input θ toward the solution of the multi-objective optimization problem through the expectation of this distribution,

$$\alpha(\theta|\mathcal{D}, \mathcal{Y}^*, r) = \int_{f(\theta)\in\mathbb{R}^n} \text{HI}(\mathbf{f}(\theta); \mathcal{Y}^*), r) p(\mathbf{f}(\theta)|\mathcal{D}) d\mathbf{f}(\theta). \tag{4}$$

[23] shows how to compute the integral in (4) in closed form.

B. Robustness

In general, robustness can have very different meanings. One may desire to ensure robustness to a certain class of disturbances, imperfections in the control system, or uncertainty in the process, for example. In control theory, the latter is often understood as robustness in the stricter sense. Specifically, *robust stability* assures that a controller stabilizes every member from a set of uncertain processes [5]. Such processes can, for example, be defined through a nominal process and variations thereof. Different variations lead to different robustness characterizations. Likewise, there are different notions of stability that are meaningful depending on the context. For example, for a deterministic system, asymptotic stability, i.e., $\mathbf{x}_t \to \overline{\mathbf{x}}$ as $t \to \infty$, where $\overline{\mathbf{x}}$ is an equilibrium of the system, is often used; for systems that are continuously excited, e.g., through noise, and thus cannot approach $\overline{\mathbf{x}}$, one may seek the above limit to hold in expectation or practical stability in the sense of a bounded

state, i.e., $\|\mathbf{x}_t - \overline{\mathbf{x}}\| \leq \mathbf{x}_{\max}$ for all $t \geq \bar{t}$. A controller is unstable when the respective condition does not hold (e.g., no asymptotic convergence, or $\mathbf{x}_t$ grows beyond any bounds).

While many sophisticated robustness metrics have been developed, stability margins such as gain and delay margins are some of the most common and intuitive ones [11, Sec. 9.3]. We consider these in this work and comment on alternatives in Sec. V. Below, we formally introduce them and we explain how to evaluate them in a model-free setting. Notice that, our data-driven definitions can be extended to any setting where a success/failure outcome can be defined and, therefore, are not limited to stability considerations.

Gain margin. In classical control, the *upper* (*lower*) *gain margin* is defined for single-input-single-output (SISO) linear systems as the largest factor $\kappa_{\text{max}} \in (1, \infty)$ (the smallest factor $\kappa_{\text{min}} \in (0, 1)$) that can multiply the open-loop transfer function so that the closed-loop system is stable [33, Sec. 9.5]. As the open-loop transfer function encodes both the process and the controller dynamics, the factor may represent uncertainty in the process gain or the actuator efficiency, for example. In this work, we consider a factor κ to be multiplied by the control action (i.e., $\mathbf{u}_t = \kappa \times \pi(\mathbf{o}_t|\theta)$), which is equivalent to the definition for linear SISO systems, but can also be used for nonlinear ones. It quantifies how much we can lower/amplify the control action before making the system unstable. In a way, it quantifies how "far" we are from instability and, thus, how much we can tolerate differences between training and testing.

Delay margin. Similarly, we define the *delay margin* as the largest time delay on the measurement $\mathbf{o}_t$ such that the controlled system is still stable. Formally, it is the largest value of $d \in (0, \infty)$ such that the closed-loop system with the delayed control action $\mathbf{u}_t = \pi(\mathbf{o}_{t-d}|\theta)$ is stable. As delay in data transmission between sensor, controller, and actuator, and in the control computation are present in most control systems, the delay margin is a very relevant measure.

Estimate from data. While the indicators above can be readily computed for linear systems, they are difficult to compute analytically if the model is nonlinear, or impossible if no model is available, as considered herein. We describe an experiment to estimate the delay margin from data in a model-free setting (those for the gain margins are analogous). For general non-linear systems, stability with respect to an equilibrium is a local property. Thus, we assume we can reset the system to a state in the neighborhood of the equilibrium of interest, i.e., we have $\mathbf{x}_0 \in \mathcal{B}(\overline{\mathbf{x}}, \rho)$, where $\mathcal{B}(\overline{\mathbf{x}}, \rho)$ is a ball centered at $\overline{\mathbf{x}}$ of radius ρ. We can establish whether the delay margin, denoted with d^*, is larger or smaller than a delay d by resetting the system near $\overline{\mathbf{x}}$, deploying the delayed controller $\pi(\mathbf{o}_{t-d}|\theta)$ and evaluating the stability of the resulting trajectory.

In practice, two problems arise with this approach: *(i)* we can evaluate a finite number of delays; and *(ii)* while stability is an asymptotic condition on the state, we do not know the state and we run finite experiments. The first problem requires us to select carefully the delays we evaluate. We know that increasing values of delay take a stable system closer to instability. Thus, given m delays $[d_1, \cdots, d_m]$, we do a binary search to find the largest one for which the closed-loop system is stable. This allows us to approximate the delay margin with $\log m$ experiments. Not knowing the state $\mathbf{x}$ can be solved by estimating it from the noisy sensor measurements $\mathbf{o}_t$ or by introducing a new definition of stability based on $\mathbf{o}_t$ rather than $\mathbf{x}_t$. Concerning the finite trajectories we note that, in practice, it is rare to have small compounding deviations from $\overline{\mathbf{x}}$ resulting in a divergent behavior emerging only in the long run. Often, a controller makes the system converge to or diverge from $\overline{\mathbf{x}}$ within a short amount of time. In our experiments, we say that a controller stabilizes the system if, after a burn-in time that accounts for the transient behavior, it keeps the state within a box around $\overline{\mathbf{x}}$. To avoid overestimating the Pareto front while retaining efficiency, only controllers with good margins are investigated further with longer experiments to eliminate potential outliers due to the finite trajectory issue.

Reliably estimating robustness indicators and stability of a system without a model is challenging. The estimation technique we presented is intuitive and easy to implement. While it does not provide formal guarantees on the estimation error, we show in Sec. IV that it is accurate enough to greatly improve the robustness of our algorithm with respect to the non-robust policy optimization baseline.

Algorithm 1 Robust policy optimization

Inputs: Reference r
$\mathcal{D}_0 \leftarrow \emptyset,\ \mathcal{Y}_0^* \leftarrow \{r\}$
for $k = 0, 1, \ldots$ **do**
  $\theta_{k+1} \leftarrow \text{argmax}_{\theta \in \Theta}\ \text{EHI}(\theta | \mathcal{D}_k, \mathcal{Y}_k^*, r)$
  $y_{k+1}^1 \leftarrow \texttt{performance\_experiment}(\theta_{k+1})$
  $y_{k+1}^2 \leftarrow \texttt{robustness\_experiment}(\theta_{k+1})$
  $\mathcal{D}_{k+1} \leftarrow \mathcal{D}_k \cup \{(\theta_{k+1}, [y_{k+1}^1, y_{k+1}^2])\}$
  $\Theta_{k+1}^*, \mathcal{Y}_{k+1}^* \leftarrow \texttt{Pareto\_set\_and\_front}(\mathcal{D}_{k+1})$
Outputs: Pareto set Θ^*, Pareto front $\mathcal{Y}^*$

C. Algorithm

In this section, we describe our robust policy optimization algorithm, for the pseudocode see Algorithm 1. At iteration k, we select the controller that maximizes the EHI criterion. Then, we run two experiments to estimate its performance and robustness. For the performance, we introduce a state and action dependent reward and we define the return as the average reward obtained over an episode. The performance index is defined as the expectation of the return, which we approximate with a Monte Carlo estimate over multiple episodes. To estimate the robustness, we use the experiments from Sec. III-B. We update the data set with the experiments results. Finally, we update the estimate of the Pareto front that is used to compute the EHI as the set of dominating points of the data set. Other options to compute such estimate from the posterior of the GP exist. However, they are computationally more expensive and they resulted in a similar performance in our experiments. In the end, the algorithm returns an estimate of the Pareto set and front. The choice of

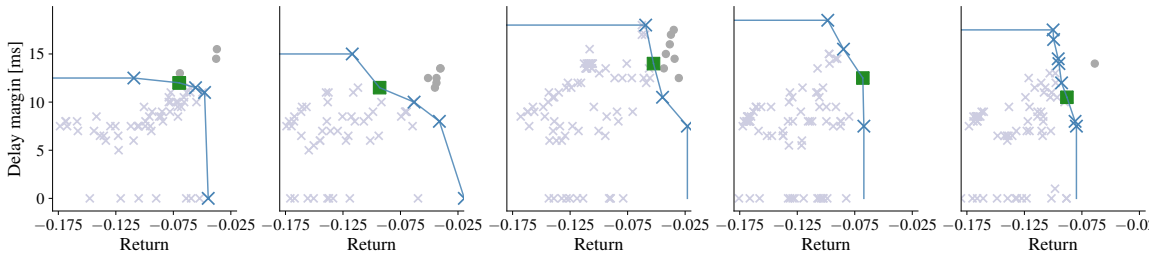

Fig. 2. Pareto fronts identified in simulation for the delay margin experiment (MOBO-DM). The green squares indicate the controller tested on the hardware, see Table I. They were chosen to be approximately on the elbow of the front without further tuning. The gray circles indicate the controllers that were discarded running longer simulations (see Sec. III-B for details).

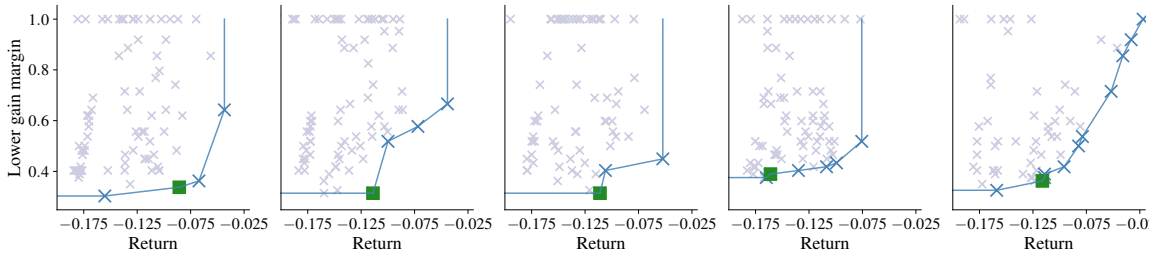

Fig. 3. Pareto fronts identified in simulation for the gain margin experiment (MOBO-GM). The green squares indicate the controller that is tested on the hardware, see Table I. They were chosen to be approximately on the elbow of the front without further tuning.

a controller from the Pareto set depends on the performance-robustness trade-off required by the test applications and, therefore, the choice is left to the practitioner.

IV. Experimental Results

We compare the robust policy optimization algorithm in Algorithm 1 to its non-robust counterpart based on scalar BO as, e.g., in [19], [34]. We use the scalar equivalent of EHI for the non-robust case, i.e., the expected improvement (EI) algorithm [17]. We present two set of experiments: training controllers in simulation and directly on hardware, respectively. In both cases, the learned controllers are tested on the hardware in a set of different conditions.

System. We learn a controller for a Furuta pendulum [35] (Figure 1), a system that is closely related to the cart pole. It replaces the cart with an arm that rotates on the horizontal plane. In our experiments, we use the Qube Servo 2 by Quanser [36], a small scale Furuta pendulum. It uses a brushless DC motor to exert a torque on the rotary arm, and it uses two incremental optical encoders with 2048 counts per revolution to measure the angle of the rotary arm and the pendulum. For sim-to-real, we use the non-linear rigid body dynamics model provided in the Qube Servo 2 manual [36]. A more detailed model is presented in [35].

Controller. We consider a state feedback controller to stabilize the pendulum about the vertical equilibrium. The system has four states, $\mathbf{x} = [\alpha, \beta, \omega, \phi]$: the angular position of the rotary arm and the pendulum, α, β, with $\beta = 0$ being the vertical position, and the corresponding angular velocities, ω, ϕ. We control the voltage applied to the motor, V_m. We use the encoder readings as estimates of the angular positions, $\hat{\alpha}, \hat{\beta}$. We apply a low-pass filter to the difference of consecutive angular positions to estimate the angular velocities, $\hat{\omega}, \hat{\phi}$. We aim to find a controller of the form $u_t = V_\mathrm{m} = [\theta_1, \theta_2, \theta_3, \theta_4][\hat{\alpha}, \hat{\beta}, \hat{\omega}, \hat{\phi}]^\top$.

Scaling and reward. We define a state and action dependent reward as the negative of a quadratic cost, $r(\mathbf{x}, \mathbf{u}) = -(\mathbf{x}^\top Q \mathbf{x} + \mathbf{u}^\top R \mathbf{u})$ with $Q = \mathrm{diag}(1, 10, 0, 0)$ and $R = 8$. The performance associated to a controller is the expected average reward it induces on the system, that is, for a trajectory of duration T, $\mathbb{E}[1/T \int_0^T r(\mathbf{x}_t, \pi(\mathbf{x}_t|\theta))dt]$. To prevent one of the objectives from dominating the contribution to the hypervolume improvement in the EHI algorithm, we must normalize them. We control the range of the robustness indicators, see Sec. III-B, and, therefore it is easy to rescale them to the $[0, 1]$ range. We observe empirically that the unnormalized return ranges in $[-500, 0]$. Thus, we clip every return value to this range and we rescale it to the $[0, 1]$ interval. Since the pendulum incurs substantially different returns when a stabilizing or destabilizing controller is used, we cannot rescale the range linearly. Instead, we use a piece-wise linear function. In particular, since we observe empirically that stabilizing controllers have a performance between -20 and 0, we rescale linearly the range $[-500, -20]$ to $[0, 0.5]$ and the range $[-20, 0]$ to $[0.5, 1]$. This differentiates coarsely the quality of unstable controllers, and it gives a more refined scale over stable ones.

Surrogate models. For the non-robust algorithm, we use a standard GP model with a zero mean prior and a Matérn kernel with $\nu = 5/2$ with automatic relevance determination (ARD). We set the hyperprior over the length-scales to $\mathrm{Lognormal}(1, 3)$ and over the standard deviation to $\mathrm{Lognormal}(0.35, 1)$. We use a Gaussian likelihood with no hyperprior. Similarly, for the robust algorithm, we use a zero prior mean. The correlation in the input space in the ICM model is captured by an ARD Matérn kernel with $\nu = 5/2$,

TABLE I

SIM-TO-REAL EXPERIMENT: WE TRAIN IN SIMULATION 5 DIFFERENT CONTROLLERS WITH SCALAR BO, MOBO WITH DELAY MARGIN OR LOWER GAIN MARGIN. WE TEST EACH ONE 5 TIMES ON THE HARDWARE IN 4 SCENARIOS AND WE COMPARE THEM ACCORDING TO 3 METRICS (SEE THE MAIN TEXT FOR AN IN DEPTH DESCRIPTION). THE ROBUST CONTROLLERS CONSISTENTLY OUTPERFORM THE NON-ROBUST ONES ACROSS ALL SCENARIOS.

	Standard			Motor noise			Sensor noise			Add 2 g		
	$\mathbb{E}[R]$	Fail	Fail time (s)	$\mathbb{E}[R]$	Fail	Fail time (s)	$\mathbb{E}[R]$	Fail	Fail time (s)	$\mathbb{E}[R]$	Fail	Fail time (s)
Scalar BO	-0.150	80%	0.92	-0.151	80%	0.97	-0.151	80%	1.05	-0.185	100%	1.03
MOBO-DM	-0.044	32%	4.61	-0.038	20%	4.07	-0.063	20%	4.32	-0.126	84%	3.21
MOBO-GM	**-0.003**	**0%**	∞	**-0.013**	**0%**	∞	**-0.057**	**0%**	∞	**-0.004**	**0%**	∞

TABLE II

HARDWARE EXPERIMENT: WE TRAIN ON HARDWARE ONE CONTROLLER WITH SCALAR BO AND ONE WITH MOBO WITH LOWER GAIN MARGIN. WE TEST EACH ONE 5 TIMES ON THE HARDWARE IN 4 SCENARIOS AND WE COMPARE THEM ACCORDING TO 3 METRICS (SEE THE MAIN TEXT FOR AN IN DEPTH DESCRIPTION). THE ROBUST CONTROLLER CONSISTENTLY OUTPERFORM THE NON-ROBUST ONES ACROSS ALL SCENARIOS.

	Add 5 g			Add 9 g			Add 10 g and 8 cm			Add 5 g, motor + sensor noise		
	$\mathbb{E}[R]$	Fail	Fail time (s)	$\mathbb{E}[R]$	Fail	Fail time (s)	$\mathbb{E}[R]$	Fail	Fail time (s)	$\mathbb{E}[R]$	Fail	Fail time (s)
Scalar BO	-0.101	**0%**	∞	-0.669	100%	4.07	-0.711	100%	3.59	**-0.259**	**0%**	∞
MOBO-GM	**-0.031**	**0%**	∞	**-0.026**	**0%**	∞	**-0.0259**	**0%**	∞	-0.366	**0%**	∞

with the same hyperprior as the non-robust case. For the correlation in the output space, we set a Gaussian hyperprior over each entry of the matrix $\mathbf{B}$, $\mathcal{N}(0, 1)$. We use a Gaussian likelihood with a diagonal covariance matrix. In both cases, we udpate the hyperparameters using a maximum a posteriori estimate after every new data point is acquired.

Training. In the sim-to-real setting, we train one controller for each of the following methods in 5 independent experiments: scalar BO (non-robust), MOBO with performance and delay margin (DM), and MOBO with performance and lower gain margin (GM). The training consists of 200 BO iterations evaluated in simulation. In the hardware training setting, we train one controller for scalar BO and one for MOBO-GM using 70 BO iterations evaluated on hardware. In both settings, MOBO requires fewer iterations than the given budget to find satisfactory solutions. Thus, using a stopping criterion would reduce the total number of iterations. We estimate performance by averaging the return over 10 independent runs. To estimate robustness, we require that the controller stabilizes the system for a given delay or gain for 5 independent runs. A trial is deemed stable if $\alpha_t \in [-8°, 8°]$ and $\beta_t \in [-4°, 4°]$ for all $t \in [4, 5]$. Every training run lasts for 5 seconds. Fig. 2 and 3 show the fronts obtained by the MOBO-DM and MOBO-GM sim-to-real training, respectively (the different estimates are due to the stochastic simulations). The gray circles correspond to controllers that appeared stabilizing at first, but that were ruled out with longer simulations, cf. Sec. III-B. The green squares indicate controllers tested on hardware. To emphasize the generality of our method, they were selected to be approximately at the elbow of the front without further tuning.

Sim-to-real test. We test each controller learned in simulation on the hardware 5 times in 4 scenarios: (*i*) standard sim-to-real, (*ii*) sim-to-real adding Gaussian noise $\mathcal{N}(0, 0.5)$ to the motor voltage, (*iii*) sim-to-real adding noise to the encoder readings following a multinomial distribution over the integers in $[-4, 4]$ with $p(0) = 0.6$ and 0.05 everywhere else, and (*iv*) sim-to-real with the pendulum mass increased by 2 g. A run is a failure if $|\beta| > 20°$. In Table I, we compare the controllers in terms of average return, failure rate, and failing time, averaged over the runs that resulted in a failure. The robust methods consistently outperform the non-robust policy optimization across all test scenarios. It appears that the lower gain margin is a more suitable robustness indicator in this setting. This may be due to the fact that, in our experience, the gain margin is less noisy to estimate.

Hardware test. We test each controller learned in hardware 5 times in 4 scenarios: (*i*) extra mass of 5 g, (*ii*) extra mass of 9 g, (*iii*) extra mass of 10 g and extra pendulum length of 8 cm, and (*iv*) extra mass of 5 g with the actuation and sensor noise used in the sim-to-real experiments. Table II summarizes the test results. Similarly to the sim-to-real setting, the robust algorithm consistenlty outperforms its non-robust counterpart.

V. CONCLUDING REMARKS

We present a data-efficient algorithm for robust policy optimization based on multi-objective Bayesian optimization. We suggest a data-driven evaluation of two common robustness indicators, which is suitable to model-free settings. Our hardware experiments on a Furuta pendulum show that *(i)* our method facilitates simulation to real transfer, and *(ii)* consistently increases robustness of the learned controllers as compared to BO with a single performance objective. Our results indicate a promising avenue toward robust learning control by leveraging robustness measures from control theory and multi-objective Bayesian optimization and point to several directions for extensions. While we show that gain and delay margings are effective in practice on a mildly nonlinear system, they may not fully characterize robust stability in general [33], [4]. Thus, investigating other relevant robustness indicators that can efficiently be estimated from data in a model-free setting is a topic for future research. Also, using multiple robustness indicator simultaneously is relevant, which our method could do at the expense of a more complex scaling to balance robustness and performance.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] B. D. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

[3] J. Doyle, "Guaranteed margins for LQG regulators," *IEEE Transactions on Automatic Control*, vol. 23, no. 4, pp. 756–757, 1978.

[4] B. Boulet and Y. Duan, "The fundamental tradeoff between performance and robustness," *IEEE Control Systems Magazine*, vol. 27, no. 3, pp. 30–44, June 2007.

[5] K. Zhou and J. C. Doyle, *Essentials of robust control*. Prentice Hall, 1998.

[6] A. Saberi, S. Peddapullaiah, and B. M. Chen, *Loop transfer recovery: analysis and design*. Springer-Verlag, 1993.

[7] M. Green and D. J. Limebeer, *Linear robust control*. Courier Corporation, 2012.

[8] A. Nilim and L. El Ghaoui, "Robust control of markov decision processes with uncertain transition matrices," *Operations Research*, vol. 53, no. 5, pp. 780–798, 2005.

[9] R. Sharma and M. Gopal, "A robust markov game controller for nonlinear systems," *Applied Soft Computing*, vol. 7, no. 3, pp. 818–827, 2007.

[10] E. Delage and S. Mannor, "Percentile optimization for markov decision processes with parameter uncertainty," *Operations research*, vol. 58, no. 1, pp. 203–213, 2010.

[11] K. J. Astrom and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton, NJ, USA: Princeton University Press, 2008.

[12] M. Neumann-Brosig, A. Marco, D. Schwarzmann, and S. Trimpe, "Data-efficient auto-tuning with bayesian optimization: An industrial control study," *IEEE Transactions on Control Systems Technology*, 2019.

[13] H. K. Venkataraman and P. J. Seiler, "Recovering robustness in model-free reinforcement learning," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 4210–4216.

[14] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *International Conference on Machine Learning*, 2017, pp. 2817–2826.

[15] J. Morimoto and K. Doya, "Robust reinforcement learning," in *Advances in Neural Information Processing Systems*, 2001, pp. 1061–1067.

[16] A. Gambier and M. Jipp, "Multi-objective optimal control: An introduction," in *8th Asian Control Conference (ASCC), 2011*. IEEE, 2011, pp. 1084–1089.

[17] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of bayesian methods for seeking the extremum," *Towards global optimization*, vol. 2, no. 117-129, p. 2, 1978.

[18] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[19] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with gaussian processes," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 491–496.

[20] A. Marco, F. Berkenkamp, P. Hennig, A. P. Schoellig, A. Krause, S. Schaal, and S. Trimpe, "Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with bayesian optimization," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1557–1563.

[21] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian optimization for learning gaits under uncertainty," *Annals of Mathematics and Artificial Intelligence*, vol. 76, no. 1, pp. 5–23, Feb 2016.

[22] R. Antonova, A. Rai, and C. G. Atkeson, "Deep kernels for optimizing locomotion controllers," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 78, 2017, pp. 47–56.

[23] M. Emmerich and J.-w. Klinkenberg, "The computation of the expected improvement in dominated hypervolume of pareto front approximations," *Rapport technique, Leiden University*, vol. 34, 2008.

[24] M. Zuluaga, A. Krause, and M. Püschel, "e-PAL: An Active Learning Approach to the Multi-Objective Optimization Problem," *Journal of Machine Learning Research*, vol. 17, no. 104, pp. 1–32, 2016.

[25] D. Hernández-Lobato, J. Hernandez-Lobato, A. Shah, and R. Adams, "Predictive entropy search for multi-objective bayesian optimization," in *International Conference on Machine Learning*, 2016, pp. 1492–1501.

[26] R. Calandra, "Bayesian modeling for optimization and control in robotics," Ph.D. dissertation, Technische Universität, 2017.

[27] M. Tesch, J. G. Schneider, and H. Choset, "Expensive multiobjective optimization for robotics," *2013 IEEE International Conference on Robotics and Automation*, pp. 973–980, 2013.

[28] R. Ariizumi, M. Tesch, H. Choset, and F. Matsuno, "Expensive multiobjective optimization for robotics with consideration of heteroscedastic noise," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, sep 2014, pp. 2230–2235.

[29] B. Yang, G. Wang, R. Calandra, D. Contreras, S. Levine, and K. Pister, "Learning flexible and reusable locomotion primitives for a microrobot," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1904–1911, 2018.

[30] Y. Collette and P. Siarry, *Multiobjective optimization: Principles and case studies*. Springer Science & Business Media, 2013.

[31] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[32] M. A. Alvarez, L. Rosasco, and N. D. Lawrence, "Kernels for vector-valued functions: A review," *Foundations and Trends in Machine Learning*, vol. 4, no. 3, pp. 195–266, 2012.

[33] K. Zhou, J. C. Doyle, K. Glover *et al.*, *Robust and optimal control*. Prentice hall New Jersey, 1996, vol. 40.

[34] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic LQR tuning based on Gaussian process global optimization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2016, pp. 270–277.

[35] B. S. Cazzolato and Z. Prime, "On the dynamics of the furuta pendulum," *Journal of Control Science and Engineering*, vol. 2011, p. 3, 2011.

[36] Quanser, *Qube Servo 2 - Student workbook*, Quanser Consulting Inc.