Robust policy search for robot navigation

Javier Garcia-Barcos and Ruben Martinez-Cantin

***Abstract*— Complex robot navigation and control problems can be framed as policy search problems. However, interactive learning in uncertain environments can be expensive, requiring the use of data-efficient methods. Bayesian optimization is an efficient nonlinear optimization method where queries are carefully selected to gather information about the optimum location. This is achieved by a surrogate model, which encodes past information, and the acquisition function for query selection. Bayesian optimization can be very sensitive to uncertainty in the input data or prior assumptions. In this work, we incorporate both robust optimization and statistical robustness, showing that both types of robustness are synergistic. For robust optimization we use an improved version of *unscented Bayesian optimization* which provides safe and repeatable policies in the presence of policy uncertainty. We also provide new theoretical insights. For statistical robustness, we use an adaptive surrogate model and we introduce the *Boltzmann selection* as a stochastic acquisition method to have convergence guarantees and improved performance even with surrogate modeling errors. We present results in several optimization benchmarks and robot tasks.**

I. INTRODUCTION

Robot navigation in uncertain environments can be framed as a policy search problem [1], a technique that has led to important achievements in robotics [2], [3], [4], [5]. Those results had been obtained using different flavours of gradient-based policy search, which might require a large number of trials and a good initialization to avoid suboptimal results in local minima. Trials for robotic applications come at a substantial cost, as each one requires to move and interact with a robot. Alternatively, high-performance robotic simulators still require a fair amount of computational resources. It is thus evident that sample efficiency is of paramount importance.

Active policy search uses Bayesian optimization to drive the search for optimality in an effective way. Bayesian optimization is a sample efficient method for nonlinear optimization that does not require gradients or good initialization to obtain global convergence [6]. The probabilistic nature of Bayesian optimization allows the use of partial or incomplete information, analogous to the stochastic gradient descent commonly used in classical policy search [1]. In fact, Bayesian optimization has been already used in some robotics and reinforcement learning setups, such as robot walking [7], [8], [9], control [10], [11], planning [12], [13], grasping [14], [15], [16] and damage recovery [17].

Bayesian optimization relies on a probabilistic surrogate model of the target function, typically a Gaussian process. In the original formulation, this model is incorporated for sample efficiency as it provides a *memory* of previous trials [18]. The surrogate model can also be exploited for other purposes that can be useful in robotics or reinforcement learning scenarios, such as, guaranteeing a minimum outcome [19], detect and remove outliers [20] or incorporate prior information [17].

Javier Garcia-Barcos and Ruben Martinez-Cantin are with the Instituto de Investigacion en Ingenieria de Aragon (I3A), University of Zaragoza, Spain. jgbarcos,rmcantin@unizar.es

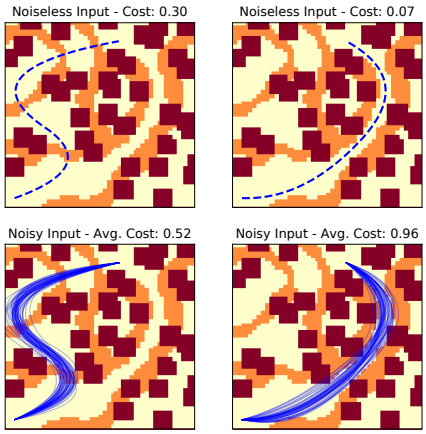

Fig. 1. Path planning on uneven terrain with obstacles, with different trajectories displayed (left and right). The orange regions represent slopes with a higher traversing cost. The red rectangles are obstacles. Top: the desired trajectories (blue dashed line). Bottom: possible deviations (blue lines) from desired trajectories due to input noise. The right trajectory is more efficient without input noise. Once we take into account input noise, it becomes unsafe as it can collide with obstacles easily. The left trajectory is safer in the presence of input noise despite being less efficient.

In this work, we are exploiting the probabilistic features of Bayesian optimization to provide *robustness* to policy search. First, we imply robustness in the sense of robust optimization or robust control. The resulting policy should perform well even if the robot or agent is not able to follow the policy with enough precision. For example, consider the navigation problem from Figure 1, although the right trajectory is shorter and cost-efficient, it is also riskier. If the trajectory uncertainty increases, as in the bottom plots, it becomes unsafe and incurs a higher cost on average. On the contrary, the left trajectory is longer and less efficient, but it is also safer. For a particular level of uncertainty, it becomes a safer and more efficient route on average. If we think on the cost function in terms of the policy parameters, the left trajectory lies in a smooth flat region while the right trajectory lies in a high variability region with a narrow valley. Intuitively, in the presence of location uncertainty or safety concerns, we want to avoid narrow optima where perturbations might push the solution to poor performance results. Instead, we focus on a broad optima where the solution is optimal even after perturbation. However, depending on the task and environmental conditions, the algorithm should be able to

model and select between narrow and flat optimum regions. In *robust optimization* and *robust control*, the performance is usually maintained in a bounded region or set. In this work, we replace that region by a probability distribution (e.g.: Gaussian), because that is the typical representation for uncertainty in robot location. Thus, we focus on optimizing the averaged performance, while classical robust optimization optimizes the worst case scenario. We have developed a variant of Bayesian optimization that relies on the unscented transformation to consider the expected policy performance under uncertainty. One advantage of our Unscented Bayesian Optimization is that it can be used just for safety reasons –for example, if we are able to train in a simulator with perfect repeatability but we want to consider possible perturbations when executed in the real robot– or if the policy uncertainty comes from noisy or perturbed trials –the policy parameters are perturbed during training–. In the experiments in this paper, we consider the most challenging scenario of having perturbations also during training.

It has been studied that reward functions found in robotics might be difficult to approximate by typical surrogate models in Bayesian optimization, resulting in unreliable performance of the optimizer [21]. Furthermore, policy perturbations during training, as discussed before, can also be problematic for the surrogate model and the optimizer performance. Therefore, in this work, we have also included another layer of *robustness* to our proposal. *Robust statistics* deals with statistical methods that perform reasonably well even when the underlying assumptions are somehow violated. For example, convergence of Bayesian optimization methods is based on the assumption that the target function *belongs* to the reproducing kernel Hilbert space spanned by the Gaussian process kernel. Bayesian optimization relies on optimal decision theory to actively select the next informative trial. When combined with an inadequate model, this can lead to poor results and lack of convergence. In this work, we employ several strategies to provide robustness in a statistical sense. First, we use an adaptive kernel for nonstationary environments [21]. This provides a much more flexible surrogate model to accommodate a larger set of target functions, such as common reward functions. Our method also selects new trials based on *Boltzmann selection* that can be robust to surrogate modeling errors [22]. The intuition behind the Boltzmann selection is to select new policies based on a softmax of the acquisition function –instead of the standard optimum–. An advantage of this approach is that it can be used to derive convergence bounds without assuming artificially injected exploration. Another advantage of the Boltzmann selection is that they trivially allow to perform distributed Bayesian optimization in a multi-robot setup or using a simulator. Figure 2 shows the different components of our approach.

We present a new architecture for robust efficient policy search and we provide new theoretical analysis and insights of the methods employed. Specifically, we introduce the first robust policy search both in terms of robust optimization and statistical robustness. Furthermore, policy search is performed using episodic Bayesian optimization, which

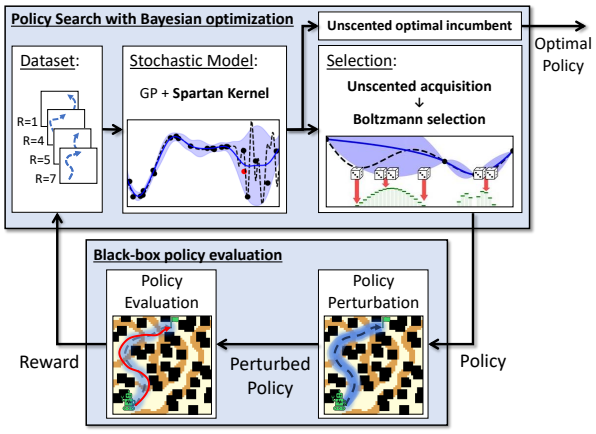

Fig. 2. Diagram showing the different components of our approach, based on policy search with Bayesian optimization. It depicts the Bayesian optimization loop (top) applied to a policy search problem (bottom). The goal is to identify the most efficient policy by sequentially querying different policies and obtaining the corresponding reward. However, as the problem has policy uncertainty, a perturbed policy will be evaluated instead. We highlight (bold text) the elements that differ from a standard policy search with Bayesian optimization: the *Spartan kernel* to model nonstationarity, the unscented transform applied in *unscented optimal incumbent* and *unscented acquisition*) to propagate the policy uncertainty and, instead of greedy acquisition function maximization, *Boltzmann selection* sampling to improve exploration in the presence of surrogate modeling errors.

requires a small number of trials to obtain optimal results. Our extensions maintain the data efficiency of standard Bayesian optimization. Our algorithm combines two novel methods: **Unscented Bayesian Optimization** and **Boltzmann selection** (preliminary versions were published in [14], [22]) with other ingredients: expected improvement and Gaussian processes with adaptive kernels to guarantee optimal and stable solutions even under broken assumptions, biased priors, query perturbations, etc. As secondary contributions: 1) we design the first fully distributed robust policy search algorithm, which allows parallel evaluations in multirobot systems and simulators. 2) we provide a theoretical interpretation of the unscented Bayesian optimization as an integrated response method with polynomial complexity and a formulation based on the scaled unscented transform, which provides more flexibility to define the safety/stability region.

II. BACKGROUND

A. Active policy search

Policy search consists of finding the optimal parameters $\mathbf{x}^*$ of a policy $\pi_{\mathbf{x}}(\mathbf{a}_k|\mathbf{s}_k)$ with respect to the expected return U^π, denoted $\mathbb{E}[U^\pi] = \mathbb{E}[\sum_{k=1}^{M} \gamma^k R^\pi(\mathbf{s}_k, \mathbf{a}_k)]$. Here, a denotes the action, s the state, γ the discounted factor and M the episode length. Without loss of generality, we assume finite horizon on episodic policy search. The expectation is under the policy and the system dynamics which together form a distribution over trajectories τ. If we use an episodic formulation, such as REINFORCE [1], the expectation is usually approximated from Monte-Carlo rollouts $\tau^{(i)} \sim \tau$ of the robot trajectory. In this setup, finding the optimal policy parameters can be framed as a pure optimization problem,

where the objective function is then computed as:

$$f(\mathbf{x}) = \mathbb{E}_\tau[U^\pi] \approx \sum_{i=1}^{N}\sum_{k=1}^{M} \gamma^k R\left(\tau_k^{(i)}\right)$$
$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} \mathbb{E}_\tau[U^\pi] \tag{1}$$

where $\mathbf{x}^*$ are the parameters of the optimal policy $\pi^* = \pi_{\mathbf{x}^*}$, $R(\tau_k^{(i)})$ is the instantaneous reward at time step k following rollout $\tau^{(i)}$ and N is the number of rollouts. Active policy search [23] computes the optimal policy parameters using Bayesian optimization. Similarly to stochastic gradient descent in gradient-based policy search, Bayesian optimization can directly be applied to stochastic optimization thanks to the probabilistic surrogate model [24]. Therefore, the expectation in equation (1) can be approximated with a small batch of rollouts or even a single episode. Algorithm 1 summarized the active policy search strategy. Section II-B details the steps of updating the surrogate model and generating the next set of policy parameters $\mathbf{x}_{t+1}$.

Algorithm 1 Active Policy Search

Input: Optimization budget T

1: Initialize $\mathbf{x}_1$ based on a low discrepancy sequence.
2: **for** each optimization iteration t until budget T **do**:
3: Generate episode $\tau_{\mathbf{x}_t} \sim \{s_0, a_0, R_1, s_1, a_1, \ldots\}$
4: $y_t \leftarrow \sum_{k=1}^{M} \gamma^k R_k$
5: Add $(\mathbf{x}_t, y_t)$ to surrogate model with equation (3)
6: Generate $\mathbf{x}_{t+1}$ using equation (2)
7: **end for**

B. Bayesian optimization

Bayesian optimization is a framework that aims to efficiently optimize noisy, expensive, blackbox functions. It uses two distinct components: a *probabilistic surrogate model* $p(f)$ that learns the properties and features of the target function using previously evaluated observations and an *acquisition function* $\alpha(x, p(f))$ that, based on the surrogate model, builds an utility function which rates how promising a subsequent query could be. Although our contributions are agnostic to these choices, for the remainder of the paper, the discussion and results are based on the use of a *Gaussian process* as the surrogate model and the *expected improvement* as the acquisition function because they are the most commonly used in the literature due to their excellent performance in a large variety of problems.

Formally, Bayesian optimization attempts to find the global optima of an expensive unknown function $f : \mathcal{X} \to \mathbb{R}$ over some domain $\mathcal{X} \subset \mathbb{R}^d$ by sequentially performing queries. At iteration t, all previously observed values $\mathbf{y} = y_{1:t}$ at queried points $\mathbf{X} = \mathbf{x}_{1:t}$ are used to learn a probabilistic surrogate model $p(f|y_{1:t}, \mathbf{x}_{1:t})$. Typically, the next query $\mathbf{x}_{t+1}$ is then determined by greedily optimizing the acquisition function in $\mathcal{X}$:

$$\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha\left(\mathbf{x}, p(f \mid y_{1:t}, \mathbf{x}_{1:t})\right) \tag{2}$$

although we will replace the greedy selection in Section IV-A.

a) Surrogate Model: The most common surrogate model is the Gaussian process (GP). For the remainder of the paper we consider a GP with zero mean and kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. The GP posterior model allows predictions at query points $\mathbf{x}_q$ which are normally distributed $y_q \sim \mathcal{N}(\mu(\mathbf{x}_q), \sigma^2(\mathbf{x}_q))$, such that:

$$\mu(\mathbf{x}_q) = \mathbf{k}(\mathbf{x}_q)^T \mathbf{K}^{-1}\mathbf{y}$$
$$\sigma^2(\mathbf{x}_q) = k(\mathbf{x}_q, \mathbf{x}_q) - \mathbf{k}(\mathbf{x}_q)^T \mathbf{K}^{-1}\mathbf{k}(\mathbf{x}_q) \tag{3}$$

where $\mathbf{k}(\mathbf{x}_q) = [k(\mathbf{x}_q, \mathbf{x}_i)]_{\mathbf{x}_i\in\mathbf{X}}$ and $\mathbf{K} = [\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)]_{\mathbf{x}_i,\mathbf{x}_j\in\mathbf{X}} + \mathbf{I}\sigma_n^2$. For the kernel, we have used the Spartan kernel which provides robustness to nonstationary function and improves convergence, which has been shown to be critical for reinforcement learning problems [25].

b) Kernel function: The Spartan kernel [25] is the combination of several local kernels k_l applied over moving regions –defined by weighting functions $\omega_l(\mathbf{x}'|\boldsymbol{\theta}_p)$–, with a global kernel k_g for the rest of the space –defined by weight $\omega_g(\mathbf{x})$–. Each weighting functions follow a normal distribution:

$$\omega_g = \mathcal{N}\left(\psi, \mathbf{I}\sigma_g^2\right),$$
$$\omega_l = \mathcal{N}\left(\boldsymbol{\theta}_p, \mathbf{I}\sigma_l^2\right) \qquad \forall\, l = 1 \ldots M \tag{4}$$

where ψ and $\boldsymbol{\theta}_p$ can be seen as the center of the influence region of each kernel while σ_g and σ_l can be interpreted as the size of each area of influence. The regions of the local kernels are centered in a single point $\boldsymbol{\theta}_p$ with multiple diameters, creating a funnel structure. In order to achieve smooth interpolation between regions, we use normalized weights $\lambda_j(\mathbf{x}) = \sqrt{\omega_j(\mathbf{x}) / \sum_p \omega_p(\mathbf{x})}$:

$$k_S(\mathbf{x}, \mathbf{x}'|\boldsymbol{\theta}_S) = \lambda_g(\mathbf{x})\lambda_g(\mathbf{x}')k_g(\mathbf{x}, \mathbf{x}'|\boldsymbol{\theta}_g) + \sum_{l=1}^{M} \lambda_l(\mathbf{x}|\boldsymbol{\theta}_p)\lambda_l(\mathbf{x}'|\boldsymbol{\theta}_p)k_l(\mathbf{x}, \mathbf{x}'|\boldsymbol{\theta}_l) \tag{5}$$

In the experiments, we have used Matérn kernels with automatic relevance determination for k_g and k_l [26]. For the local kernels, we estimate the center of the funnel structure $\boldsymbol{\theta}_p$ based on the data gathered. Thus, we consider $\boldsymbol{\theta}_p$ as part of the hyperparameters jointly with the Matérn hyperparmeters $\boldsymbol{\theta}_g$ and $\boldsymbol{\theta}_l$:

$$\boldsymbol{\theta}_S = [\boldsymbol{\theta}_g, \boldsymbol{\theta}_{l_1}, \ldots, \boldsymbol{\theta}_{l_M}, \boldsymbol{\theta}_p] \tag{6}$$

c) Hyperparameter estimation: In many GP applications, including Bayesian optimization, kernel hyperparameters are estimated using the *empirical Bayes* approach. In that case, a point estimate like the maximum likelihood or maximum a posteriori is used, resulting in an overconfident estimate of the GP uncertainty [26]. Instead, we use a *fully Bayesian* approach based on Markov chain Monte Carlo (MCMC) to generate a set of samples $\{\boldsymbol{\theta}_i\}_{i=1}^N$ with $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta}|\mathbf{y}, \mathbf{X})$. In particular, we use the slice sampling algorithm which has already been used successfully in Bayesian optimization [27].

d) Acquisition Function: The expected improvement (EI) [28] is a standard acquisition function defined in terms of the query improvement at iteration t and is defined as:

$$EI_t(\mathbf{x}) = (\rho_t - \mu_t)\,\Phi(z_t) + \sigma_t \phi(z_t) \tag{7}$$

where ϕ and Φ are the corresponding Gaussian probability density function (PDF) and cumulative density function (CDF), being $z_t = (\rho_t - \mu_t)/\sigma_t$. In this case, (μ, σ^2) are the prediction parameters computed with (3) and $\rho_t = \max(y_1, \ldots, y_t)$ is the incumbent optimum at that iteration.

III. ROBUST OPTIMIZATION

Robust optimization is the field of optimization that deals with uncertainty in the parameters of the problem itself or its solution. Specifically, local robustness guarantees that the optimality of the solution is valid even after small perturbations of the solution. Usually, these perturbations are defined in terms of a valid set or region. However, in robotics, perturbations such as location error are represented with probabilistic distributions. Thus, instead of selecting the point that optimizes a single outcome, we select the point that optimizes an *integrated outcome*:

$$g(\mathbf{x}) = \int_{\mathcal{X}} f(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} = \mathbb{E}_{p(\mathbf{x})}[f(x)] \tag{8}$$

where $f(\mathbf{x})$ is the expected utility from equation (1) and $p(\mathbf{x})$ corresponds to the probability associated with the local perturbations. It can be interpreted objectively as noise on the input variables $\mathbf{x}$ or, subjectively, as a probabilistic representation of the local stability or safety region. That is, a region that guarantees good results even if the query is repeated several times. Instead of $f(\cdot)$, the *integrated outcome* $g(\cdot)$ becomes the function that will be optimized. For the remainder of the paper, we assume that the perturbations –or safety region– are normally distributed, that is, $p(\mathbf{x}) = \mathcal{N}(\mathbf{0}, \Sigma_{\mathbf{x}})$.

Classical robust optimization, considering the worst case scenario, has been previously studied in the context of Bayesian optimization [29]. Input noise has been addressed to find narrow optima despite query perturbations [30].

In this paper, we present an integrated response method based on the unscented transformation, which serves as a cheap and scalable numerical integration method. A preliminary version of this method was published in Nogueira et al. [14]. In this case, we use a flexible variant of the unscented transformation, called the scaled unscented transformation [31], to allow more control on the stability region and avoid numerical issues. Furthermore, previous preliminary work [14] interpreted $p(\mathbf{x})$ as a subjective safety region of stability without actual input noise during training. In this case, we consider the more challenging scenario of both stability and input noise, where the objective is not only to find a broad maximum, but queries are also perturbed $\mathbf{x} \pm \Delta\mathbf{x}$ during training.

A. Scaled unscented Bayesian optimization

The unscented transformation is a method to propagate probability distributions through nonlinear transformations with a trade off between computational cost and accuracy. The unscented transformation uses a set of deterministically selected samples from the original distribution (called *sigma points*) and transforms them through the nonlinear function $f(\cdot)$. Then, the transformed distribution is computed based on the weighted combination of the transformed sigma points:

$$\mathcal{X}_{0:d} = \left\{ \bar{\mathbf{x}},\ \bar{\mathbf{x}} \pm \left(\sqrt{(d+\gamma)\Sigma_{\mathbf{x}}} \right)_i \right\} \qquad \forall\, i = 1 \ldots d \tag{9}$$

where $(\sqrt{\cdot})_i$ is the i-th row or column of the corresponding matrix square root, $\gamma = \alpha^2(2d + \kappa) - d$ and $\Sigma_{\mathbf{x}}$ is the covariance matrix of the perturbation probability distribution. The weight for the initial point is $w^0 = \frac{\gamma}{d+\gamma}$ and $w^{(i)} = \frac{1}{2(d+\gamma)}$ for the rest. The parameters should follow $\kappa \geq 0$ and $0 \leq \alpha \leq 1$, while the standard unscented transformation is a special case for $\alpha = 1$. As pointed out by van der Merwe [32], we recommend a $\kappa = 0$ and α close to 1. For the matrix square root function, we use the Cholesky decomposition for its numerical stability and robustness [32].

Using the unscented transform, we can approximate $\mathbb{E}_{p(x)}[f(x)] \approx \sum_{i=0}^{2d} f(\mathcal{X}_i) w^i$. The unscented transformation is used twice in our algorithm. First, we need to drive the queries towards points where the improvement is very likely within the safety or perturbed region. Thus, we apply the unscented transformation to the acquisition function, resulting in the *unscented expected improvement*, that is, $UEI(x) = \sum_{i=0}^{2d} EI(\mathcal{X}_i) w^i$. Intuitively, the UEI will be large when the expected improvement is also large within a region –for example, if the optimum is flat– and UEI will be small when the expected improvement is high only in a small region –like a narrow optimum–. However, when exploring, the algorithm might query and select narrow optima by chance. Thus, the incumbent ρ_t cannot be chosen solely based on observed values because we might end up selecting an unstable solution. Therefore, we further compute the *unscented optimal incumbent* to select the most stable optimum $\rho_t = \max_{\mathbf{x} \in \mathbf{X}} \mathbb{E}_{p(x)}[f(x)] \approx \max_{\mathbf{x} \in \mathbf{X}} \sum_{i=0}^{2d} f(\mathcal{X}_i) w^i$. Note that, in the context of Bayesian optimization, we want to reduce the number of evaluations of $f(\mathbf{x})$. Therefore, the sigma points are evaluated in the surrogate model, using the GP mean function as an approximation of the target function $\mu(\mathbf{x}) \approx f(\mathbf{x})$. For that, we approximate the integrated outcome $\widehat{g}(\mathbf{x}) = \sum_{i=0}^{2d} \mu(\mathcal{X}_i) w^i$, where $\mu(\mathbf{x})$ is obtained from equation (3). We define the unscented optimal incumbent as $\widetilde{\rho}_t = \max_{\mathbf{x} \in \mathbf{X}} \widehat{g}(\mathbf{x})$, that can be used in equation (7). Most importantly, we select the final optimal incumbent $\widetilde{\rho}_T \approx \max_{\mathbf{x} \in \mathbf{X}} \mathbb{E}_{p(\mathbf{x})}[f(x)]$ as the output of the optimization process.

B. The Unscented transform as integration

The unscented transform can be interpreted as a probabilistic integration method. The unscented transformation with $\alpha = 1$ and $\kappa = 0$ is equivalent to the three point Gauss-Hermite quadrature rule [32]. While the Gauss-Hermite

method computes the integral *exactly* under the Gaussian and polynomial assumptions, it has a cost of $\mathcal{O}(n^d)$ where n is the polynomial order of the function in the region. Meanwhile the unscented transform, has a quadratic cost $\mathcal{O}(d^2)$ for computing the integrated response [32]. The low cost of the unscented transformation is also an advantage compared to other more advanced integration methods such as Monte Carlo or Bayesian quadrature, which have higher computational cost. Note that, during optimization the integrated outcome $g(\mathbf{x})$ is always approximated with respect to the Gaussian process mean function $\mu(\mathbf{x})$ to avoid increasing the number of queries of $f(\mathbf{x})$. Therefore, the integral would be as accurate as the Gaussian process with respect to the target function. We found that, in practice, it is more efficient to employ the computational resources to improve the surrogate model (for example, using MCMC on the kernel hyperparameters), than to provide a better integrated outcome.

IV. STATISTICAL ROBUSTNESS

In this section, we focus on mitigating the effect of surrogate modeling errors. In Bayesian optimization, the exploration and exploitation is guided by the surrogate model. By selecting a specific surrogate model (specific GP kernels, Bayesian neural network architectures...) we introduce some assumptions that the target function might not satisfy. Having input noise during the optimization is another source of misleading observations, as the observed query will deviate from the intended query. In practice, a biased or overconfident model produces limited exploration or erroneous exploitation, resulting in even more biased data. This might result in lack of convergence. We propose using an alternative acquisition method, called *Boltzmann selection*, to enhance exploration and provide statistical robustness.

A. Boltzmann selection

As a sequential decision making process, we can interpret the Bayesian optimization framework as a *partially observable Markov decision process* (POMDP) [33], [22]. In this interpretation, the state is the target function, the action is the next query point, the belief is the surrogate model and the action-value (Q-function) is the acquisition function for each possible query. Note that this POMDP model would represent the actual learning/optimization process during policy search. Therefore, equation (2) can be interpreted as a *meta-policy*, since it is used on a higher abstraction level to learn the actual robot policy $\pi_{\mathbf{x}}$. We can see that the Bayesian optimization meta-policies found in the literature are greedy, that is, they select the single action or next query that maximizes the acquisition function or *Q-function*.

Our approach consist on replacing the greedy policy of equation (2) with a stochastic policy such as the Boltzmann policy (also known as Gibbs or softmax policy):

$$p(\mathbf{x}_{t+1} \mid y_{1:t}, \mathbf{x}_{1:t}) = \frac{e^{\beta_t \alpha(x_{t+1}, p(f \mid y_{1:t}, \mathbf{x}_{1:t}))}}{\int_{x \in \mathcal{X}} e^{\beta_t \alpha(x, p(f \mid y_{1:t}, \mathbf{x}_{1:t}))} dx} \quad (10)$$

which defines a probability distribution for the next query or action [22]. Thus, the actual next query is selected by sampling that distribution $\mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1} \mid y_{1:t}, \mathbf{x}_{1:t})$. This policy allows exploration even if the model is completely biased or overconfident, increasing the performance in the presence of modeling errors. This approach can be applied to any acquisition function or surrogate model that can be found in the literature. Since it relies on sampling from a Boltzmann policy, we refer to this query selection approach as the *Boltzmann selection*.

Theoretical analysis shows that the standard *greedy expected improvement* policy may not converge for unknown GP hyperparameters. Instead, in order to guarantee near-optimal convergence rates, it is necessary a *greedy in the limit with infinite exploration* (GLIE) policy [34], [22]. A simple and well-known GLIE policy is the ϵ-greedy strategy. This was the policy used to obtain the convergence rates [34], but it is highly inefficient and it is never used in practice. Selecting an appropriate β_t sequence, the policy from equation (10) is GLIE by construction [22]. Thus, the Boltzmann policy used in this work is the first policy based on the *expected improvement* that has both good performance in practice and guaranteed theoretical convergence.

B. Distributed Bayesian optimization

A secondary advantage of using the Boltzmann selection is that they also trivially enable distributed optimization, where different policy parameters can be evaluated in parallel in a fully distributed fashion. This could be applied in multi-robot scenarios or for simulation-based reinforcement learning. Many parallel methods for Bayesian optimization have been proposed in the past few years with heuristics to enforce diverse queries. Some authors include artificially augmented data by hallucinated observations [35], [27], combine optimization with some degree of active learning in order maximize the knowledge about the target function [36], [37], [38] or enforce spatial coverage [39]. All these methods have in common the need for a central node that shares and synchronizes the data and the queries to enforce diversity in the parallel runs.

Sampling using this Boltzmann selection already ensures diverse queries with random numbers [22]. Thus, a centralized node is not required and all computation can be done in each node in a fully distributed manner. In terms of communication, the nodes only need to broadcast their latest evaluated query and observation value $\{\mathbf{x}_t, y_t\}$, requiring minimal communication bandwidth. Communication can be asynchronous and it is robust to delays or failures in the network, as the order of the queries and observations is irrelevant.

C. Baseline surrogate model

Surrogate model selection introduces some assumptions that might not be satisfied in the problem at hand. This might result in a biased model as discussed before. This effect can be mitigated by choosing an expressive and flexible model, compatible with Bayesian optimization. In Gaussian processes, model expressiveness is related to the choice of kernel, which also encodes the prior information about

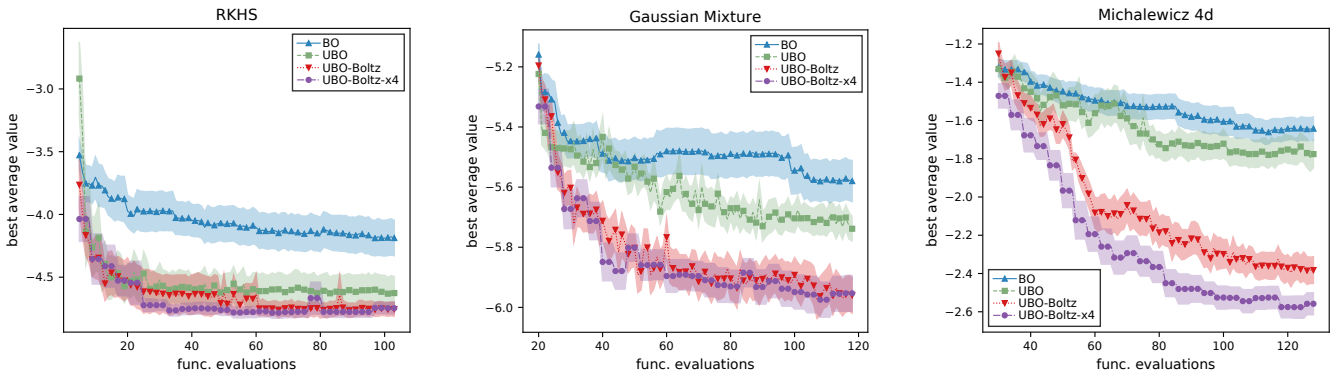

Fig. 3. Benchmark functions optimization results. In general, UBO is able to find a more stable solution than the vanilla BO, resulting in a better average value. However, using Boltzmann selection results in an improved stability. Parallelized runs had a much lower walltime without a penalty in performance.

the function space. The most frequent kernels in Bayesian optimization are stationary kernels $k(x, x') = k(|x - x'|)$ like the Matérn kernel. However, robotic and reinforcement learning settings typically showcase nonstationarity [21], therefore a nonstationary kernel is needed. Consider again the example from Figure 1. Given that the optimum can be either trajectory depending on the trajectory uncertainty level, our optimization algorithm must be able to model both if needed. Thus, it must be able to model the shape or both narrow and broad optima in different regions. This is known to be problematic in Bayesian optimization. For that reason, we have incorporated the Spartan kernel [25] as presented in Section II-B. Furthermore, we estimate the kernel hyperparameters using MCMC. Contrary to maximum likelihood or maximum a posteriori estimates which tend to underestimate the model uncertainty, MCMC allows a more flexible model. Note that both unscented Bayesian optimization and the Boltzmann selection are agnostic of the kernel or estimation algorithm choice. By selecting a flexible surrogate as a baseline, we highlight the importance of Boltzmann selection even with limited bias.

V. RESULTS

In this section we describe the experiments used to compare the performance of different Bayesian optimization methods in the presence of input noise. We compare a vanilla implementation of Bayesian optimization (BO), the unscented Bayesian optimization with a greedy policy (UBO) and the unscented Bayesian optimization with the Boltzmann selection (UBO-Boltz). We also compare with a parallel version of the Boltzmann selection with 4 nodes (UBO-Boltz-x4) to study the performance impact of adding parallelization. Note that in the results we show the number of function evaluations, not iterations. For example, at the 20 evaluation, the UBO-Boltz-x4 method had run only for 5 iterations, therefore requiring less wall time and using only the previous information from 16 trials instead of 19.

All methods share the same configuration as a baseline: expected improvement (EI) as the acquisition function and a Gaussian process as the surrogate model with the Spartan kernel and MCMC for the hyperparameters. Given that EI is known to be unstable during the first iterations due to lack of information [18], [34], the optimization is initialized with p evaluations from a low discrepancy Sobol sequence.

The performance of each method was evaluated in the following way: For every function evaluation $\mathbf{x}_t$, each method computes their best solution (the optimal incumbent ρ_t or the unscented optimal incumbent $\widetilde{\rho}_t$) using the observations $(\mathbf{x}_{1:t}, y_{1:t})$ and according to their model at that point of the optimization. Then, we evaluate the integrated outcome at the best solution $g(\widetilde{\mathbf{x}}_t)$ by approximating (8) using 1000 Monte Carlo samples from $p(\mathbf{x})$ over the actual function $f(\cdot)$. For the plots, we repeat each optimization 20 times and display the mean value with 95% confidence interval. Common random numbers were used for all the methods. We assume isotropic input noise, reported as σ such that $\Sigma_x = I\sigma^2$ from (9). The input space is normalized on all the problems between 0 and 1, so the reported input noise σ is already normalized.

A. Benchmark Optimization Functions

We have evaluated the methods on synthetic benchmark functions for optimization. We have used the functions previously used in the BO literature: the RKHS function [40] and a Mixture of 2D Gaussian distributions (GM) [14]. These functions have unstable global optima for certain levels of input noise. This means that in order to locate the safe optima, we need to model and take into account the input noise. We have also used a 4D Michalewicz function[1], a popular test problem for global optimization because of its sharp edges and the large number of local optima. All benchmark functions use input noise $\sigma = 0.02$ and 100 evaluations. The number of initial samples is set based on the dimensions of each problem to 5, 20 and 30 samples for RKHS, GM and Michalewicz.

Figure 3 shows the results on the benchmark functions. Although UBO finds better stable optima than BO, using the Boltzmann selection further improves the performance. It also shows that adding parallelization barely impacts the optimization results. This means that we can achieve better performance and wall-time using the parallel approach.

[1]https://www.sfu.ca/ ssurjano/optimization.html

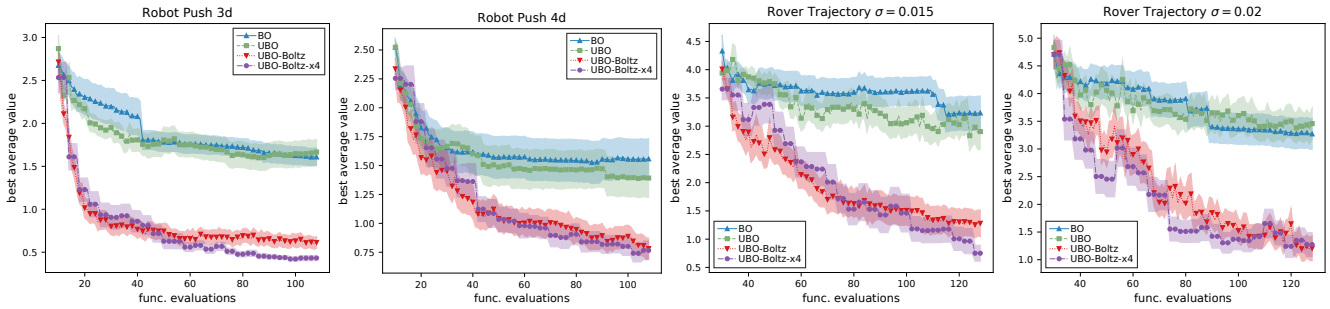

Fig. 4. Robot pushing problem and rover path planning optimization results. For the more complex problems, the UBO is not able to find a stable solution, unlike the Boltzmann selection.

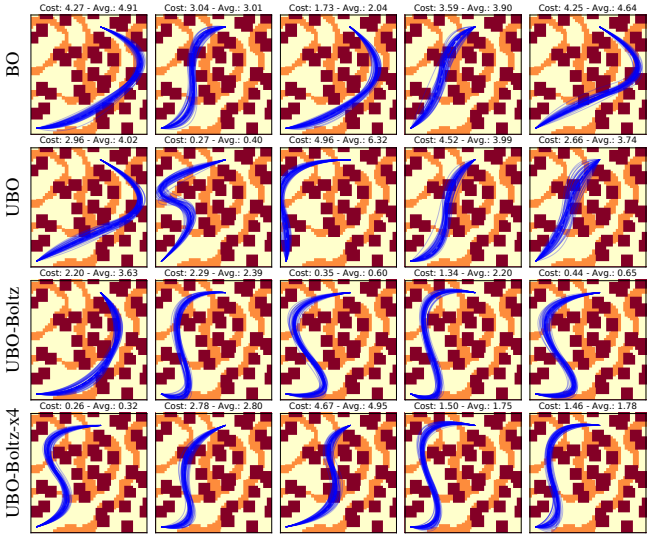

Fig. 5. Examples of optimized trajectories found by different methods (rows) and trials (columns), showing the possible deviations from the trajectories by simulating input noise $\sigma = 0.02$. We display the cost of the desired trajectory (assuming no input noise) and the average cost from possible deviations over each result. Note that BO (first row) does not find the safe path in any of the trials. The best trial is found by UBO (2nd row and column: noiseless cost 0.27, noisy cost 0.40), however most of the time lacks exploration to find a good one. The Boltzmann selection (serial in row 3 and parallel in row 4) improves exploration allowing better results overall.

B. Robot Pushing

Next, we have used the *active learning for robot pushing* setup and code from Wang et al. [41]. The task is to push an object towards a designated goal location. In the 3D problem, the policy parameters are the robot location $(r_x, r_y) \in [-5, 5]$ and pushing duration $t_r \in [1, 30]$, while pushing direction r_θ towards the object is assumed to be known. In the 4D problem, $r_\theta \in [0, 2\pi)$ is included as the fourth parameter. The problem is simulated in a 2D environment with a circular robot and a square object. The robot follows a constant linear motion from (r_x, r_y) in r_θ direction during t_r timesteps and the simulation is run until both object and robot reach zero velocity. The policy cost is the distance between the object and the designated goal. In both functions, we use 10 initial queries and a budget of 100 function evaluations during optimization. The 3D version uses $\sigma = 0.02$ while the 4D version uses $\sigma = 0.01$. We reduced the input noise in the 4D function because the robot angle parameter r_θ is very sensitive to input noise, as a small change in direction of the robot might result in completely missing the goal.

Figure 4, shows the Robot Pushing results. Contrary to benchmark results, UBO no longer guarantees stable solutions. Instead, it shows that Boltzmann selection is required to achieve good results. Parallel performance remains similar.

C. Robot Path Planning

In this section we cover the problem of safe path planning. The objective is to find a stable and efficient trajectory of a rover through a rugged terrain with multiple obstacles. It is based on *optimizing rover trajectories* from Wang et al. [42]. In this case, there are 4 policy parameters within the $[0, 1]$ interval, a pair of 2D reference points for a trajectory defined by a quadratic B-Spline with smoothness condition set to 0 (to enforce it to go through the reference points). The robot has to perform path planning while avoiding obstacles, which might be dangerous for the rover to collide with, and steep slopes, which might be dangerous as the rover can tip over. The cost for traversing obstacles or getting out of bounds is 20.0, for slopes is 1.0 and a constant cost of 0.05 for penalizing trajectory length. The overall policy cost is computed by integrating along the trajectory. In the figures, obstacles are red rectangles and slopes are orange regions. We are interested in finding stable trajectories that avoid the danger that might arise from trajectory deviations. This is a common problem in robot navigation as localization errors might result in the robot not following the desired trajectory accurately [23], [12].

We study this problem using 2 different input noises: $\sigma = 0.015$ and $\sigma = 0.02$. We use 30 initial samples and 100 function evaluations during optimization. Figure 4 shows the resulting optimization performance of each method. Figure 5 shows some trajectories obtained using different methods. We can see how both BO and UBO are prone to get stuck in suboptimal trajectories as solutions while Boltzmann selection methods return a solution closer to the safe trajectory.

VI. CONCLUSIONS

In this paper, we propose the first active policy search algorithm that is robust both in a statistical and optimization point of view. For achieving both types of robustness, we use

multiple techniques, such as, the unscented transformation to deal with input noise and local perturbations, a Boltzmann selection to enhance the exploration and convergence guarantees of the acquisition function. Therefore, this paper presents several contributions that provide synergistic results. First, we have presented a new formulation and interpretation of the unscented Bayesian optimization algorithm and shown its robustness both for safety/stability conditions and against small perturbations in the queries. Second, for statistical robustness, we have used a Boltzmann selection of the acquisition function for actively preventing surrogate bias and guaranteeing near-optimal convergence. This further highlights previous results that indicates that the ubiquitous greedy strategy in the Bayesian optimization literature can be suboptimal in many applications. In the experiments, we have used an an adaptive Gaussian process with the Spartan kernel for modeling the nonstationarity of reward/cost functions in robotic applications, to guarantee a strong and flexible baseline. The method has been evaluated on several benchmark functions and robotic applications which showcase the influence of input noise, such as safe robot navigation. We also take advantage of the embarrassingly parallel nature of the Boltzmann selection that could be used in multi-robot setups or simulation environments.

ACKNOWLEDGMENT

This work was partly supported by projects RTI2018-096903-B-I00 (MINECO/FEDER) and RED2018-102511-T.

REFERENCES

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

[2] M. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1-2, pp. 1–142, 2013.

[3] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *IEEE/RSJ IROS*, 2006.

[4] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *IEEE ICRA*, 2004.

[5] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *NeurIPS*, 2014, pp. 1071–1079.

[6] B. Shahriari, K. Swersky, Z. Wang, R. 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.

[7] D. Lizotte, T. Wang, M. Bowling, and D. Schuurmans., "Automatic gait optimization with Gaussian process regression," in *IJCAI*, 2007.

[8] R. Calandra, A. Seyfarth, J. Peters, and M. Deisenroth, "Bayesian optimization for learning gaits under uncertainty," *AMAI*, vol. 1 1, pp. 1–19 1–19, 2015.

[9] A. Rai, R. Antonova, F. Meier, and C. G. Atkeson, "Using simulation to improve sample-efficiency of Bayesian optimization for bipedal robots." *JMLR*, vol. 20, no. 49, pp. 1–24, 2019.

[10] M. Tesch, J. Schneider, and H. Choset, "Adapting control policies for expensive systems to changing environments," in *IEEE/RSJ IROS*, 2011.

[11] S. R. Kuindersma, R. A. Grupen, and A. G. Barto, "Variable risk control via stochastic optimization," *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 806–825, 2013.

[12] R. Martinez-Cantin, N. de Freitas, E. Brochu, J. Castellanos, and A. Doucet, "A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot." *Autonomous Robots*, vol. 27, no. 3, pp. 93–103, 2009.

[13] R. Marchant and F. Ramos, "Bayesian optimisation for informative continuous path planning," in *IEEE ICRA*, 2014, pp. 6136–6143.

[14] J. Nogueira, R. Martinez-Cantin, A. Bernardino, and L. Jamone, "Unscented Bayesian optimization for safe robot grasping," in *IEEE/RSJ IROS*, 2016, pp. 1967–1972.

[15] J. Castanheira, P. Vicente, R. Martinez-Cantin, L. Jamone, and A. Bernardino, "Finding safe 3d robot grasps through efficient haptic exploration with unscented Bayesian optimization and collision penalty," in *IEEE/RSJ IROS*, 2018, pp. 1643–1648.

[16] C. Daniel, O. Kroemer, M. Viering, J. Metz, and J. Peters, "Active reward learning with a novel acquisition function," *Autonomous Robots*, vol. 39, no. 3, pp. 389–405, 2015.

[17] A. Cully, J. Clune, D. Tarapore, and J. B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, p. 503–507, 2015.

[18] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[19] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, "Safe exploration for optimization with Gaussian processes," in *ICML*, 2015, pp. 997–1005.

[20] R. Martinez-Cantin, K. Tee, and M. McCourt, "Practical Bayesian optimization in the presence of outliers," in *AISTATS*, 2018, pp. 1722–1731.

[21] R. Martinez-Cantin, "Bayesian optimization with adaptive kernels for robot control," in *ICRA*, 2017, pp. 3350–3356.

[22] J. Garcia-Barcos and R. Martinez-Cantin, "Fully distributed Bayesian optimization with stochastic policies," in *IJCAI*, 2019, pp. 2357–2363.

[23] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. A. Castellanos, "Active policy learning for robot planning and exploration under uncertainty," in *Robotics: Science and Systems*, 2007.

[24] D. Huang, T. T. Allen, W. I. Notz, and N. Zheng, "Global optimization of stochastic black-box systems via sequential kriging meta-models," *Journal of Global Optimization*, vol. 34, no. 3, pp. 441– 466, 2006.

[25] R. Martinez-Cantin, "Funneled Bayesian optimization for design, tuning and control of autonomous systems," *IEEE Trans Cybern*, vol. 49, no. 4, pp. 1489–1500, 2019.

[26] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[27] J. Snoek, H. Larochelle, and R. Adams, "Practical Bayesian optimization of machine learning algorithms," in *NeurIPS*, 2012, pp. 2960–2968.

[28] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of Bayesian methods for seeking the extremum," in *Towards Global Optimisation 2*. Elsevier, 1978, pp. 117–129.

[29] I. Bogunovic, J. Scarlett, S. Jegelka, and V. Cevher, "Adversarially robust optimization with Gaussian processes," in *NeurIPS*, 2018, pp. 5760–5770.

[30] R. Oliveira, L. Ott, and F. Ramos, "Bayesian optimisation under uncertain inputs," in *AISTATS*, 2019, pp. 1177–1184.

[31] S. Julier and J. Uhlmann, "The scaled unscented transformation," in *IEEE ACC*, Anchorage AK, USA, 8–10 May 2002, pp. 4555–4559.

[32] R. van der Merwe, "Sigma-point Kalman filters for probabilistic inference in dynamic state-space models," Ph.D. dissertation, OGI School of Science & Eng., Oregon Health & Science University, 2004.

[33] M. Toussaint, "The Bayesian search game," in *Theory and Principled Methods for Designing Metaheuristics*. Springer, 2014.

[34] A. D. Bull, "Convergence rates of efficient global optimization algorithms," *JMLR*, vol. 12, pp. 2879–2904, 2011.

[35] D. Ginsbourger, R. Le Riche, and L. Carraro, "Kriging is well-suited to parallelize optimization," in *Computational intelligence in expensive optimization problems*. Springer, 2010, pp. 131–162.

[36] T. Desautels, A. Krause, and J. Burdick, "Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization," *JMLR*, vol. 15, no. 1, pp. 3873–3923, 2014.

[37] E. Contal, D. Buffoni, A. Robicquet, and N. Vayatis, "Parallel Gaussian process optimization with upper confidence bound and pure exploration," in *ECMLKDD*. Springer, 2013, pp. 225–240.

[38] A. Shah and Z. Ghahramani, "Parallel predictive entropy search for batch global optimization of expensive objective functions," in *NeurIPS*, 2015.

[39] J. González, Z. Dai, P. Hennig, and N. Lawrence, "Batch Bayesian optimization via local penalization," in *AISTATS*, 2016, pp. 648–657.

[40] J. Assel, Z. Wang, B. Shahriari, and N. Freitas, "Heteroscedastic treed Bayesian optimizaion," *arXiv*, vol. abs/1410.7172v2, 2015.

[41] Z. Wang and S. Jegelka, "Max-value entropy search for efficient Bayesian optimization," in *ICML*, vol. 70, 2017, pp. 3627–3635.

[42] Z. Wang, C. Gehring, P. Kohli, and S. Jegelka, "Batched large-scale Bayesian optimization in high-dimensional spaces," in *AISTATS*, 2018.