Learning Flexible and Reusable Locomotion Primitives for a Microrobot

Brian Yang, Grant Wang, Roberto Calandra, Daniel Contreras, Sergey Levine, and Kristofer Pister

***Abstract*—The design of gaits for robot locomotion can be a daunting process, which requires significant expert knowledge and engineering. This process is even more challenging for robots that do not have an accurate physical model, such as compliant or micro-scale robots. Data-driven gait optimization provides an automated alternative to analytical gait design. In this letter, we propose a novel approach to efficiently learn a wide range of locomotion tasks with walking robots. This approach formalizes locomotion as a contextual policy search task to collect data, and subsequently uses that data to learn multiobjective locomotion primitives that can be used for planning. As a proof-of-concept we consider a simulated hexapod modeled after a recently developed microrobot, and we thoroughly evaluate the performance of this microrobot on different tasks and gaits. Our results validate the proposed controller and learning scheme on single and multiobjective locomotion tasks. Moreover, the experimental simulations show that without any prior knowledge about the robot used (e.g., dynamics model), our approach is capable of learning locomotion primitives within 250 trials and subsequently using them to successfully navigate through a maze.**

***Index Terms*—Learning and adaptive systems, micro/nano robots, legged robots.**

I. Introduction

SUBSTANTIAL progress has been made in recent years towards the development of fully autonomous microrobots [1], [2]. However, gait design for robot locomotion at the sub-centimeter scale is not a well-studied problem. Completing more complicated locomotion tasks like navigating complex environments is even more challenging. These issues become exacerbated when dealing with legged locomotion, where even walking straight is still an active area of study for normal-sized robots. In this letter, we present a novel approach for the autonomous optimization of locomotion primitives and gaits.

While locomotion on larger-scale robots has been thoroughly investigated, transferring many of these proven approaches to the millimeter scale poses many unique challenges. One such obstacle is the lack of access to sufficiently accurate simulated models at the millimeter scale. Even simulation environments designed to simulate dynamics at this scale are generally unequipped for usage in robotics contexts. Additionally, working with microrobots can place severe limitations on the number of iterations as trials become much more time-consuming and expensive to run.

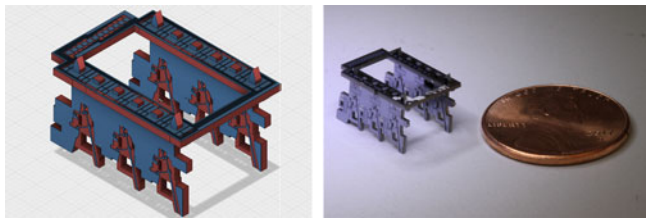

Fig. 1. The six-legged micro walker considered in our study as a CAD model (left) and an assembled prototype (right).

While microrobot locomotion has been addressed in the past, much of the work is primarily concerned with the mechanical design and manufacturing of microrobots. Accomplishing more sophisticated locomotion tasks on the sub-centimeter scale remains an open area for research. Analytical implementations of various gait behaviors have worked on microrobots [3], [4], but these solutions can become unwieldy for robots with higher DOF such as legged walkers (e.g., our micro-hexapod). Data-driven automatic gait optimization is a viable alternative to analytical gait design and optimization, but using these techniques can be challenging due to the high number of trials that might be necessary to perform in order to learn viable gaits.

Our contributions are two-fold: 1) we validate the use of both CPG controllers and Bayesian optimization for microrobots on a wide range of single and multi-objective locomotion tasks. 2) we introduce a novel approach to efficiently learn gaits and motor primitives from scratch without the need for prior knowledge (e.g., a dynamics model). This is accomplished by collecting data on various motor primitives using contextual policy search and using those evaluations to reformulate the problem into a multi-objective optimization task, providing us a model that can map any set of parameters to a predicted trajectory. Using this model, we can optimize our parameters on various trajectories for subsequent use in path planning. This approach is not tied exclusively to microrobots, but can be used for any walking robot.

To evaluate our approach, we used a simulated hexapod microrobot modeled after a recently developed microrobot [5] shown in Fig. 1. We first validate the use of a CPG controller

Manuscript received September 9, 2017; accepted January 27, 2018. Date of publication February 14, 2018; date of current version March 15, 2018. This letter was recommended for publication by Associate Editor M. Rakotondrabe and Editor Y. Sun upon evaluation of the reviewers' comments. This work was supported in part by the Berkeley Sensor and Actuator Center and in part by Berkeley DeepDrive. *(Brian Yang and Grant Wang contributed equally to this work.) (Corresponding author: Roberto Calandra.)*

The authors are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Berkeley, CA 94709 USA (e-mail: brianhyang@berkeley.edu; grant.wang5@berkeley.edu; roberto.calandra@berkeley.edu; dscontreras@berkeley.edu; svlevine@eecs.berkeley.edu; ksjp@berkeley.edu).

Digital Object Identifier 10.1109/LRA.2018.2806083

on our microrobot to reduce the number of parameters during optimization. Then, we validate the use of Bayesian optimization and existing techniques on a curriculum of progressively more difficult tasks including learning single-objective, contextual, and multi-objective gaits. As a proof of concept, we evaluated our approach by learning motor primitives from 250 trials and subsequently using them to successfully navigate through a maze.

II. Related Work

There has been an abundance of work published on the design and development of walking [6] and flying millimeter-scale microrobots [7]–[9]. Much of this work focuses on hardware considerations such as the design of micro-sized joints and actuators rather than control. To our knowledge, no previous work has implemented a CPG-based controller for on-board control of a walking microrobot, nor has learning been used for locomotion on a microrobots.

While hexapod gaits have been thoroughly studied and tested [10], [11], much of the work did not easily transfer to our microrobot due to the drastically different leg dynamics. Most hexapods make use of rotational joints with higher DOF while our walker uses only two prismatic spring joints per leg, resulting in less control and unique constraints on leg retraction and actuation.

While sufficient for simple controllers with few parameters, manually tuning controller parameters can require an immense amount of domain expertise and time. As such, automatic gait optimization is an important research field that has been studied with a wide variety of approaches in both the single-objective [12]–[19] and multi-objective setting [17], [18], [20], [21]. Evolutionary algorithms have been successfully used to train quadrupedal robots [13], [17], but this approach often requires thousands of experiments before producing good results, which is unfeasible on fragile microrobots.

A more data-efficient approach used before to learn gaits for snake and bipedal robots is Bayesian optimization [15], [16], [19], [22]. Bayesian optimization has been applied to contextual policy search in the context of robot manipulation [23]. Our contribution builds off of this work by applying and extending the contextual framework to learning movement trajectories and path planning. Another extension of Bayesian optimization related to our work is Multi-objective Bayesian optimization, which has also been previously applied in the context of robotic locomotion [21]. However, past work is only concerned with using multi-objective optimization to balance the trade-off between various competing goals. Our main contribution demonstrates an entirely novel application of multi-objective optimization to learning motor primitives that does not involve the trade-off between various goals, but instead uses a multi-objective model to learn over an area of possible trajectories for path planning.

III. The Hexapod Microrobot

We now introduce the hexapod microrobot considered in this letter. This robot is of particular interest due to the unique challenges that arise when attempting traditional gait design techniques. The micro-scale of the walker makes it very challenging to obtain an accurate dynamics model. Moreover, the robot is subject to wear-and-tear, and therefore any learning approach employed must be capable of learning gaits within a limited number of trials.

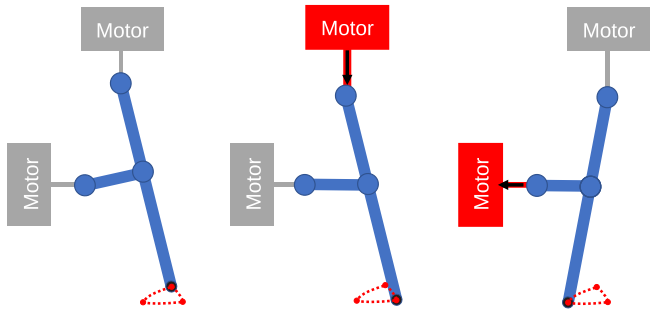

Fig. 2. Diagram of the robot leg showing the actuation sequence (active motors are shown in red). Each leg has 2 motors, each one independently actuating a single DOF.

A. Physical Description

The hexapod microrobot is based on silicon microelectromechanical systems (MEMS) technology. The robot's legs are made using linear motors actuating planar pin-joint linkages [24]. A tethered single-legged walking robot was previously demonstrated using this technology [5]. The hexapodal robot is assembled using three chips. The two chips on the side each have 3 of the leg assemblies, granting six 2 degree-of-freedom (DOF) legs for the whole robot. The top chip acts to hold the leg chips together for support, and to route the signals for off-board power and control. Overall, the robot measures 13 mm long by 9.6 mm wide and stands at 8 mm tall with an overall weight of approximately 200 mg.

B. Actuation

Each of the robot's legs has 2-DOF in the plane of fabrication, as shown in Fig. 2. Both DOFs are actuated, thus the leg has 2 motors, one to actuate the vertical DOF to lift the robot's body and a second to actuate the horizontal DOF for the vertical stride. The actuators used for the legs are electrostatic gap-closing inchworm motors [25]. During a full cycle, each leg moves 0.6 mm vertically with a horizontal stride of 2 mm. For more details on the actuation mechanism used on our microrobot, we refer readers to [26].

C. Simulator

In our experimental simulations, we used the robotics simulator V-REP [27] for constructing a scaled-up simulated model of the physical microrobot (see Fig. 3). Since V-REP was not designed with simulation of microrobots in mind, it was not capable of simulating the dynamics of the leg joints accurately and would produce wildly unstable models at the desired scale. We chose to scale up the size of the robot in simulation by a factor of 100 in order to account for the issues with scaling in simulation (all the experimental results are re-normalized to the dimensions of the real robot). We believe that this re-scaling still allows meaningful results to be produced for several reasons. First, the

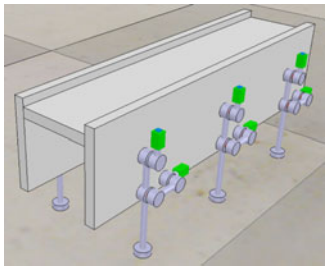

Fig. 3. The simulated micro walker.

experiments performed in this letter are meant to demonstrate the validity of the proposed controller, and the learning approach for training an actual physical microrobot. The policies trained are not meant to work on the real robot without any re-tuning or modification. Second, the simulator still allows to test the basic motion patterns we want to implement on the microrobot. Finally, our contribution lends credibility to the potential application of Bayesian-inspired optimization methods to a setting where evaluations can be costly and time consuming.

IV. Background

A. Central Pattern Generators

Central pattern generators (CPGs) are neural circuits found in nearly all vertebrates, which produce periodic outputs without sensory input [28]. CPGs are also a common choice for designing gaits for robot locomotion [29]. We chose to use CPGs for our controller because they are capable of reproducing a wide variety of different gaits simply by manipulating the relative coupling phase biases between oscillators. This allows us to easily produce a variety of gait patterns without having to manually program those behaviors. In addition, CPGs are not computationally intensive and can have on-chip hardware implementations using VLSI or FPGA. This makes them well suited to be eventually used in our physical microrobot, where the processing power is limited. CPGs can be modeled as a network of coupled non-linear oscillators where the dynamics of the network are determined by the set of differential equations

$$\dot{\phi}_i = \omega_i + \sum_j (\omega_{ij} r_j \sin(\phi_j - \phi_i - \varphi_{ij})) \,, \tag{1}$$

$$\ddot{r}_i = a_r \left(\frac{a_r}{4}(R_i - r_i) - \dot{r}_i \right) \,, \tag{2}$$

$$\ddot{x}_i = a_x \left(\frac{a_x}{4}(X_i - x_i) - \dot{x}_i \right) \,, \tag{3}$$

where ϕ_i is a state variable corresponding to the phase of the oscillations and ω_i is the target frequency for the oscillations. ω_{ij} and φ_{ij} are the coupling weights and phase biases which change how the oscillators influence each other. To implement our desired gaits, we only need to modify the phase biases between the oscillators ϕ_{ij}. r_i and x_i are state variables for the amplitude and offset of each oscillator, and R_i and X_i are control parameters for the desired amplitude and offset. The constants a_r and a_x are constant positive gains and allow us to control how quickly the amplitude and offset variables can be modulated. A more detailed explanation of the network can be found in Crespi's original work [30]. One of the foremost benefits of using a CPG controller is a drastic reduction in the number of parameters $\boldsymbol{\theta}_i$ we need to optimize. Overall, the parameters that we consider during the optimization are $\boldsymbol{\theta} = [\omega, R, X_l, X_r]$ where ω is the frequency of the oscillators and R is the phase difference between each of the vertical-horizontal oscillator pairs. In order to allow for directional control, X_l and X_r are the amplitudes of the left and right side oscillators respectively.

B. Bayesian Optimization

Even with a complete CPG network, some amount of parameter tuning is necessary to obtain efficient locomotion. To automate the parameter tuning, we use Bayesian optimization (BO), an approach often used for global optimization of black box functions [19], [31], [32]. We formulate the tuning of the CPG parameters as the optimization

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) \,, \tag{4}$$

where $\boldsymbol{\theta}$ are the CPG parameters to be optimized w.r.t. the objective function of choice f (e.g., walking speed, which we investigate in Section VI-B). At each iteration, BO learns a model $\tilde{f} : \boldsymbol{\theta} \rightarrow f(\boldsymbol{\theta})$ from the dataset of the previously evaluated parameters and corresponding objective values measured $\mathcal{D} = \{\boldsymbol{\theta}, f(\boldsymbol{\theta})\}$. Subsequently, the learned model $\tilde{f}$ is used to perform a "virtual" optimization through the use of an acquisition function which controls the trade-off between exploration and exploitation. Once the model is optimized, the resulting set of parameters $\boldsymbol{\theta}^*$ is finally evaluated on the real system, and is added to the dataset together with the corresponding measurement $f(\boldsymbol{\theta}^*)$ before starting a new iteration. A common model used in BO for learning the underlying objective, and the one that we consider, is Gaussian processes [33]. For more information regarding BO, we refer the readers to [32], [34].

C. Multi-Objective Bayesian Optimization

A special case of the optimization task of (4) is multi-objective optimization [35]. Often times in robotics[1], there are multiple conflicting objectives that need to be optimized simultaneously, resulting in design trade-offs (e.g., walking speed vs energy efficiency which we investigate in Section VI-C). When multiple objectives are taken into consideration, there is no longer necessarily a single optimum solution, but rather the goal of the optimization became to find the set of Pareto optimal solutions [37], which also takes the name of Pareto front (PF). Formally, the PF is the set of parameters that are not dominated, where a set of parameters $\boldsymbol{\theta}_1$ is said to dominate $\boldsymbol{\theta}_2$ when

$$\begin{cases} \forall i \in \{1, \ldots, n\} : \;\; f_i(\boldsymbol{\theta}_1) \leq f_i(\boldsymbol{\theta}_2) \\ \exists j \in \{1, \ldots, n\} : \;\; f_j(\boldsymbol{\theta}_1) < f_j(\boldsymbol{\theta}_2) \end{cases} \tag{5}$$

Intuitively, if $\boldsymbol{\theta}_1 \succ \boldsymbol{\theta}_2$, then $\boldsymbol{\theta}_1$ is preferable to $\boldsymbol{\theta}_2$ as it never performs worse, but at least in one objective function it performs strictly better. However, different dominant variables are equivalent in terms of optimality as they represent different trade-offs.

[1] As well as in nature [36].

Multi-objective optimization can often be difficult to perform as it might require a significant amount of experiments. This is especially true with our microrobot where large number of experiments can wear-and-tear the robot. As a result, the number of evaluations allowed to find the Pareto set of solutions is limited. Luckily for us, there exist extensions of BO which address multi-objective optimization. In particular, the multi-objective Bayesian optimization algorithm that we consider is ParEGO [38]. The main intuition of ParEGO is that at every iteration, the multiple objectives can be randomly scalarized into a single objective (via the augmented Tchebycheff function), which is subsequently optimized as in the standard Bayesian optimization algorithm (by creating a response surface, and then optimizing its acquisition function). For more information about multi-objective Bayesian optimization we refer the reader to [39].

D. Contextual Bayesian Optimization

Another special case of the optimization task of (4), is contextual optimization. In contextual optimization, we assume that there are multiple correlated, but slightly different, tasks which we want to solve, and that they are identified by a context variable $\boldsymbol{c}$. An example (which we investigate in Section VI-E) might be walking on inclined slopes, where the contextual variable is the angle of the slope. The contextual optimization can hence be formalized as

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} f(\boldsymbol{\theta}, \boldsymbol{c}) \,, \tag{6}$$

where for each context $\boldsymbol{c}$, a potentially different set of parameters $\boldsymbol{\theta}^*$ exists. The main advantage compared to treating each task independently is that, in contextual optimization, we can exploit the correlation between the tasks to generalize, and as a result quickly learn how to solve a new context. Specifically, in this letter we consider contextual Bayesian optimization (cBO) [23] which extends the classic BO framework from Section IV-B. Contextual Bayesian Optimization learns a joint model $\tilde{f} : \{\boldsymbol{\theta}, \boldsymbol{c}\} \rightarrow f(\boldsymbol{\theta})$, but now, at every iteration the acquisition function is optimized with a constrained optimization where the context $\boldsymbol{c}$ is provided by the environment. However, because the model jointly model the context-parameter space, experience learned in one context can be generalized to similar contexts. By utilizing cBO, we will show in Section VI that our microrobot can learn to walk (and generalize) to different environmental contexts such as walking uphill and curving.

V. Learning Locomotion Primitives for Path Planning

We now present our novel approach to learn motor primitives for path planning. This approach relies on the possibility of reusing the evaluations collected using cBO to convert the task into a multi-objective optimization problem. We specifically consider a cBO task where we want to optimize the parameters $\boldsymbol{\theta}$ to reach different target positions $\boldsymbol{c} = [\Delta x_{\text{des}}, \Delta y_{\text{des}}]$ (this setting is evaluated in Section VI-F). The objective function in this case can be defined as the Euclidean distance

$$f = \sqrt{(\Delta x_{\text{des}} - \Delta x_{\text{obs}})^2 + (\Delta y_{\text{des}} - \Delta y_{\text{obs}})^2} \,, \tag{7}$$

where $\Delta x_{\text{obs}}, \Delta y_{\text{obs}}$ are the actual positions measured after evaluating a set of parameters. The cBO model would map $\tilde{f} : [\boldsymbol{\theta}, \Delta x_{\text{des}}, \Delta y_{\text{des}}] \rightarrow f(\boldsymbol{\theta})$. However, in order to compute f it would need to measure $\Delta x_{\text{obs}}, \Delta y_{\text{obs}}$, effectively generating data of the form

$$[\boldsymbol{\theta}, \Delta x_{\text{des}}, \Delta y_{\text{des}}] \rightarrow [\Delta x_{\text{obs}}, \Delta y_{\text{obs}}, f(\boldsymbol{\theta})] \tag{8}$$

We can now re-use the data generated from this contextual optimization to learn a motor primitive model in the form $g : \boldsymbol{\theta} \rightarrow [\Delta x_{\text{obs}}, \Delta y_{\text{obs}}]$. The purpose of this learned model g is now to provide an estimate of the final displacement obtained for a set of parameters independently from the optimization process that generated it. Once such a model is learned, we can use it to compute parameters that lead to the desired displacement $\Delta x^*_{\text{obs}}, \Delta y^*_{\text{obs}}$ by optimizing the parameters w.r.t. the output of the model

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} z(g(\boldsymbol{\theta})) \,, \tag{9}$$

where z is a scalarization function of our choice (e.g., the Euclidean distance). This is equivalent to learning a continuous function that generates motor primitives from the desired displacement. It should be noted that this optimization is performed on the model g and therefore does not require any physical interaction with the robot. Moreover, we can optimize the parameters over a series of multiple displacements to obtain a path planning optimization. In Section VI-G, when performing path planning using the learned motor primitives we will employ a simple shooting method optimization which randomly samples multiple candidate parameters and selects the best outcome.

VI. Experimental Simulation Results

In this section, we discuss our controller implementation as well as the performance of our simulated microrobot on various locomotion tasks. The code used for performing the simulation and videos of the various locomotion tasks are available online at https://sites.google.com/view/learning-locomotion-primitives.

A. Controller Implementation

We built our controller following the setup described in Section IV-A, using a network of 12 coupled phase oscillators (one per motor). In order to translate the output of each of the oscillators into motor actuation, we calculate the oscillator outputs for each vertical-horizontal motor pair using the piecewise function

$$\begin{cases} x_i + r_i \cos(\phi_i), x_j + r_j \cos(\phi_j) & \text{if } \phi_i > \pi, \phi_j > \pi \,, \\ x_i + r_i, x_j + r_j \cos(\phi_j) & \text{if } \phi_i \leq \pi, \phi_j > \pi \,, \\ x_i + r_i, x_j + r_j & \text{if } \phi_i \leq \pi, \phi_j \leq \pi \,, \\ x_i + r_i \cos(\phi_i), x_j - r_j & \text{if } \phi_i \leq \pi, \phi_j > \pi \,, \end{cases} \tag{10}$$

where the ith oscillator outputs to its respective vertical motor and the jth oscillator outputs to its respective horizontal motor. This allows us to discard the parts of the oscillator output that are not consistent with the physical constraints of the physical robot, since the actual leg actuators cannot partially retract (see Fig. 4). We choose to mutually couple all six of the vertical oscillators (with a coupling weight of 4 to ensure quick convergence on

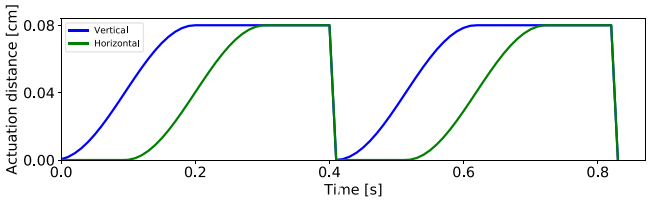

Fig. 4. Output of one vertical-horizontal oscillator pair in the CPG network, which corresponds to one leg on the robot. The retraction phase of both motors occurs concurrently and rapidly in order to simulate the physical constraints on the actual physical microrobot.

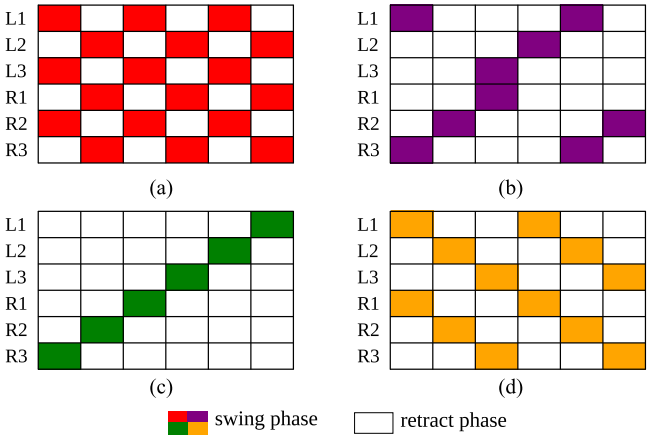

Fig. 5. Contact/swing patterns for different gaits.

stable limit cycles). We refer the reader to [30] for a more comprehensive discussion of oscillator coupling in CPGs. Each of the horizontal oscillators are also coupled with their respective vertical oscillator in order to encapsulate the dynamics of each leg. We chose to implement four different gaits with the CPG – tripod, ripple, wave, and four-two (see Fig. 5). For a more detailed description of these gaits we refer the reader to [40]. We use the same frequency and phase difference for the whole network in order to reduce the number of parameters and speed up the rate of convergence. We use two separate parameters for amplitude, each controlling the left and right set of legs respectively. This choice of parameters allows us to control the turning of the robot which is necessary for path planning and corrections for not walking straight.

B. Learning to Walk Straight

We optimized the four gaits considered (i.e., dual tripod, ripple, wave, and four-two) using as our objective function the walking speed of the robot (measured as the distance traveled after 1 s). Since some gaits result in curved motions, we also penalized the speed objective with a term proportional to the drift from the axis of locomotion. The optimization used the 4 parameters outlined in Section IV-A and was repeated 50 times for each of the gaits. In Fig. 6, we show the median and 65th percentiles of the best solution obtained so far in the trials. The results show that the optimizer was able to learn to walk from scratch within 50 iterations. Moreover, it can be noted that the optimized tripod and ripple are the fastest gaits at $\sim$1.1 cm/s and $\sim$1.2 cm/s respectively.

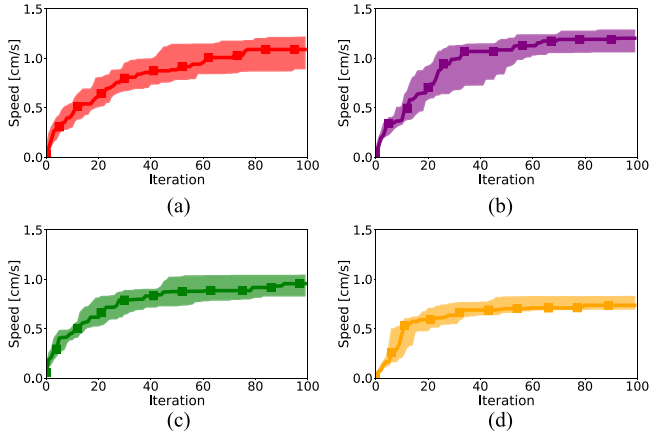

Fig. 6. Learning curve for the four gaits (median and 65th percentile). We can see how, for all the gaits, BO learns to walk from scratch within 50 iterations. After the optimization, Dual Tripod and Ripple are the fastest gaits at $\sim$1.1 cm/s and $\sim$1.2 cm/s respectively. (a) Dual Tripod. (b) Ripple. (c) Wave. (d) Four-Two.

C. Multi-Objective Gait Optimization

In the previous simulation we only considered walking speed as our objective. However, for practical gait design, energy efficiency is another objective of great interest, particularly when it comes to designing gaits for a microrobot with real energy restrictions. For this reason, we now consider a multi-objective optimization setting and compare the different gaits w.r.t. both walking speed, and energy consumption. The energy consumption of the robot was computed by measuring the forces exerted by each of the 12 motors along the axis of actuation and calculating the power used to actuate the motors. Since the retraction of the legs is spring powered, the energy input in the cycle is only during motor extension. Hence, we only consider the cost of extending the legs. With the mass of the robot and the time of each trial being held constant, we quantify the energy efficiency of a gait and estimate the cost of transport.

We optimized the four gaits again with the same 4 parameters as the previous optimization, but this time using multi-objective Bayesian optimization with a budget of 50 iterations.

In Fig. 7 we can see the performance measured and Pareto fronts obtained for the different gaits. To better compare the PF from the different gaits, we also visualized just the PFs together in Fig. 8. From these results, we can see how the tripod gait dominates the other gaits for speed 0.6 cm/s, while Ripple dominates when the speed is $>$0.6 cm/s, hence giving a clear indication of which gait is preferable under different circumstances.

D. Discovering New Gaits With Multi-Objective Optimization

In addition to optimizing the four nature-inspired gaits, we also tested multi-objective optimization on the walker without constraining to using predefined gaits. To parametrize the oscillator couplings, we thus discretized each gait into intervals of constant length. Within each of these intervals, we assume that each leg steps exactly once, keeping each of the oscillators in the CPG in phase with each other. This allows us to parametrize gaits by assigning each leg a point during each interval where it begins stepping. While this parametrization excludes certain

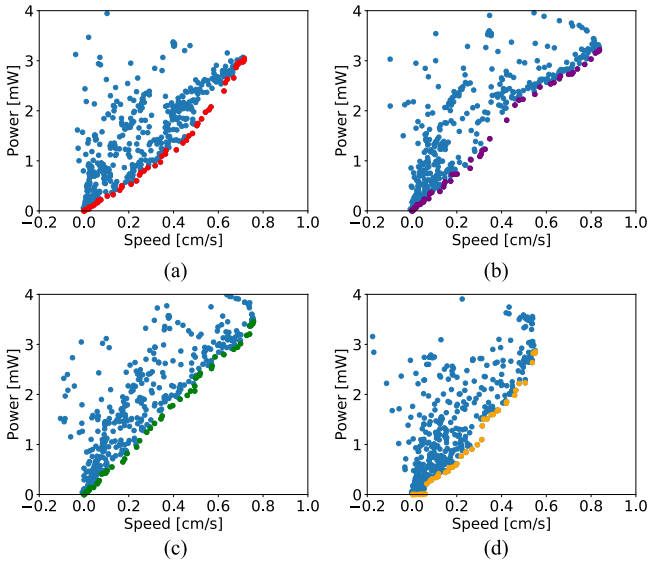

Fig. 7. Performance measured for the four gaits, and the corresponding PFs. ParEGO is able to quickly explore the PF for each of our four gaits. (a) Dual Tripod. (b) Ripple. (c) Wave. (d) Four-Two.

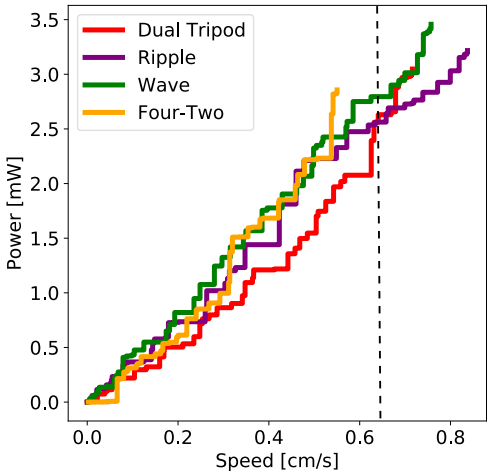

Fig. 8. Comparison of the PFs obtained for the different gaits.

gaits that cannot be expressed in this form, we leave the study of more sophisticated gait parameterizations for gait discovery to future works.

The resulting multi-objective optimization task had 8 parameters (frequency, phase difference between horizontal and vertical motors, and the six gait coupling parameters). Due to the higher parameter dimensionality, and because this training was not intended for on-line training, we ran the optimization for 250 iterations in order to allow a more comprehensive exploration of the optimization space. We also repeated the optimization five times for a total of 1250 trials. In Fig. 9 we can see the Pareto front for the resulting gaits. We found that the fastest discovered gaits were actually able to outperform the four nature-inspired gaits implemented by a substantial margin. Even while penalizing curved paths, the fastest discovered gait outperformed Ripple (the fastest nature-inspired gait we found) by almost 50%. However, for low-speed gaits, the nature inspired gaits

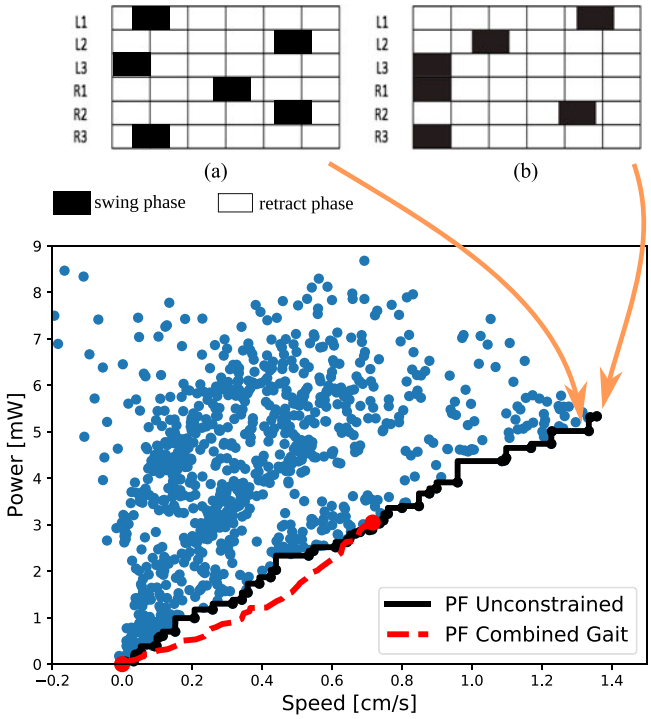

Fig. 9. PF of the unrestrained gait optimization versus the best performance of the four nature-inspired gaits. The faster solutions outperform the fastest nature-inspired gaits, albeit with more energy expenditure. However, the inability of the optimizer to match the performance of the gaits at lower speeds within 1250 trials shows that the gait parametrization can help limit the search space to find better solutions easier. *(top)* Pattern for two of the discovered gaits.

out-perform the gaits produced by the unconstrained optimization, indicating the optimization did not yet fully converged to the optimal PF.

E. Learning to Walk on Inclined Surfaces

We now consider the case of contextual optimization and specifically the task of gait optimization for slopes with different inclinations. We framed learning to walk on inclined terrain as a contextual policy search, where the angle of the inclination is the context. In this simulation, we decided to use Dual Tripod for our gait with mostly the same open parameters as the previous simulations. We used a single parameter to represent the amplitude for the entire network in order to keep the number of parameters low with the addition of a contextual variable, leaving us with 3 parameters and 1 contextual parameter. To respect real world constraints, where testing randomly sampled incline angles over a continuous interval can be excessively time-consuming, we chose at training time to perform simulations only from a small number of inclines: 5, 10, and 15 degrees.

After optimizing the gaits for these three inclines over 50 iterations, we studied how the contextual optimizer is able to generalize across the context space by testing the performance of the contextual policy for a wide range of inclines. In Fig. 10 we can see that the policy performs well on intermediary inclines and seems to smoothly interpolate between the training inclines as is desirable. The gradual decrease in performance as the

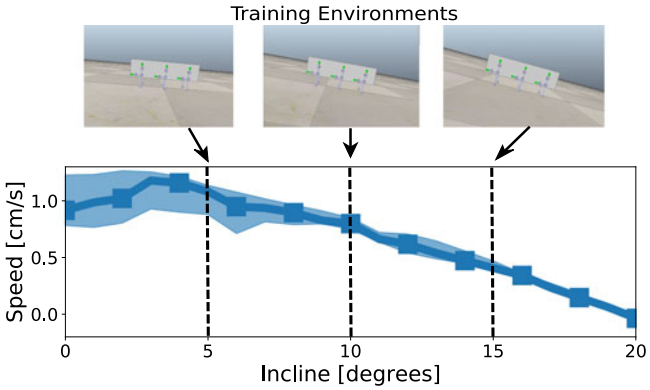

Fig. 10. Performance of the contextual policy (median and 65th percentile) for a wide range of inclines. The policy was trained only at 5, 10 and 15 degrees, but it was capable of generalizing smoothly to unseen inclinations.

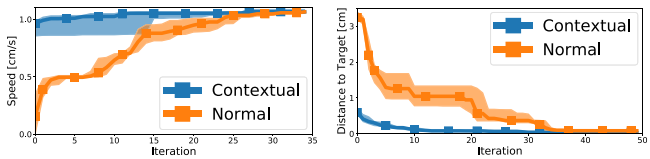

Fig. 11. Comparison between the optimization performance of a contextual optimizer and a normal optimizer for two different tasks: (a) walking on inclines (b) walking curved trajectories. In both cases, the contextual optimizer can leverage prior simulations to obtain high-performing gaits in fewer simulations. (a) Inclined surface. (b) Curved trajectory.

inclines get steeper can be attributed to the increasing physical difficulty for climbing up steeper inclines. We also compared cBO against using standard BO to train the robot for an untested incline. As shown in Fig. 11(a), the contextual optimization was able to converge on optimal performance significantly faster than standard BO. This result demonstrate the ability of cBO to efficiently use data accumulated in previous contexts to quickly reach optimize gaits in new unseen contexts.

F. Learning to Curve

Another useful task that can be framed as contextual optimization is learning motor primitives to walk curved trajectories for use in path planning. We used the same parameters as in Section VI-B and the contextual parameters in this case were the target displacements along both the x and y axes from the point of origin. In order to train particular trajectories, we selected five evenly spaced target points along the front quadrant of the field of vision. Since the primary objective was to reach the desired destination, we chose to use the distance of the final position to the target position as our sole objective function. We found that over 10 repetitions, the walker was able to accurately move and turn towards all of the target points within 250 iterations. In Fig. 11(b), we compared the performance of cBO against standard BO on a previously unseen target position $(4\cos\pi/16, 4\sin\pi/16)$. We found that, as in the case of inclinations, the contextual policy was able to learn the optimal parameters for a novel trajectory within very few iterations.

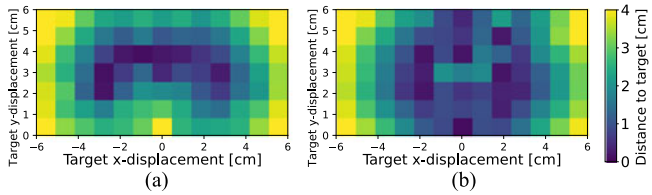

Fig. 12. Comparison of the performances of cBO and our approach for learning motor primitives (using the same data). With the robot having an initial position of $(0, 0)$, we evaluated the error between the desired position (indicated by the element of the grid) and the reached position. Darker color indicates better target accuracy. While cBO accurately learned trajectories near the training targets, it did not generalize well to unseen targets. In contrast, our approach had a more comprehensive coverage as it could leverage better information about the environment to improve generalization.

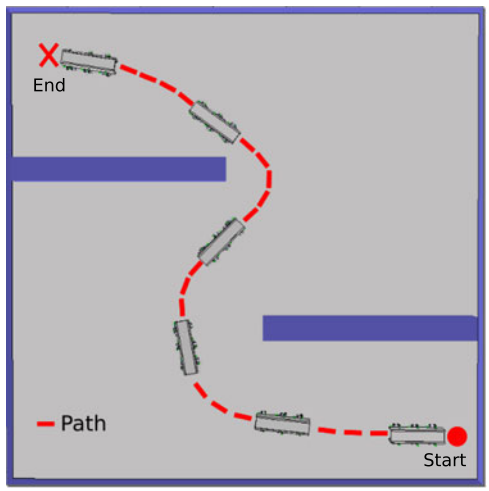

Fig. 13. Path constructed using the locomotion primitives learned with our approach.

G. Learning Motor Primitives for Path Planning

In the previous simulation we learned motor primitives capable of walking curved trajectories. While the model handled trajectories near and between the targets quite well, the performance on trajectories well within the physical capabilities of the robot but not in proximity to the targets left much to be desired, as shown in Fig. 12. We now demonstrate how our approach presented in Section V can be used to significantly improve the movement accuracy (compared to cBO using the same data), as well as how such motor primitives can be used to perform path planning. First, we reused the data from the previous simulation in order to reformulate the task as a multi-objective optimization as described in Section V. Then, we used our trained model to sample 10,000 trajectories by randomly sampling from the parameter space. Out of all these trajectories, we selected the one with the smallest expected error subject to not walking through the wall. Evaluating the resulting sequence of motor primitives on the real system (i.e., the simulator) demonstrated that the expected trajectory was capable of navigating the maze, as shown in Fig. 13.

VII. Conclusion

Designing controllers for locomotion is a daunting task. In this letter, we demonstrated on a simulated microrobot that

this process can be significantly automated. Our main contributions are two-fold: 1) we introduced a coherent curriculum of increasing challenging tasks, which we use to evaluate the CPG controller of our microrobot using Bayesian optimization. 2) we presented a new approach that enables walking robots to efficiently learn motor primitives from scratch. By using the data collected from contextual optimization we reformulate the problem into a multi-objective optimization task, and learn a model that can map any set of parameters to a predicted trajectory. This model can subsequently be used for path planning. Our experimental simulation results demonstrate that using this approach a microrobot can successfully learn accurate locomotion primitives within 250 trials, and subsequently use them to navigate through a maze, without any prior knowledge about the environment or its own dynamics.

The gaits obtained on the simulated microrobot might not yield good results when applied to the real microrobot, due to the low-fidelity of the simulator used. However, the methodology used to obtain them is realistically applicable to real microrobots, and is uniquely able to address concerns that exist on the sub-centimeter scale (e.g., lack of a precise physics simulator and budgeting of physical experiments). In future work, we plan to evaluate our approach and findings on the physical hexapod microrobot.

REFERENCES

[1] K. Saito, K. Iwata, Y. Ishihara, K. Sugita, M. Takato, and F. Uchikoba, "Miniaturized rotary actuators using shape memory alloy for insect-type MEMS microrobot," *Micromachines*, vol. 7, no. 4, 2016, doi: 10.3390/mi7040058.

[2] D. Vogtmann, R. S. Pierre, and S. Bergbreiter, "A 25 mg magnetically actuated microrobot walking at >5 body lengths/sec," in *Proc. IEEE Int. Conf. Micro Electro Mech. Syst.*, Jan. 2017, pp. 179–182.

[3] T. Ebefors, J. U. Mattsson, E. Klvesten, and G. Stemme, "A walking silicon micro-robot," in *Proc. Int. Conf. Solid-State Sensors Actuators*, 1999, pp. 1202–1205.

[4] S. Hollar, A. Flynn, C. Bellew, and K. S. J. Pister, "Solar powered 10 mg silicon robot," in *Proc. IEEE Int. Conf. Micro Electro Mech. Syst.*, 2003, pp. 706–711.

[5] D. S. Contreras, D. S. Drew, and K. S. Pister, "First steps of a millimeter-scale walking silicon robot," in *Proc. Int. Conf. Solid-State Sensors, Actuators Microsyst.*, 2017, pp. 910–913.

[6] F. D. Ambroggi, L. Fortuna, and G. Muscato, "PLIF: Piezo light intelligent flea-new micro-robots controlled by self-learning techniques," in *Proc. Int. Conf. Robot. Autom.*, Apr. 1997, vol. 2, pp. 1767–1772.

[7] R. J. Wood, "The first takeoff of a biologically inspired at-scale robotic insect," *IEEE Trans. Robot.*, vol. 24, no. 2, pp. 341–347, Apr. 2008.

[8] D. S. Drew and K. S. J. Pister, "First takeoff of a flying microrobot with no moving parts," in *Proc. Int. Conf. Manipulation, Autom. Robot. Small Scales*, 2017, pp. 1–5.

[9] B. G. Kilberg, D. S. Contreras, J. Greenspun, and K. S. J. Pister, "Mems aerodynamic control surfaces for millimeter-scale rockets," in *Proc. Int. Conf. Manipulation, Autom. Robot. Small Scales*, 2017, pp. 1–5.

[10] R. Altendorfer *et al.* "Rhex: A biologically inspired hexapod runner," *Auton. Robots*, vol. 11, no. 3, pp. 207–213, 2001.

[11] A. M. Hoover, E. Steltz, and R. S. Fearing, "Roach: An autonomous 2.4 g crawling hexapod robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 26–33.

[12] R. Tedrake, T. W. Zhang, and H. S. Seung, "Stochastic policy gradient reinforcement learning on a simple 3d biped," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2004, vol. 3, pp. 2849–2854.

[13] S. Chernova and M. Veloso, "An evolutionary approach to gait learning for four-legged robots," in *Proc. Int. Conf. Intell. Robots Syst.*, 2004, vol. 3, pp. 2562–2567.

[14] C. Niehaus, T. Röfer, and T. Laue, "Gait optimization on a humanoid robot using particle swarm optimization," in *Proc. Second Workshop Humanoid Soccer Robots*, 2007.

[15] D. J. Lizotte, T. Wang, M. Bowling, and D. Schuurmans, "Automatic gait optimization with Gaussian process regression," in *Proc. Int. Joint Conf. Artif. Intell.*, 2007, pp. 944–949.

[16] M. Tesch, J. Schneider, and H. Choset, "Using response surfaces and expected improvement to optimize snake robot gait parameters," in *Proc. Int. Conf. Intell. Robots Syst.*, 2011, pp. 1069–1074.

[17] M. Oliveira, L. Costa, A. Rocha, C. Santos, and M. Ferreira, "Multi-objective optimization of a quadruped robot locomotion using a genetic algorithm," in *Soft Computing in Industrial Applications*, vol. 96. Berlin, Germany: Springer-Verlag, 2011, pp. 427–436.

[18] M. Oliveira, V. Matos, C. P. Santos, and L. Costa, "Multi-objective parameter CPG optimization for gait generation of a biped robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 3130–3135.

[19] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian optimization for learning gaits under uncertainty," *Annals Math. Artif. Intell.*, vol. 76, no. 1, pp. 5–23, 2015.

[20] G. Capi, M. Yokota, and K. Mitobe, "A new humanoid robot gait generation based on multiobjective optimization," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatron.*, 2005, pp. 450–454.

[21] M. Tesch, J. Schneider, and H. Choset, "Expensive multiobjective optimization for robotics," in *Proc. Int. Conf. Robot. Autom.*, 2013, pp. 973–980.

[22] R. Antonova, A. Rai, and C. G. Atkeson, "Deep kernels for optimizing locomotion controllers," in *Proc. Conf. Robot Learn.*, 2017, pp. 47–56.

[23] J. H. Metzen, A. Fabisch, and J. Hansen, "Bayesian optimization for contextual policy search," in *Proc. IROS Workshop Mach. Learn. Plan. Control Robot. Motion*, 2015.

[24] D. S. Contreras and K. S. J. Pister, "Durability of silicon pin-joints for microrobotics," *Int. Conf. Manipulation, Autom. Robot. Small Scales*, 2016, pp. 1–6.

[25] I. Penskiy and S. Bergbreiter, "Optimized electrostatic inchworm motors using a flexible driving ARM," *J. Micromech. Microeng.*, vol. 23, no. 1, Jan. 2013.

[26] D. S. Contreras and K. S. J. Pister, "Dynamics of electrostatic inchworm motors for silicon microrobots," *Int. Conf. Manipulation, Autom. Robot. Small Scales*, 2017, pp. 1–6.

[27] "Robot simulator V-REP," 2018. [Online]. Available: http://www.coppeliarobotics.com/

[28] J. Yu, M. Tan, J. Chen, and J. Zhang, "A survey on CPG-inspired control models and system implementation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 3, pp. 441–456, Mar. 2014.

[29] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Netw.*, vol. 21, no. 4, pp. 642–653, 2008.

[30] A. Crespi, D. Lachat, A. Pasquier, and A. J. Ijspeert, "Controlling swimming and crawling in a fish robot using a central pattern generator," *Auton. Robots*, vol. 25, no. 1/2, pp. 3–13, Dec. 2007.

[31] H. J. Kushner, "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise," *J. Basic Eng.*, vol. 86, pp. 97–106, 1964.

[32] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *J. Global Optim.*, vol. 21, no. 4, pp. 345–383, 2001.

[33] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.

[34] 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.

[35] J. Branke, K. Deb, K. Miettinen, and R. Slowiński, *Multiobjective Optimization: Interactive and Evolutionary Approaches*, vol. 5252. Berlin, Germany: Springer-Verlag, 2008.

[36] D. F. Hoyt and C. R. Taylor, "Gait and the energetics of locomotion in horses," *Nature*, vol. 292, no. 5820, pp. 239–240, 1981.

[37] V. Pareto, *Manuale di Economia Politica*, vol. 13. Milano, Italy: Edizioni Studio Tesi, 1906.

[38] J. Knowles, "ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 50–66, Jan. 2006.

[39] T. Wagner, M. Emmerich, A. Deutz, and W. Ponweiser, "On expected-improvement criteria for model-based multi-objective optimization," in *Parallel Problem Solving from Nature (PPSN) XI*. Berlin, Germany: Springer-Verlag, 2010, pp. 718–727.

[40] R. Campos, V. Matos, and C. Santos, "Hexapod locomotion: A nonlinear dynamical systems approach," in *Proc. Annu. Conf. IEEE Ind. Electron. Soc.*, Nov. 2010, pp. 1546–1551.