Robot Learning with Crash Constraints

Alonso Marco[1], Dominik Baumann[4,1], Majid Khadiv[1], Philipp Hennig[2,1], Ludovic Righetti[3,1] and Sebastian Trimpe[4,1]

***Abstract*— In the past decade, numerous machine learning algorithms have been shown to successfully learn optimal policies to control real robotic systems. However, it is common to encounter failing behaviors as the learning loop progresses. Specifically, in robot applications where failing is undesired but not catastrophic, many algorithms struggle with leveraging data obtained from failures. This is usually caused by (i) the failed experiment ending prematurely, or (ii) the acquired data being scarce or corrupted. Both complicate the design of proper reward functions to penalize failures. In this paper, we propose a framework that addresses those issues. We consider failing behaviors as those that violate a constraint and address the problem of *learning with crash constraints*, where no data is obtained upon constraint violation. The no-data case is addressed by a novel GP model (GPCR) for the constraint that combines discrete events (failure/success) with continuous observations (only obtained upon success). We demonstrate the effectiveness of our framework on simulated benchmarks and on a real jumping quadruped, where the constraint threshold is unknown a priori. Experimental data is collected, by means of constrained Bayesian optimization, directly on the real robot. Our results outperform manual tuning and GPCR proves useful on estimating the constraint threshold.**

I. INTRODUCTION

During the past decades, developments in machine learning (ML) have boosted the usage of algorithms that learn directly from data in real systems. In the context of robotics, data-efficient reinforcement learning methods have allowed to increase automation and mitigate human intervention during the learning process. An increasingly popular family of algorithms that has made this possible is Bayesian optimization (BO) [1], which has been proposed to iteratively search for optimal controller parameters from a few evaluations. To compensate for data scarcity, prior assumptions are placed on the unknown performance objective using a Gaussian process (GP) [2] model. This is iteratively optimized by collecting informative data points through experiments.

While BO has shown impressive results in learning from data on real robots, such as bipedal locomotion [3]–[6], quadrotor hovering [7], and manipulation [8]–[10], not all issues have been solved yet. For example, during the search, some controller parametrizations may yield unstable behavior (i.e., one of the robot states growing unboundedly).

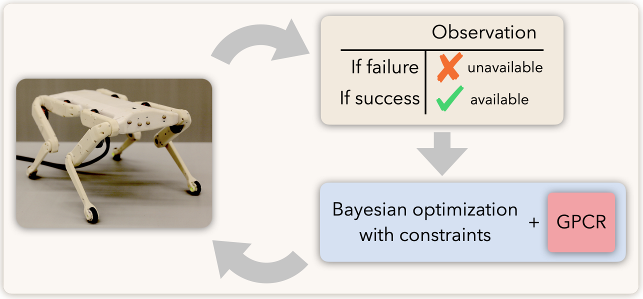

Fig. 1: Learning loop with a quadruped that learns to jump using Bayesian optimization with constraints and GPCR. While we acquire height and current measurements for successful jumps, failed jumps report no observations.

This can lead to the robot failing to accomplish the task, for instance, by falling down [3]–[5], [11], or by having to emergency-stop it prematurely [7], [8]. While failures must be completely avoided in safety-critical applications [7], herein we consider scenarios where failing is undesired but not catastrophic. In such scenarios, failures still provide a valuable source of information, yet, designing effective reward functions to account for failures is difficult. Indeed, failures are often handled by using heuristic user-defined fixed penalties to the reward. Such practice usually requires expert (or domain) knowledge [3], [4], [6], [8]–[11]. For example, quantifying the reward of a quadruped that falls down while learning to jump implies analyzing the state trajectories and calibrating a substantially different reward from that of successful jumps. This requires detailed knowledge about the platform and the task itself.

Because expert knowledge might not always be available (or impractical to expect it), we propose herein a reward function that is *not defined* at failing parametrizations, i.e., those in which the robot fails to complete the task. In this paper, we refer to this as the problem of *learning with crash constraints* (LCC): we obtain a reward value in case of successful execution, but in case of a failure we only know that the robot failed. We address it using Bayesian optimization with unknown constraints (BOC).

In a nutshell, BOC extends BO by adding black-box constraint functions that are expensive-to-evaluate, modeled with GPs. Such framework avoids visiting areas where the constraints are violated, as these are revealed during the search, hence speeding up convergence to the constrained optimum [12]–[15].

Because in BOC it is assumed that both, the constraint

This work was supported by the International Max Planck Research School for Intelligent Systems (IMPRS-IS).

[1]Max Planck Institute for Intelligent Systems, Tübingen/Stuttgart, Germany; {amarco,dbaumann,mkhadiv}@tuebingen.mpg.de
[2]University of Tübingen, Computer Science Department, 72074 Tübingen, Germany; philipp.hennig@uni-tuebingen.de
[3]New York University, Brooklyn, NY 11201 USA; ludovic.righetti@nyu.edu [4]RWTH Aachen, 52062 Aachen, Germany; trimpe@dsme.rwth-aachen.de

and the reward function are defined upon failure, it cannot be used straightforwardly to tackle the LCC problem. Depending on the nature of the constraint, BOC assumes that it can report either a real number [14]–[16] or a binary outcome [13], [17], [18] when evaluated. In the former case, the constraint is usually modeled with standard GP regression, while in the latter it is modeled with a GP classifier [2, Sec. 3.2]. However, in the LCC problem, none of those two modeling options are adequate. On one hand, a GP regressor is impractical because no real values are reported upon failure. On the other hand, a GP classifier (GPC) expects binary observations while, upon success, the constraint observations are real valued. In such case, modeling the constraint using GPC implies discarding valuable information about "how well was the constraint satisfied" that could otherwise be leveraged throughout the optimization process to avoid visiting unpromising regions.

Herein, we propose a novel GP model for the constraint that integrates *hybrid* observations, i.e., (i) a discrete label indicating whether the task was accomplished or not and (ii) a continuously valued observation only in case the task was accomplished. Due to the conceptual closeness of this model with a GP classifier, but reinterpreting the observations in the GP regressor, we call it *Gaussian process for classified regression* (GPCR). As opposed to existing multi-output Gaussian process models, GPCR combines discrete labels and continuous observations in a single-output GP model.

The GPCR model is tailored to the LCC problem and exhibits two main benefits with respect to existing approaches for robot learning with BO. First, it removes the need of penalizing failure cases with user-defined penalties to the reward. Instead, those failure cases are captured in the GPCR that models the constraint. Second, GPCR treats the constraint threshold as a hyperparameter and estimates it from data. This brings two advantages along: (i) mitigating the need of expert knowledge to define the constraint threshold, which is usually a challenging problem [7], and (ii) reducing possible human bias introduced by ad hoc choices.

Contributions: This paper proposes three main contributions. First, a novel BOC framework that specifically addresses the LCC problem by (i) transferring the modeling effort of robot failures to the constraint and (ii) mitigating failures during learning. Within this framework, any available BOC acquisition function can be used. However, because the popular expected improvement with constraints (EIC) [14] is employed herein, we name our framework *expected improvement with crash constraints* (EIC^2). In general, EIC^2 becomes useful when the "reason for failure" can be measured upon success, but estimating the constraint threshold requires large amounts of expert knowledge. Second, we propose a novel single-output GP model (GPCR) for the constraint that handles hybrid data (discrete labels and real values) and treats the constraint threshold as a hyperparameter, hence reducing the need for expert knowledge. Third, we propose for the first time a solution to the LCC problem when learning from data on a real system, specifically, on a real jumping quadruped robot (cf. Fig. 1). Our approach enabled the quadruped to learn significantly higher jumps than those found with manual tuning.

Related work: In contexts other than robotics, the LCC problem first appeared in [19], and has adopted different names ever since [18], [20]. In the field of global optimization, a number of recent BOC algorithms have been proposed to address the LCC problem on simulated scenarios [17], [21], and numerical benchmarks [20].

In the area of robot learning with BO, the LCC problem is present, but it has not been treated so far with well founded methodologies. Instead, it is usually overcome with ad hoc heuristic strategies that range in diversity depending on the task at hand. A common heuristic is to discard any acquired data upon failure and assign a user-defined pre-fixed penalty to the reward. This approach was followed in [6] to prevent the torso of the robot from rotating above a certain angle, and in [8] to prevent some robot states from leaving a predefined safety zone. Another heuristic is to use whatever data becomes available before failing. For example, in [3], [4], [11], the reward of a walking robot is the distance walked before falling, which acts as a penalty for the fall.

As in this work, EI has previously been extended to address the LCC problem, where the constraint is modeled using different forms of a GP classifier, such as logistic regression [17] and probit regression [13]. Also in the context of robotics, [9], [10] propose a different BOC algorithm that specifically addresses the LCC problem. Therein, the constraint is modeled as a GP classifier, and its observations are assumed to be binary. Such references differ from this work in two main aspects: (i) contrary to GPCR, the GP that they use to model the constraint does not explicitly account for hybrid data (discrete labels and continuous values), (ii) the constraint threshold is inherently assumed zero or fixed, while we treat it as a learnable hyperparameter, and (iii) their method expands only locally the initial safe region.

In [22]–[24], several multi-output Gaussian process models are proposed to tackle continuous and discrete observations. They jointly model the objective as a GP regressor and a set of binary constraints as GP classifiers. Albeit the LCC problem can also be addressed with such models, (i) they are restricted to cases in which observations are given as binary, (ii) they cannot learn the constraint threshold from data, and (iii) the applicability of such models to robot settings is unclear.

II. PRELIMINARIES

A. Problem setting: robot learning with crash constraints

Let a dynamical system be controlled with a parametrized policy to execute a specific task. A real experiment successfully executed on the system with policy parameters $x \in \mathcal{X}_\mathrm{S} \subseteq \mathcal{X} \subset \mathbb{R}^D$ yields a performance $f(x)$, where the mapping $f : \mathcal{X}_\mathrm{S} \to \mathbb{R}$ is unknown. The goal is to solve the constrained optimization problem

$$x_* = \underset{x \in \mathcal{X}_\mathrm{S}}{\operatorname{argmin}} f(x), \qquad (1)$$

where $\mathcal{X}_\mathrm{S}$ is a safe region, outside of which the controller parameters make the system *crash* during task execution.

Such safe region is determined as $\mathcal{X}_\text{S} = \{x : g_1(x) \leq c_1, \dots, g_G(x) \leq c_G\}$, where $g_j : \mathcal{X}_\text{S} \to \mathbb{R}$, $\forall j = 1, \dots, G$ are G constraints that indicate whether the aforementioned task succeeds or not and c_j is the constraint threshold. We assume coupled evaluations of f and g_j, i.e., all queries $f(x), g_1(x), \dots, g_G(x)$ are obtained simultaneously by running a robot experiment with controller parametrization x. Furthermore, neither the objective f nor the constraints g_j are defined outside the safe region $\mathcal{X}_\text{S}$. Finally, the constraint thresholds c_j are unknown.

B. Gaussian process (GP)

We model the objective as a Gaussian process, $f \sim \mathcal{GP}(0, k(x, x'))$, with covariance function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and zero prior mean. Observations $y(x) = f(x) + \varepsilon$ are modeled using additive Gaussian noise $\varepsilon \sim \mathcal{N}(\varepsilon; 0, \sigma_\text{no}^2)$. After having collected n observations from the objective $\mathcal{D}_n = \{\boldsymbol{x}_n, \boldsymbol{y}_n\} = \{x_1, \dots, x_n, y_1, \dots, y_n\}$, its predictive distribution at location x is given by $p(f|\mathcal{D}_n, x) = \mathcal{N}(f(x); \mu(x|\mathcal{D}_n), \sigma^2(x|\mathcal{D}_n))$, with predictive mean $\mu(x|\mathcal{D}_n) = \boldsymbol{k}_n^\top(x)[K_n + \sigma_\text{no}^2 I]^{-1}\boldsymbol{y}_n$, where the entries of the vector $\boldsymbol{k}_n(x)$ are $[\boldsymbol{k}_n(x)]_i = k(x_i, x)$, the entries of the Gram matrix K_n are $[K_n]_{i,j} = k(x_i, x_j)$, and the entries of the vector of observations $\boldsymbol{y}_n$ are $[\boldsymbol{y}_n]_i = y_i$. The predictive variance is given by $\sigma^2(x|\mathcal{D}_n) = k(x, x) - \boldsymbol{k}_n^\top(x)[K_n + \sigma_\text{no}^2 I]^{-1}\boldsymbol{k}_n(x)$. In the remainder of the paper, we drop the dependency on the current data set $\mathcal{D}_n$ and write $\mu(x)$, $\sigma(x)$ to refer to $\mu(x|\mathcal{D}_n)$, $\sigma(x|\mathcal{D}_n)$, respectively.

C. Bayesian optimization with and without constraints

We address the constrained optimization problem (1) using BOC, where both the objective f and each constraint g_i are modeled with a GP. The GP posterior is used by BOC algorithms to steer the search toward regions inside $\mathcal{X}$ where the constraints are satisfied with high probability. To this end, the next candidate point x_{n+1} is iteratively computed as maximizer of an acquisition function $\alpha : \mathcal{X} \to \mathbb{R}$, i.e., $x_{n+1} = \arg\max_{x \in \mathcal{X}} \alpha(x)$. The acquisition α implicitly depends on the GP models of the objective f and the constraints g_i conditioned on the acquired data points.

In this work we use *expected improvement with constraints* (EIC) [14]. EIC extends the well-known *expected improvement* (EI) [25], which is a BO strategy for unconstrained problems, to the constrained case. In the following, we describe both methods.

1) Expected improvement (EI): This algorithm reveals areas of low cost function values that are expected to improve upon the best observation so far. The improvement is defined as $\max\{\eta - f(x), 0\}$ and its expectation is given by

$$\alpha_\text{EI}(x) = \mathbb{E}_{f(x) \sim p(f|\mathcal{D}_n, x)}\left[\max\{\eta - f(x), 0\}\right],$$

where $\eta = \min_{r \in \{1, \dots, n\}} y_r$ is the best observation so far.

2) Expected improvement with constraints (EIC): Akin to EI, this method defines the improvement as $\max\{\eta_\text{cons} - f(x), 0\}$, where $\eta_\text{cons} = \min_{r \in \{1, \dots, n\}} y_r$, s.t. $g_j(x_r) \leq 0 \ \forall j$ is the best constrained observation so far. As EI, it reveals areas where such improvement is high in expectation.

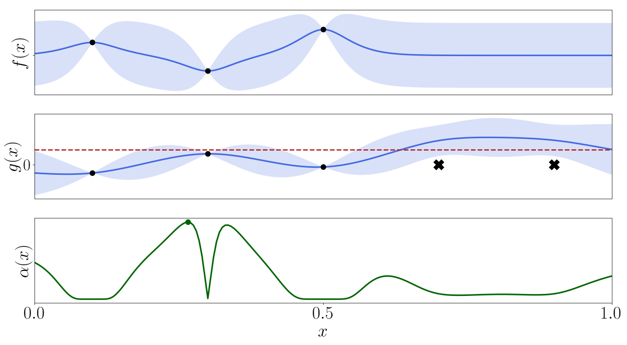

Fig. 2: Top: Standard GP that models the unknown objective f, conditioned on three successful observations. Predictive mean (solid line) and ± 2 standard deviation (surface path). Middle: GPCR model of the unknown constraint g, conditioned on five observations: three successful (circles) and two failures (crosses, both plotted at zero for convenience). The GPCR model pushes the probability mass above the learned constraint threshold $\hat{c}$ (dashed line) near the two failure observations. Bottom: EIC acquisition function and next candidate point (green dot) that maximizes it.

In addition, it requires those areas to be safe with high probability. Its acquisition function is given by

$$\alpha_\text{EIC}(x) = \mathbb{E}_{f(x) \sim p(f|\mathcal{D}_n, x)}\left[\max\{\eta_\text{cons} - f(x), 0\}\right]\Gamma(x), \quad (2)$$

where $\Gamma(x) = \prod_{j=1}^{G} \Pr(g_j(x) \leq 0)$ and $\Pr(g_j(x) \leq 0)$ is the probability of satisfying constraint j. Although (2) requires knowing η_cons, such value can only be computed when at least one safe data point has been found. In the absence of safe data points (e.g., at early stages of the exploration), EIC ignores the objective f and explores using the information of the constraint only, i.e., $\alpha_\text{EIC}(x) = \Gamma(x)$.

III. BAYESIAN OPTIMIZATION WITH CRASH CONSTRAINTS

In this section, we introduce the EIC^2 framework, which differs from standard BOC in two aspects, described next.

The first difference is about the way each constraint g_j is modeled. While BOC typically uses either a GP regressor or a GP classifier, EIC^2 tackles the absence of data by using a novel GP model: GPCR. Although we explain in detail GPCR in Sec. IV, herein we provide a high-level intuition. GPCR is updated with a discrete label indicating "failure" when the constraints are violated and with a real value when the constraint is satisfied. The GPCR model is illustrated on a one-dimensional toy example in Fig. 2 (middle). Therein, the two failures (crosses) are given as discrete labels; they push upwards the probability mass of the GP model above the constraint threshold, to discourage such region from being explored. Furthermore, the same model behaves approximately as a standard GP around the three successful observations (black dots). The objective cost f is modeled with a standard GP, $f \sim \mathcal{GP}(0, k(x, x'))$. Because no cost observation is obtained upon failure, the GP model

Algorithm 1 Expected improvement with crash constraints

```
1:  Input: Hyperpriors $p(\theta)$, $p(c)$, covariance function $k$, $M > 1$,
    $\tau$, $\delta$, $f \sim \mathcal{GP}(0, k(x, x'))$, $g \sim \mathcal{GPCR}(0, k(x, x'))$
2:  Initialization Choose $x_1 \in \mathcal{X}$ randomly
3:  for $n = 1$ to $M - 1$ do
4:      $\{y_n^f, y_n^g, l_n\} \leftarrow$ RobotExperiment($x_n$)
5:      $\mathcal{D}_n^f \leftarrow \mathcal{D}_n^f \cup \{y_n^f, x_n\}$
6:      $\mathcal{D}_n^g \leftarrow \mathcal{D}_n^g \cup \{y_n^g, l_n, x_n\}$
7:      Optimize hyperparameters $\hat{\theta}_n^f$ using MAP                ▷ [2]
8:      Optimize hyperparameters $(\hat{\theta}_n^g, \hat{c}_n^g)$ using MAP      ▷ (10)
9:      $x_{n+1} = \arg\max_{x \in \mathcal{X}} \alpha(x)$          ▷ Next candidate via (2)
10: Estimate constrained global minimum $x_{\text{best}}$                ▷ (3)
11: return $x_{\text{best}}$

12: function RobotExperiment($x_n$)
13:     Run robot experiment for $\tau$ time steps.
14:     if Task failure then
15:         return $\{\varnothing, \varnothing, l = 0\}$
16:     else
17:         return $\{y^f, y^g, l = +1\}$
```

in f is not updated with such failures, as shown in Fig. 2 (top). The same approach has been recently adopted in [9], in the context of robot learning.

The second difference is that in EIC^2, the constraint threshold c_j (cf. Sec. II-A) is assumed unknown and estimated from data, while in BOC, c_j is user-defined. In Fig. 2 (middle), GPCR estimates the constraint threshold $\hat{c}$ (dashed line) right above the successful observations.

The EIC^2 framework uses the acquisition function (2), which is illustrated in Fig. 2 (bottom). As can be seen, the areas in the right, where the constraint is violated, are discouraged. After M iterations, EIC^2 reports the expected objective cost that satisfies the constraints with high probability [14], computed as

$$x_{\text{best}} = \underset{x \in \mathcal{X}}{\operatorname{argmin}}\, \mu(x) \quad \text{s.t. } \Gamma(x) \geq 1 - \delta, \tag{3}$$

where $\delta \in [0, 1]$ is typically set to a small number and Γ is given in (2). Algorithm 1 summarizes EIC^2 in pseudocode with a single constraint for simplicity, where y_t^f and y_t^g constitute noisy observations of the objective f and the constraint g, respectively.

In the following, we explain the GPCR model that EIC^2 uses to model the constraints and also how this model can be used to estimate the constraint threshold while learning. For ease of presentation, we consider therein a single constraint g.

IV. BAYESIAN MODEL FOR HYBRID DATA AND UNKNOWN CONSTRAINT THRESHOLD

The considered problem setting assumes that neither the objective f nor the constraint g can be observed upon failure (cf. Sec. II-A). In such case, the only acquired data is a binary label indicating "failure." In contrast, upon constraint satisfaction we obtain real observations of f and g *and* a binary label indicating "success." In the following, we propose a GP model for g able to handle *hybrid* data: binary labels and continuous values.

A. *Gaussian process for classified regression (*GPCR*)*

Let us assume that for each controller parametrization x_i we simultaneously obtain two types of observations: (i) a binary label $l_i \in \{0, +1\}$ that determines whether the experiment was failure or success, respectively, and (ii) a noisy constraint value $y_i \in \mathbb{R}$ only when the experiment is successful:

$$(y_i, l_i) = \begin{cases} (g_i + \varepsilon, \quad +1), & \text{if } x_i \text{ is success} \\ (\varnothing, \quad\quad\; 0), & \text{if } x_i \text{ is failure} \end{cases}, \tag{4}$$

where we have used the shorthand notation $g_i = g(x_i)$, $y_i = y(x_i)$, and $l_i = l(x_i)$, and $\varepsilon \sim \mathcal{N}(\varepsilon; 0, \sigma_{\text{no}}^2)$. The observation model can be represented with a probabilistic likelihood

$$p(y_i, l_i | g_i) = \left[H(c - g_i)\mathcal{N}(y_i; g_i, \sigma_{\text{no}}^2)\right]^{l_i} \left[H(g_i - c)\right]^{1 - l_i}, \tag{5}$$

where $H(z)$ is the Heaviside function, i.e., $H(z) = 1$, if $z \geq 0$, and 0 otherwise. The likelihood (5) captures our knowledge about the latent constraint[1]: if x_i is a failure ($l_i = 0$), all we know about g_i is that it takes any possible value above the threshold c, with all values $g_i \geq c$ being equally likely, but we never specify what that value is. To this end, the likelihood $p(y_i, l_i = 0 | g_i) = H(g_i - c)$ places all probability mass above the threshold c. On the contrary, if x_i is successful ($l_i = 1$), all the probability mass falls below c for consistency, shaped as truncated Gaussian noise centered at g_i, i.e., $p(y_i, l_i = 1 | g_i) = H(c - g_i)\mathcal{N}(y_i; g_i, \sigma_{\text{no}}^2)$.

Let the latent constraint be observed at locations $X = \{X_{\text{s}}, X_{\text{u}}\}$, which entails both, successful $X_{\text{s}} = \{x_i\}_{i=1}^{N_{\text{s}}}$ and failing controllers $X_{\text{u}} = \{x_i\}_{i=N_{\text{s}}+1}^{N}$. The corresponding latent constraint values at X_{s} and X_{u} are $\boldsymbol{g}_{\text{s}} = [g_1, g_2, \ldots, g_{N_{\text{s}}}]^\top$, and $\boldsymbol{g}_{\text{u}} = [g_{N_{\text{s}}+1}, g_{N_{\text{s}}+2}, \ldots, g_N]^\top$, respectively. The latent constraint values $\boldsymbol{g} = (\boldsymbol{g}_{\text{s}}, \boldsymbol{g}_{\text{u}})$ induce observations grouped in the hybrid set of observations $\mathcal{D} = \{y_i, l_i\}_{i=1}^{N_{\text{s}}} \cup \{l_i\}_{i=N_{\text{s}}+1}^{N}$, which contains both, discrete labels and real scalar values. We assume such observations to be independent and identically distributed, which allows the likelihood to factorize $p(\mathcal{D}|\boldsymbol{g}) = \prod_{i=1}^{N} p(y_i, l_i | g_i)$. Then, the likelihood over the data set $\mathcal{D}$ becomes

$$p(\mathcal{D}|\boldsymbol{g}) = \prod_{i=1}^{N_{\text{s}}} H(c - g_i)\mathcal{N}(y_i | g_i, \sigma_{\text{no}}^2) \prod_{i=N_{\text{s}}+1}^{N} H(g_i - c). \tag{6}$$

The posterior follows from Bayes theorem $p(\boldsymbol{g}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{g})p(\boldsymbol{g})$, with zero-mean[2] Gaussian prior $p(\boldsymbol{g}) = \mathcal{N}(\boldsymbol{g}|\boldsymbol{0}, \boldsymbol{K})$, $[\boldsymbol{K}]_{i,j} = k(x_i, x_j)$, and multivariate likelihood (6). The terms in (6) can be permuted and the Gaussian factors can be grouped in a multivariate Gaussian $\prod_{i=1}^{N_{\text{s}}} \mathcal{N}(y_i | g_i, \sigma_{\text{no}}^2) = \mathcal{N}(\boldsymbol{y}_{\text{s}} | \boldsymbol{g}_{\text{s}}, \sigma_{\text{no}}^2 \boldsymbol{I})$. The product of two multivariate Gaussians of different dimensionality $\mathcal{N}(\boldsymbol{y}_{\text{s}} | \boldsymbol{g}_{\text{s}}, \sigma_{\text{no}}^2 \boldsymbol{I})\mathcal{N}(\boldsymbol{g}|\boldsymbol{0}, \boldsymbol{K})$ equals another unnormalized Gaussian $\kappa\mathcal{N}(\boldsymbol{g}; \tilde{\boldsymbol{m}}, \boldsymbol{\Sigma})$, whose mean, variance and scaling factor depend on the observations and the noise, i.e., $\tilde{\boldsymbol{m}} =$

[1]The likelihood function (5) is an unnormalized density due to the presence of the Heaviside functions. Using unnormalized likelihood functions is common in the context of GP classification models [18], [26].

[2]A non-zero constant mean function could be included in the model with no additional complexity.

$\tilde{\boldsymbol{m}}(\boldsymbol{y}_{\text{s}}, \sigma_{\text{no}}^2)$, $\tilde{\boldsymbol{\Sigma}} = \tilde{\boldsymbol{\Sigma}}(\sigma_{\text{no}}^2)$, and $\kappa = \kappa(\boldsymbol{y}_{\text{s}}, \sigma_{\text{no}}^2)$. Hence, the posterior becomes

$$p(\boldsymbol{g}|\mathcal{D}) \propto \mathcal{N}(\boldsymbol{g}; \tilde{\boldsymbol{m}}, \tilde{\Sigma}) \prod_{i=1}^{N_{\text{s}}} H(c - g_i) \prod_{i=N_{\text{s}}+1}^{N} H(g_i - c). \quad (7)$$

The Heaviside functions in (7) restrict the support of $p(\boldsymbol{g}|\mathcal{D})$ to an unbounded hyperrectangle with a single corner at location $g_i = c$, $\forall i \in \{1, \ldots, N\}$. Because this particular shape complicates the computation of the predictive distribution, we approximate the posterior $p(\boldsymbol{g}|\mathcal{D})$. Among the many possible methods [27], we opted here for expectation propagation (EP), a variational approach that approximates well the right hand side of (7) with a multivariate Gaussian $q(\boldsymbol{g}) = Z_{\text{EP}}\mathcal{N}(\boldsymbol{g}; \boldsymbol{\mu}_{\text{EP}}, \boldsymbol{\Sigma}_{\text{EP}})$ [28]. In EP, the moments $\boldsymbol{\mu}_{\text{EP}}$, $\boldsymbol{\Sigma}_{\text{EP}}$ and $Z_{\text{EP}} \in \mathbb{R}_{>0}$ are matched to those of the right hand side of (7). The zero-th order moment Z_{EP} is a constant that approximates the model evidence $p(\mathcal{D}) = \int_{\boldsymbol{g}} p(\mathcal{D}|\boldsymbol{g})p(\boldsymbol{g})\mathrm{d}\boldsymbol{g} \simeq \int_{\boldsymbol{g}} q(\boldsymbol{g})\mathrm{d}\boldsymbol{g} = Z_{\text{EP}}$, which will be further discussed in Sec. IV-B.

Since the approximate posterior $q(\boldsymbol{g}) \simeq p(\boldsymbol{g}|\mathcal{D})$ is Gaussian, the predictive distribution at some unobserved location x is a univariate Gaussian $p(g(x)) = \mathcal{N}(g(x); \mu(x), \sigma^2(x))$, whose moments are given analytically [2, Sec. 3.4.2]

$$\begin{aligned} \mu(x) = &\quad \boldsymbol{k}^\top(x) K^{-1} \boldsymbol{\mu}_{\text{EP}} \\ \sigma^2(x) = &\quad k(x,x) - \boldsymbol{k}^\top(x) K^{-1} \left(\boldsymbol{I} - \boldsymbol{\Sigma}_{\text{EP}} K^{-1}\right) \boldsymbol{k}(x). \end{aligned} \quad (8)$$

The above equations are also valid for a set of unobserved locations $\boldsymbol{x} = [x_1, \ldots, x_T]$, in which case $p(g(\boldsymbol{x})) = \mathcal{N}(g(\boldsymbol{x}); \mu(\boldsymbol{x}), \Sigma(\boldsymbol{x}))$ is the sought GPCR model, indexed at $\boldsymbol{x}$. Generally, we say that $g \sim \mathcal{GPCR}(0, k(x, x'))$ is modeled with GPCR, with zero-mean and kernel k.

The predictive probability of constraint satisfaction at location x is given by

$$\Pr(l(x) = +1) = \int_{g(x)} H(c - g(x)) p(g(x)) \mathrm{d}g(x),$$

where the dependency on x has been introduced for clarity. Since $p(g(x))$ is a univariate Gaussian, such integral can be resolved analytically

$$\Pr(l(x) = +1) = \Phi\left(\tfrac{c - \mu(x)}{\sigma(x)}\right), \quad (9)$$

where $\mu(x)$ and $\sigma(x)$ are given in (8), and $\Phi(\cdot)$ is the cumulative density function of a standard normal distribution.

Both, the likelihood model (6) and the approximate posterior (7), depend on the threshold parameter c, which can be seen as a discriminator that distinguishes instability from stability in the axis of the constraint value. While c has been treated so far as a fixed value, we consider it now a hyperparameter. Next, we explain how it can be estimated.

B. Constraint threshold as a hyperparameter

We treat fully Bayesian the model hyperparameters θ (e.g., lengthscales of the kernel) and the constraint threshold c (cf. Sec. IV-A) and seek to compute their posterior probability distribution $p(c, \theta|\mathcal{D}) \propto p(c)p(\theta)p(\mathcal{D}|c, \theta)$. While $p(c)$ and $p(\theta)$ are given, the model evidence $p(\mathcal{D}|c, \theta) = \int_{\boldsymbol{g}} p(\mathcal{D}|\boldsymbol{g}, c, \theta)p(\boldsymbol{g})\mathrm{d}\boldsymbol{g}$ involves computing an intractable integral. Such integral can be approximated as $\int_{\boldsymbol{g}} q(\boldsymbol{g}; c, \theta)\mathrm{d}\boldsymbol{g} = Z_{\text{EP}}(c, \theta)$ [28] (cf. Sec. IV-A), where the dependency on c and θ has been introduced for clarity. This enables many different approximations for $p(c, \theta|\mathcal{D})$. Herein, we directly estimate c and θ via *maximum a posteriori* (MAP) [2], given by

$$\hat{c}, \hat{\theta} = \underset{c > y_{\max},\, \theta}{\operatorname{argmax}} \log Z_{\text{EP}}(c, \theta) + \log p(c) + \log p(\theta), \quad (10)$$

where the prior over the constraint threshold is defined as $p(c) = \text{Gamma}(c - y_{\max})$. We shift the origin of the distribution to restrict the support above the worst constraint value among the successful observations, i.e., $c > y_{\max}$, with $y_{\max} = \max\{y_i\}_{i=1}^{N_{\text{s}}}$.

C. Constraint modeled with GPCR *in* EIC^2

The proposed EIC^2 framework suggests candidate evaluations according to (2), where the constraints are modeled with GPCR. With $G \geq 1$ constraints, the probability of the constraints being satisfied is given as $\Gamma(x) = \prod_{j=1}^{G} \Pr(l_j(x) = +1)$, where $\Pr(l_j(x) = +1)$ is computed as in (9). For simplicity, we consider in the following section $G = 1$. However, EIC^2 is generally applicable when $G \geq 1$, as its acquisition function (2) (EIC) has been shown to work with multiple constraints [16] and it only requires the constraint to be modeled with a GP.

V. RESULTS

In this section, we assess the performance of EIC^2 in two constrained optimization scenarios where no data is obtained upon constraint violation: numerical benchmarks and a real robot platform. The overall goal is to show that expert knowledge can be reduced for solving the problem of crash constraints by (i) comparing the performance of EIC^2 against existing heuristic strategies that penalize failed experiments with user-defined costs and (ii) showing that the constraint threshold can be learned from real experimental data. In the following, we describe the experimental settings common to both scenarios and then describe the experiments and results of each scenario.

A. Overall experimental setting

We quantify the learning performance using *simple regret*, which is a common performance metric in BO [16], [29] and is defined as $r_n = y(x_n) - \min_{x \in \mathcal{X}} f(x)$. The input domain is normalized to the unit hypercube $\mathcal{X} = [0, 1]^D$.

For all GPs we assume a zero-mean prior and use Matérn kernel $5/2$ [2, Sec. 4.2] with scaling factor $\kappa > 0$ and lengthscale $\lambda > 0$. A prior probability distribution is assumed over the lengthscales and variance of the kernel, and over the constraint threshold c for the GPCR model. The observation noise for both f and g is modeled as in Sec. II-B and (4), respectively, with fixed noise variance σ_{no}^2. The hyperparameters $\theta = \{\lambda, \kappa\}$ are re-estimated at each iteration using maximum a posteriori (MAP), which in the case of the GPCR model is computed as in (10). Further

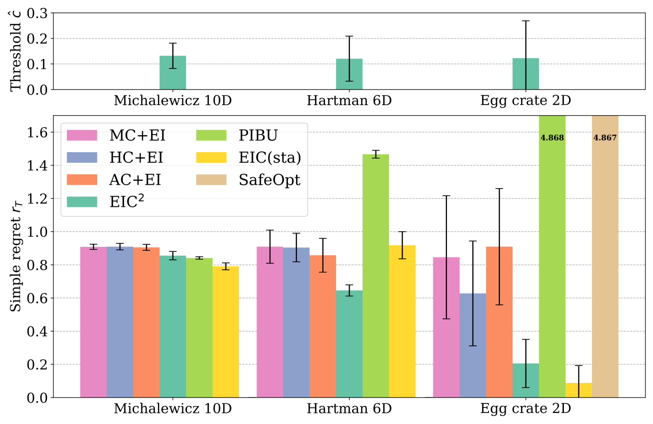

Fig. 3: Top: Estimated averaged constraint threshold $\hat{c}$ computed via (10) within the GPCR model used in EIC^2. The black error bars represent one standard deviation. Bottom: For each benchmark, the simple regret at the last iteration is reported for each heuristics and EIC^2. The black error bars represent half a standard deviation.

details about the hyperparameters can be found in our Python implementation[3] of EIC^2 and GPCR.

B. *Comparing against BO with heuristic penalties and BOC*

In related work on learning controller parameters with BO [3], [4], [6], [8], [11], robot failures are not interpreted as constraint violation. Instead, such works solve an unconstrained optimization problem using BO and simply assign a high cost to failure cases, to emphasize the poor performance of such experiments. However, such high cost is heuristically chosen and typically requires expert knowledge. Herein, we compare, in numerical benchmarks, EIC^2 against three plausible heuristic strategies to penalize failures: using a high cost (HC) [4], [6], [8], [11], using a middle cost (MC), and using an adaptive cost (AC). In the first, the penalizing cost is fixed before starting the learning experiments and is chosen as an upper bound on the function to optimize[4]. Similarly, in the second, the penalizing cost is chosen as the value of the initial evaluation. Finally, in the third, the penalizing cost of all failed experiments is re-adapted at each iteration to be the maximum (worst) observed cost value so far. As in [3], [4], [11], the unconstrained BO algorithm used for each strategies is EI (cf. Sec. II-C.1). The purpose of these comparisons is to emphasize that with EIC^2 no expert knowledge is required to penalize failures, and that the constraint threshold can be learned from data.

We also compare EIC^2 against three BOC methods: the well-known EIC (cf. Sec. II-C.2) as a baseline, and two popular methods with applications in robotics, PIBU[5] [9], [10] and SafeOpt [7], whose implementations are available online. Contrary to EIC^2, EIC and SafeOpt model the constraint using a standard GP and need to be informed about the value of the constraint threshold c. PIBU assumes binary constraint observations, hence, it models the constraint using a GP classifier.

1) Experimental choices: We use three challenging benchmarks for global minimization that exhibit multiple local minima: Michalewicz 10-dimensional function, Hartman 6-dimensional, and Egg Crate two-dimensional function [30], which are common benchmarks in BO, also used in [29].

The aforementioned cost objectives are constrained to a scalar function $g(x) = \prod_{d=1}^{D} \sin(2\pi x_d)$, which divides the volume in 2^D sub-hypercubes, among which there are 2^{D-1} safe sub-hypercubes, alternated with the unsafe ones. Constraint satisfaction is determined by $g(x) \leq c$, with $c = 0$. To compute the regret r_n we use the unconstrained global minimum value of each function, reported in [30].

We run Algorithm 1 and the other methods for $M = 100$ iterations and assess consistency by repeating it 100 times. In all cases, the initial point is randomly sampled within the safe regions for simplicity.

2) Results: In Fig. 3 (bottom), we see that EIC^2 consistently achieves lower average regret than the aforementioned heuristics, which confirms that (i) treating failures apart by modeling them using a constraint is more beneficial than heuristically penalizing failures and (ii) not knowing the constraint threshold a priori does not affect significantly the learning performance of EIC^2. Moreover, EIC^2 outperforms the other methods in the 6D case. Unsurprisingly, in the other cases, EIC exhibits lower regret than EIC^2, as EIC is informed about the true constraint threshold ($c = 0$), while EIC^2 learns it. While PIBU exhibits similar regret in the 10D case, it performs worse in 6D and 2D. The reason is that PIBU explores locally around the initial point in order to expand the safe area, hence, missing alternative safe areas of lower regret. Similarly, SafeOpt never leaves the initial safe region. Because SafeOpt scales poorly with dimensionality, we could not compute results for 6D and 10D.

TABLE I: Percentage of safe evaluations. Average and standard deviation $(\cdot)$ over 100 experiments.

	Michalewicz 10D	Hartman 6D	Egg crate 2D
MC + EI	50.48 (5.28)	63.54 (7.65)	70.38 (5.87)
HC + EI	53.02 (5.59)	64.53 (6.92)	69.98 (4.86)
AC + EI	51.39 (5.11)	64.16 (7.20)	69.19 (6.19)
EIC^2	77.90 (7.78)	69.40 (5.44)	79.41 (12.98)
PIBU	86.13 (0.97)	80.86 (1.95)	96.12 (1.38)
EIC	92.92 (7.93)	48.14 (15.71)	75.12 (4.66)
SafeOpt	–	–	100.00 (0.00)

Table I shows that EIC^2 finishes the exploration with a higher number of safe evaluations on average than the BO heuristics, as it leverages the information of GPCR to reason about unpromising areas. Also, EIC^2 finds the best trade-off between failure avoidance and performance while not being informed about the constraint threshold. The baseline, EIC, reaches a higher number of safe evaluations in 10D, and a similar number in 2D, while showing the lowest regret in 2D. However, EIC plays with two advantages with respect to

[3]https://github.com/alonrot/classified_regression

[4]Such upper bound is accessible in numerical benchmarks, but generally unknown when learning on real systems.

[5]https://github.com/etpr/con_bopt

EIC^2: (i) it is informed about the true constraint threshold and (ii) upon failure, it receives the true constraint observation.

In Fig. 3 (top), the constraint threshold $\hat{c}$ is estimated consistently near the true threshold ($c = 0$), i.e., a $\sim 5\,\%$ of the total range ($[-1, +1]$). On average, the $\hat{c}$ values are reported slightly above zero because the Gamma hyperprior $p(c)$ places the mean of the distribution slightly above $y_{\max}$ (cf. Sec. IV-B).

C. *Experiments on a real jumping quadruped*

Herein, we describe and analyze a set of experiments in which a quadruped learns to jump as high as possible. The higher the jump, the higher is the current needed to achieve it. Excessive demands of current cause a voltage drop that triggers a safe mode on the control boards. When this happens, all the motors shut down, which results in a failed jump, in which the robot lands improperly and crashes. The purpose of these experiments are two fold: (i) illustrate that EIC^2 exploits failures on its benefit to steer the search towards promising areas, although no data is obtained upon failure, and (ii) show that EIC^2 allows to estimate the maximum current above which the robot will fail.

1) Experimental setting: For the learning experiments we use Solo [31] (cf. Fig. 1), a lightweight quadruped with two degrees of freedom per leg that uses high-torque brushless DC motors, which enable very high vertical jumps. A kinodynamic planner [32] uses an approximate robot model to compute offline the needed feedforward torques and desired state trajectories for a high jump. To account for modeling errors, we use a PD controller with variable gains. Such gains change smoothly from an initial to a final value, both of which need to be tuned in order to achieve the desired jump. Poor tuning results in poor tracking and extremely low jumps. Hence, we use EIC^2 to automatically tune the gains toward better tracking and higher jumps. By using symmetry properties in the platform and the task, we can reduce the dimensionality of the tuning problem. To this end, we couple the P and D gains from all the knees into a single set of gains, and same for all the hips. In addition, we fix the final value of the P and D variable gains, which leaves four parameters to tune: $x = \left(\mathrm{P}^{\text{knee}}_{\text{init}}, \mathrm{P}^{\text{hip}}_{\text{init}}, \mathrm{D}^{\text{knee}}_{\text{init}}, \mathrm{D}^{\text{hip}}_{\text{init}}\right)^{\top}$. The parameters are normalized to lie within the unit hypercube $x \in [0,1]^4$.

For a certain controller parametrization that results in no failure, the cost objective is defined as $f(x) = -\max\{h_1, \ldots, h_\tau\}$, where h_k is the height of the jump in meters measured at time step k and τ is the number of time steps elapsed until landing. Each h_k is measured at the center of the robot base using a motion capture system which operates at $200\,\mathrm{Hz}$. The quadruped initiates all jumps from the same resting position, with the legs slightly flexed. In this position the robot is $0.25\,\mathrm{m}$ high. The constraint is defined as $g(x) = \sum_{m=1}^{8} i_m$, where i_m is the peak current measured in motor m during the jump. Constraint satisfaction is defined as $g(x) \leq c$, where c represents the maximum sum of currents, above which the robot will shut down and crash. The constraint threshold c is unknown a priori and can only be revealed with empirical evidence[6]. Critically, an underestimated threshold would wrongly reduce the space of successful controllers while an overestimated threshold would wrongly classify failure regions as successful. Instead, we let EIC^2 estimate the constraint threshold c using the GPCR that models g.

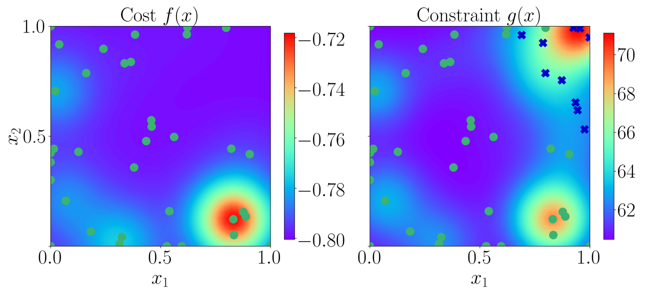

Fig. 4: Two-dimensional slice of the GP posterior mean, computed by fixing the third and fourth coordinates of the input vector, i.e, $x = (x_1, x_2, 0.7, 0.4)$. Successful evaluations (green dots) are shown in both plots, while failures (blue crosses) are shown only in the left plot. This is to emphasize that the GP that models f is not updated with the failure data points, while GPCR is updated using both: the successful observations and the failures.

The noise variances are measured by repeating ten times an initial safe jump and computing the empirical variance of the height and the current.

We start the experiments at a corner of the domain, i.e., $x = (0, 0, 1, 1)$ and run them for $M = 50$ iterations, after which we report the best successful observation obtained.

2) Results: After the exploration, EIC^2 succeeded to report a remarkably high jump. Importantly, the GPCR that models g was able to handle failures cases in which no current measurement was obtained, together with successful observations, in which the current is measured. In addition, GPCR was able to estimate the constraint threshold.

As shown in Fig. 4, most successful jumps were encountered during the search by EIC^2 in the upper half of the two-dimensional slice. However, 10 failures were encountered at the top right corner, as a consequence of high P gains. Between iterations 12 and 34, EIC^2 reported mostly high jumps mixed in with the failures. Because of this, it is likely that the constrained minima lies close to the constraint boundary. Consequently, underestimated user-defined thresholds could have discouraged such constraint boundary from being thoroughly explored, hence potentially missing high jumps. After iteration 34, EIC^2 led the search away from the region of failures and and targeted safer areas where jumps were generally not that high.

The controller parametrization that provided the highest jump was found at $x_{\text{highest}} = (0.62, 1.0, 0.44, 0.58)$ (iteration 26), which is close to the area of failures. After the search,

[6]Alternatively, a thorough analysis of the limitations of the DC motors given by the manufacturer could lead to a theoretical approximation of this value. However, this alternative typically requires expert knowledge, which might not always be available.

we double-checked x_{highest} by executing ten jumps, which resulted in a current of 104.48 ± 1.32 A and a height of 0.784 ± 0.0023 m, which is about $\times 3$ times higher than its resting position. The complementary video[7] summarizes the learning process and shows the learned jump. In [31], significant manual tuning was required to achieve high jumps on the same robot. However, their highest jump was reported at $0.65\,\text{m}$, while our framework automates the tuning process and gains about $13\,\text{cm}$ (about $34\,\%$ improvement). We also tested the estimated constrained global minimum x_{best} by solving (3) with $\delta = 0.1$. However, the performance was slightly worse than the one reported with x_{highest}.

The constraint threshold was reported by GPCR at $\hat{c} = 112.34\,\text{A}$. This value was unknown and unaccessible before the experiments, but learned as a hyperparameter of GPCR. This value can be stored for later implementations of safety mechanisms on the robot, e.g., for further learning experiments or demonstrations.

VI. Conclusions

In this paper, we have addressed the problem of learning with crash constraints, in which no data is obtained upon failure. To this end, we have proposed EIC^2, a constrained Bayesian optimization framework that models the constraint with GPCR, a novel GP model able to handle discrete and continuous observations. We demonstrated the effectiveness of EIC^2 in numerical benchmarks up to ten dimensions and on a real jumping quadruped, where the learned best jump improved by $34\,\%$ over the results obtained with manual tuning. Also, GPCR learned the constraint threshold, unknown a priori, which mitigates the need of expert knowledge to determine it. In future work, we will perform further robot experiments in more challenging and higher dimensional robot platforms with multiple constraints and compare EIC^2 against multi-output Gaussian process models.

[7]https://youtu.be/RAiIo0l6_rE

References

[1] 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," *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[2] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[3] 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. 1-2, pp. 5–23, 2016.

[4] A. Rai, R. Antonova, and S. Song et al., "Bayesian optimization using domain knowledge on the ATRIAS biped," in *International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1771–1778.

[5] M. H. Yeganegi, M. Khadiv, S. A. A. Moosavian, J. Zhu, A. Del Prete, and L. Righetti, "Robust humanoid locomotion using trajectory optimization and sample-efficient learning," in *International Conference on Humanoid Robots (Humanoids)*, 2019, pp. 170–177.

[6] K. Yuan, I. Chatzinikolaidis, and Z. Li, "Bayesian optimization for whole-body control of high-degree-of-freedom robots through reduction of dimensionality," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2268–2275, 2019.

[7] 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)*, 2016.

[8] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic LQR tuning based on Gaussian process global optimization," in *IEEE international conference on robotics and automation (ICRA)*, 2016.

[9] P. Englert and M. Toussaint, "Learning manipulation skills from a single demonstration," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 137–154, 2018.

[10] D. Drieß, P. Englert, and M. Toussaint, "Constrained bayesian optimization of combined interaction force/task space controllers for manipulations," in *International Conference on Robotics and Automation (ICRA)*, 2017, pp. 902–907.

[11] R. Antonova, A. Rai, and C. G. Atkeson, "Sample efficient optimization for learning controllers for bipedal locomotion," in *International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 22–28.

[12] R.-R. Griffiths and J. M. Hernández-Lobato, "Constrained Bayesian optimization for automatic chemical design," in *Conference on Neural Information Processing Systems (NIPS)*, 2017.

[13] J. Snoek, "Bayesian optimization and semiparametric models with applications to assistive technology," Ph.D. dissertation, University of Toronto, 2013.

[14] M. A. Gelbart, J. Snoek, and R. P. Adams, "Bayesian optimization with unknown constraints," in *Conference on Uncertainty in Artificial Intelligence*, 2014, pp. 250–259.

[15] M. Schonlau, W. J. Welch, and D. R. Jones, "Global versus local search in constrained optimization of computer models," *Lecture Notes-Monograph Series*, pp. 11–25, 1998.

[16] J. M. Hernández-Lobato, M. A. Gelbart, R. P. Adams, M. W. Hoffman, and Z. Ghahramani, "A general framework for constrained bayesian optimization using information-based search," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 5549–5601, 2016.

[17] D. V. Lindberg and H. K. Lee, "Optimization under constraints by applying an asymmetric entropy measure," *Journal of Computational and Graphical Statistics*, vol. 24, no. 2, pp. 379–393, 2015.

[18] F. Bachoc, C. Helbert, and V. Picheny, "Gaussian process optimization with failures: Classification and convergence proof," *HAL id: hal-02100819, version*, vol. 1, 2019.

[19] V. Donskoi, "Partially defined optimization problems: An approach to a solution that is based on pattern recognition theory," *Journal of Soviet Mathematics*, vol. 65, no. 3, pp. 1664–1668, 1993.

[20] C. Antonio, "Sequential model based optimization of partially defined functions under unknown constraints," *Journal of Global Optimization*, pp. 1–23, 2019.

[21] H. Lee, R. Gramacy, C. Linkletter, and G. Gray, "Optimization subject to hidden constraints via statistical emulation," *Pacific Journal of Optimization*, vol. 7, no. 3, pp. 467–478, 2011.

[22] T. Pourmohamad, H. K. Lee *et al.*, "Multivariate stochastic process models for correlated responses of mixed type," *Bayesian Analysis*, vol. 11, no. 3, pp. 797–820, 2016.

[23] Y. Zhang, Z. Dai, and K. H. Low, "Bayesian optimization with binary auxiliary information," in *35th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2019.

[24] B. Ru, A. S. Alvi, V. Nguyen, M. A. Osborne, and S. J. Roberts, "Bayesian optimisation over multiple continuous and categorical inputs," *arXiv preprint arXiv:1906.08878*, 2019.

[25] J. Mockus, V. Tiesis, and A. Zilinskas, "Toward global optimization," *Bayesian Methods for Seeking the Extremum*, vol. 2, 1978.

[26] D. Hernández-Lobato, V. Sharmanska, K. Kersting, C. H. Lampert, and N. Quadrianto, "Mind the nuisance: Gaussian process classification using privileged noise," in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 837–845.

[27] H. Nickisch and C. E. Rasmussen, "Approximations for binary Gaussian process classification," *Journal of Machine Learning Research*, vol. 9, no. Oct, pp. 2035–2078, 2008.

[28] J. P. Cunningham, P. Hennig, and S. Lacoste-Julien, "Gaussian probabilities and expectation propagation," *preprint arXiv:1111.6832*, 2011.

[29] Z. Wang and S. Jegelka, "Max-value entropy search for efficient Bayesian optimization," in *International Conference on Machine Learning (ICML)*, 2017, pp. 3627–3635.

[30] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimization problems," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, 2013.

[31] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, and M. Wüthrich et al., "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, 2020.

[32] B. Ponton, M. Khadiv, A. Meduri, and L. Righetti, "Efficient multi-contact pattern generation with sequential convex approximations of the centroidal dynamics," *preprint arXiv:2010.01215*, 2020.