Data-efficient Learning of Morphology and Controller for a Microrobot

Thomas Liao[1], Grant Wang[1], Brian Yang[1], Rene Lee[1], Kristofer Pister[1], Sergey Levine[1], and Roberto Calandra[2]

***Abstract*— Robot design is often a slow and difficult process requiring the iterative construction and testing of prototypes, with the goal of sequentially optimizing the design. For most robots, this process is further complicated by the need, when validating the capabilities of the hardware to solve the desired task, to already have an appropriate controller, which is in turn designed and tuned for the specific hardware. In this paper, we propose a novel approach, HPC-BBO, to efficiently and automatically design hardware configurations, and evaluate them by also automatically tuning the corresponding controller. HPC-BBO is based on a hierarchical Bayesian optimization process which iteratively optimizes morphology configurations (based on the performance of the previous designs during the controller learning process) and subsequently learns the corresponding controllers (exploiting the knowledge collected from optimizing for previous morphologies). Moreover, HPC-BBO can select a "batch" of multiple morphology designs at once, thus parallelizing hardware validation and reducing the number of time-consuming production cycles. We validate HPC-BBO on the design of the morphology and controller for a simulated 6-legged microrobot. Experimental results show that HPC-BBO outperforms multiple competitive baselines, and yields a 360% reduction in production cycles over standard Bayesian optimization, thus reducing the hypothetical manufacturing time of our microrobot from 21 to 4 months.**

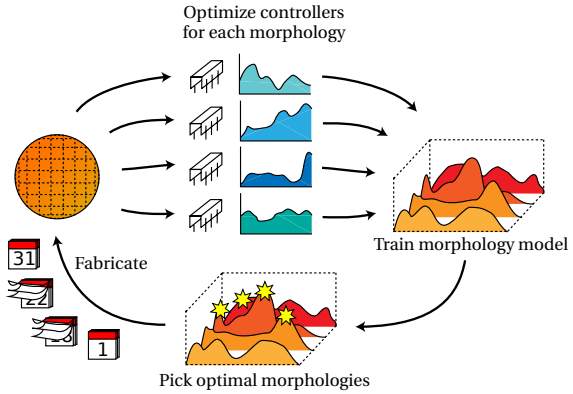

Fig. 1: Our approach for learning morphology and controller is motivated by a setting where the manufacturing of a batch of microrobots is costly (both in terms of time and resources), but multiple robots can be produced in parallel on the same silicon waffer. Each robot can independently learn a controller and, at the end of the learning, a new batch of morphologies is selected to be manufactured next, based on the performance achieved by the current prototypes.

I. Introduction

Designing intelligent robots to solve complex real-world tasks can be a daunting challenge. The dominant paradigm is based on the flawed assumption that the morphology of a robot (i.e., the hardware) can to a large extent be designed independently of the underlying controllers. In practice, this results in either designing general-purpose morphologies which can in theory solve a wide range of tasks, at the expense of being sub-optimal for any specific one, or in using an iterative process where each morphology design is followed by the design of an appropriate controller, with modifications (based on expert knowledge) to the previous morphology design to improve the chances of achieving a better controller at the next iteration. Both of these approaches usually require a significant amount of expert knowledge which heavily influences the ultimate performance of the system. Moreover, this process often requires a significant amount of design *and* manufacturing time for each morphology and controller.

One alternative to this paradigm is to autonomously optimize both morphology and controller, based directly on the performance achieved on the specific application of interest. This paradigm is conceptually similar to the evolutionary theory of Lamarckian inheritance [1], [2], where the physical features of a species are directly guided through the generation by the "usefulness" of the features to the specific environment inhabited. Although this evocative idea has already been proposed in past robotic literature [3], [4], the proposed approaches are difficult to apply to real-world applications due to the need for accurate analytical models, or a high number of evaluations to find good morphology/controller configurations.

The motivating application of this paper is the design of a hexapod microrobot and its corresponding controller. Due to the lack of sufficiently accurate models at the micro-scale, previous efforts [5] have proven dependent on expert knowledge. This robot is printed on silicon wafer, involving an expensive process with long production times (wafer delivery approximately 4-6 weeks after ordering). With the current manufacturing process, each wafer has sufficient space to contain up to 5 printed robots. After printing and assembly, it is still necessary to design a controller for the micro-robot, another design process largely guided by human experience. Moreover, it is important to design this controller within a small number of experiments before the robot wears out.

*This work was supported by Berkeley DeepDrive
[1] University of California, Berkeley, CA, USA
{thomas.liao, grant.wang5, brianhyang, renelee, ksjp}@berkeley.edu, svlevine@eecs.berkeley.edu
[2] Facebook AI Research, Menlo Park, CA, USA
rcalandra@fb.com

As a result of these real-world manufacturing and design constraints, we focus in this paper on the need for an efficient optimization process, able to find suitable designs within an extremely limited number of iterations. Hence, we investigate the following question: How can we design an algorithm for morphology/controller adaptation that is sufficiently data-efficient to be applicable to real-world robots?

Our contribution is two-fold: First, we propose a new optimization algorithm for co-optimization of robot morphology and controller in a data-efficient manner; Second, we validate this approach in simulation and show that it outperforms other state-of-the-art learning methods. Our method exploits the hierarchical relationship between morphology and controller to produce optimal robots despite limited wall time. A sketch of this process is shown in Figure 1. To demonstrate our method, we optimize the design and controller of a recently developed hexapod microrobot [5]. We use a simulation of this microrobot to validate our approach, but our method is not dependent on such a simulation existing. Indeed, the data-efficient nature of our approach allows application to the real-world microrobot. Our approach is – without loss of generality – applicable to other robotic applications that benefit from joint morphology/controller optimization. This in an exciting frontier towards enabling real-world robots that can quickly adapt both morphology and controller to perform specific tasks with high performance.

II. Related Work

A. Bayesian Optimization

Bayesian optimization has been used to optimize both the design of mechanical systems [6], as well as control policies [7], [8], [9], including those for microrobots [10]. Our work differs from these previous works because we jointly optimize hardware design *and* control policies. While control policies are relatively cheap to optimize, new hardware designs incur substantial costs in both money and time. We propose a hierarchical batch contextual Bayesian optimization approach which identifies multiple promising hardware candidates at once (i.e., to be printed on the same silicon wafer) and leverages past evaluations when optimizing the control policy. Our approach yields significant performance improvements compared to standard Bayesian optimization, as demonstrated in Section VI.

B. Morphology Optimization

Our work is related to efforts to optimize controls for robots with non-fixed or reconfigurable morphologies [11], [12], [13], [14], [15], where the controller must handle changes in the robot's physical configuration, either to cope with unanticipated damage [13], [14] or prepare for different tasks [15]. Notably, [16] jointly evolve both morphological and gait parameters on a physical robot able to change its leg lengths dynamically. Application of this method is restricted to robots able to rapidly, repeatedly, and precisely alter their own morphology. This kind of online body modification is impossible for our microrobot because each change requires fabrication and assembly of a new robot.

Other works allow for changes in morphology between iterations, but not dynamic adjustments (i.e., the robot cannot adjust its morphology during simulation or the real-world). [17] include muscle routing in their optimization of gaits of bipedal creatures, allowing variation of muscle attachment points within bounded regions. Although they produce highly natural-looking gaits, it is unclear how muscle routing translates to a robot without an analogous concept of muscles. Furthermore, such aesthetic virtues are not relevant to our robot. The detailed modeling of muscles and ligaments also entails a massive number of parameters to optimize – up to thirty parameters for muscle physiology alone – and a resultant increase in optimization time. In contrast, we only have 3 hardware parameters to optimize.

C. Evolutionary Robotics

Substantial work in evolutionary robotics has focused on jointly evolving morphology and controls of virtual lifeforms [18], [3], [19], [20]. However, evolving body plans from scratch rather than tuning existing designs increases optimization time substantially. Such works also often blur the lines between morphology and control parameters [21], which is disadvantageous when morphological changes are orders of magnitude more costly to evaluate. Our method produces highly performing robots with far fewer morphological changes when compared to evolutionary techniques.

D. Embodied Machines

Both [17] and [22] optimize morphology and controls simultaneously but do not exploit the hierarchical relationship between morphology and control. Such optimization often leads to convergence of morphology before convergence of control [23], [24] and fails to adequately explore the space of morphology parameters. Anecdotally, we observed this in experiments using regular Bayesian optimization, which frequently converged to a poorly performing morphology where only the front four legs touched the ground.

One proposed explanation by [24] suggests that morphology mediates the role of the controller by functioning like an interface to the real world. Simultaneously changing morphology and control parameters is therefore counterproductive because each controller is specialized for some particular morphology. Early convergence of morphology is a consequence of heavy penalization of controller changes, since updating body plans negatively affects the performance of a controller optimized for a different body plan. In this vein, [25] conduct policy search for several hardware schemes to learn optimal control-hardware combinations, running reinforcement learning for each hardware design they explore. This work is most similar to ours because it jointly optimizes morphology and controls, but does not do so simultaneously and thereby avoids the early convergence problem. However, all the morphologies were bio-inspired and designed in advance by hand, whereas we include a large

design space of morphologies in our optimization. Engineering morphologies by hand for our robot is notably harder because of a novel leg design that precludes straightforward transfer of other work [26], [27] in legged locomotion.

III. Problem Formulation

We formulate learning the morphology and controller of our microrobot as the optimization

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) , \tag{1}$$

of the parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}^m, \boldsymbol{\theta}^c]$, where $\boldsymbol{\theta}^m \in \mathbb{R}^n$ denotes the parameters of the morphology and $\boldsymbol{\theta}^c \in \mathbb{R}^m$ the parameters of the controllers, w.r.t. the desired objective function f.

Although this problem can be solved as a single joint optimization task, changing the morphology at each optimization step is extremely costly for our microrobot as each fabrication takes up to a month. Hence, it is crucial to minimize the amount of morphology evaluations. On the other hand, once a morphology is available we can perform hundreds of controller evaluations at little cost. This difference in evaluation cost already suggests that the formulation as a single joint optimization might not be desirable.

An alternative formulation is as a hierarchical optimization task with two independent levels of optimization, morphology optimization on top and controller below, where we alternate between selecting a morphology and optimizing the corresponding controller. This formulation offers a natural way of decoupling the number of evaluations performed on the controller from the evaluations of the morphology. A further improvement on this formulation is to consider the batch nature of the morphology evaluations for our application, selecting multiple morphologies to be manufactured (and later evaluated) at once. One drawback of this hierarchical formulation is that decoupling the two levels of optimization prevents information sharing between them. In practice, this means each controller optimization process cannot make use of information provided by previous controller optimizations, and thus needs to start from scratch.

Our approach, presented in Section V, extends the hierarchical batch formulation and allows to make full use of data collected from previous controller optimizations, thus further improving the data-efficiency.

IV. Background

A. Central Pattern Generators

Central pattern generators (CPGs) are neural circuits commonly found in vertebrates that do not need sensory input to produce periodic outputs [28]. They have been used widely in the design of gaits for robotic locomotion [29], [30], [31]. We chose to use CPGs for our controller. For reasons why CPGs are a good choice of controller for microrobots, we refer readers to [10]. The dynamics of CPGs are modeled as a network of coupled non-linear oscillators. For an in-depth explanation of how the oscillators in CPG networks work, we refer readers to [31].

A major benefit of using CPGs for our controller is the low number of parameters $\boldsymbol{\theta}^c$ to optimize. Usually, the parameters optimized are $\boldsymbol{\theta}^c = [\omega, R, X_l, X_r]$ where ω is the desired frequency of the oscillators, R is the phase difference between each vertical-horizontal oscillator pairs, and X_l and X_r are the amplitudes of the left and right side oscillators and allow for directional control of the microrobot.

B. Bayesian Optimization

One method often used to automate the parameter tuning process is Bayesian optimization (BO). BO is a zero-order black-box optimizer often used for global optimization of expensive functions [32], [33]. At every iteration of the optimization, BO learns a model $\tilde{f} : \boldsymbol{\theta} \rightarrow f(\boldsymbol{\theta})$ from the dataset of previously evaluated parameters and their returned objective values $\mathcal{D} = \{\boldsymbol{\theta}, f(\boldsymbol{\theta})\}$. The learned model is then used to execute a virtual optimization by using an acquisition function which controls the trade-off between exploitation and exploration. The returned parameters $\boldsymbol{\theta}^*$ from this optimized model are then evaluated on the real system to obtain an objective value $f(\boldsymbol{\theta}^*)$. Finally, the parameters evaluated $\boldsymbol{\theta}^*$ and the corresponding objective value obtained from the real system $f(\boldsymbol{\theta}^*)$ are added to the dataset, and a new iteration of the optimization begins. The choice of model is important for BO to learn the underlying objective. One commonly used model, and the one which we use in this paper, is the Gaussian process (GP) model [34]. For a more in-depth background on BO, we direct readers to [33], [32], [35].

An extension of standard BO we use as a subcomponent is *contextual Bayesian optimization* (cBO) [36]. cBO extends the standard BO framework by augmenting the optimization problem with an additional context parameter $\boldsymbol{c}$, and learns a joint policy $\tilde{f} : \{\boldsymbol{\theta}, \boldsymbol{c}\} \rightarrow f(\boldsymbol{\theta})$, where $\boldsymbol{c}$ is fixed during optimization (i.e., it is observable, but not controllable). In our approach, we use cBO to optimize the controller, as described in Section V. By encoding the morphologies as contexts, cBO takes advantage of the similarities between different morphologies and generalizes to good polices for unseen designs faster.

Within our approach, we also use another variant of BO called *batch Bayesian optimization* (BBO) [37] to optimize the morphology. In contrast to a fully sequential algorithm, which alternates between choosing individual points and evaluating them on the true reward function, BBO queries the acquisition function for multiple points, then evaluates them in parallel before selecting another. The first set of parameters of each batch is selected as in a sequential policy and the next set chosen with an acquisition function. However, rather than immediately evaluate the returned parameters on the real system, BBO defers evaluation of the reward function on this point until the entire batch is selected, temporarily substituting for its reward a prediction $\mathcal{H}$ made by the GP (also called *hallucinated observations* [38] or *fantasies* [35]). The GP model is updated with the data point $\mathcal{D} = \{\boldsymbol{\theta}, \mathcal{H}\}$ of returned parameters and respective hallucinated observation and is then used to select the next point. Once the entire batch is selected, all points are evaluated and the hallucinated observations are replaced by real ones. Although hallucinated

Algorithm 1 Hierarchical Process Constrained Batch Bayesian Optimization (HPC-BBO) – contextual case

$\boldsymbol{\theta}^m_{1:K} \leftarrow [\boldsymbol{\theta}^m_1, ..., \boldsymbol{\theta}^m_K]$ ▷ Randomly initialize batch
while $b < NumBatches$ **do**
  for $k = 1, .., K$ **do** ▷ Learn controllers
    while $i < NumIters$ **do**
      $\boldsymbol{\theta}^{c*} \leftarrow \arg\max_{\boldsymbol{\theta}^c}$ GP-UCB$(\boldsymbol{\theta}^c, \boldsymbol{\theta}^m_k | \mathcal{G}_c)$
      $f(\boldsymbol{\theta}^{c*}, \boldsymbol{\theta}^m_k) \leftarrow \boldsymbol{\theta}^{c*}$ ▷ evaluate on real system
      $\mathcal{G}_c \leftarrow \{\boldsymbol{\theta}^{c*}, f(\boldsymbol{\theta}^{c*}, \boldsymbol{\theta}^m_k)\}$
    ${\mathcal{R}^*}_k \leftarrow f(\boldsymbol{\theta}^{c*}, \boldsymbol{\theta}^m_k)$ ▷ Keep best reward
    $\mathcal{G}_m \leftarrow \{\boldsymbol{\theta}^m_k, {\mathcal{R}^*}_k\}$
  ${\mathcal{G}'}_m \leftarrow \mathcal{G}_m$ ▷ Temporary model with hallucinations
  for $k = 1, .., K$ **do** ▷ Generate new morphologies
    $\boldsymbol{\theta}^{m*} \leftarrow \arg\max_{\boldsymbol{\theta}^m}$ GP-UCB$(\boldsymbol{\theta}^m | {\mathcal{G}'}_m)$
    $\boldsymbol{\theta}^m_k \leftarrow \boldsymbol{\theta}^{m*}$ ▷ Add to batch
    $\mathcal{H}^* \leftarrow h(\boldsymbol{\theta}^m_k | \mathcal{G}_m)$ ▷ Hallucinate a reward
    ${\mathcal{G}'}_m \leftarrow \{\boldsymbol{\theta}^{m*}_k, \mathcal{H}^*\}$

rewards are less informative, batching saves time since points in the batch can be evaluated in parallel. The particular implementation of BBO we use as a subcomponent in our approach is PC-BBO [37].

V. Hierarchical Process Constrained Batch Bayesian Optimization

Our algorithm is called *hierarchical process constrained batch Bayesian optimization* (HPC-BBO). In this context, "process-constrained" refers not to classical constrained optimization, but rather to a physical limitation that restricts how frequently a particular parameter can be changed. In our case, it would be relatively straightforward to change control parameters (unconstrained) on a physical microrobot, compared to fabricating a new hexapod to test a different morphology (constrained).

We now detail Algorithm 1. Given a set of morphology parameters $\boldsymbol{\theta}^m$ and controller parameters $\boldsymbol{\theta}^c$, we want to jointly optimize them. We use BBO to select the morphology parameters $\boldsymbol{\theta}^m$ and evaluate them by having a nested optimization procedure for the controller parameters $\boldsymbol{\theta}^c$. At the end of the controller optimization, we return to the morphology optimizer the best reward obtained for that specific morphology. We initialize a batch of size K by picking morphology parameters with random search. In the contextual case, for each of step $1, \ldots, K$ of the current batch, we set the morphology parameters as a context, and perform contextual Bayesian optimization over the controller parameters using a GP model $\mathcal{G}_c$ that learns a policy $\tilde{f} : \{\boldsymbol{\theta}^c, \boldsymbol{c}\} \rightarrow f(\boldsymbol{\theta}^c)$, where $\boldsymbol{c} = \boldsymbol{\theta}^m_k$. The noncontextual variant of our algorithm uses standard BO instead of cBO to optimize the controller, which necessitates training a new GP for each morphology in the batch for a total of K controller GPs per batch. Both the contextual GP and noncontextual GPs must first be initialized with randomly chosen controller parameters. After the first batch is evaluated, another GP model $\mathcal{G}_m$ is used to learn a policy $\tilde{f} : \boldsymbol{\theta}^m \rightarrow f(\boldsymbol{\theta}^c)$ and

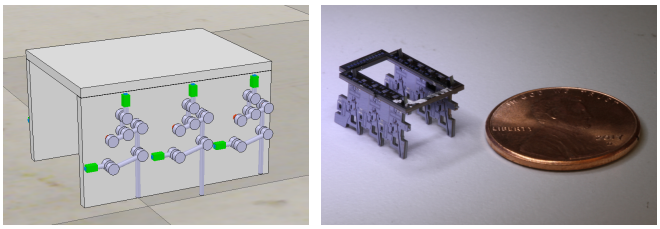

Fig. 2: The simulated (*left*) and real microrobot walker (*right*) considered in our work.

updated with the batch and corresponding best rewards for each batch element evaluated from the controller optimization. We query this updated model $\mathcal{G}_m$ using an acquisition function (GP-UCB in our case) for another batch of morphology parameters, this time with updated knowledge of what parameters generated the best rewards from the software optimization, and evaluate this batch as before. Regardless of whether noncontextual HPC-BBO or contextual HPC-BBO is employed, the use of BBO to optimize morphology allows all controller optimizations for a batch to be done in parallel. By evaluating multiple morphologies in one batch, we drastically reduce wall time for optimization since we can evaluate K morphologies in one production cycle, whereas regular BO evaluates one. In essence, contextual HPC-BBO leverages the data efficiency of BBO to optimize the expensive constrained parameters while also taking advantage of the information learned across different contexts with cBO to optimize the unconstrained parameters. Using our approach, we can co-optimize robot control and morphology in a much more data-efficient manner, allowing us to evaluate more microrobots per fabrication cycle.

VI. Experimental Results

A. Experimental Setting

In our experiments, we used the robotic simulator V-REP to model a hexapod microrobot similar to the one described in [10]. Each of the six legs is driven by 2 motors: one motor actuates the leg vertically, and the other motor moves the leg back and forth – resulting in the legs having a circular-like sweep. The simulated microrobot is scaled 100 times larger than its physical analogue since V-REP is unable to handle dynamics at the micrometer scale. It is important to notice that our algorithm does not require a simulator to work, and that the simulator is here used only to evaluate the performance of the robots, similar to what would happen in the real world.

For the controller, we optimize six CPG parameters, which correspond to the frequency, amplitude, and offsets of the vertical and horizontal motors. In addition, we separately consider parameters that control the amplitude of leg swings on the left and right sets of legs respectively. Although most of the parameters are related to our controller of choice, the CPG, our method is agnostic to the type of controller. The three morphology parameters control the lengths of pairs of legs (front, middle, and rear) and are encoded as ratios relative to a normalized leg length. We use a batch size $K =$

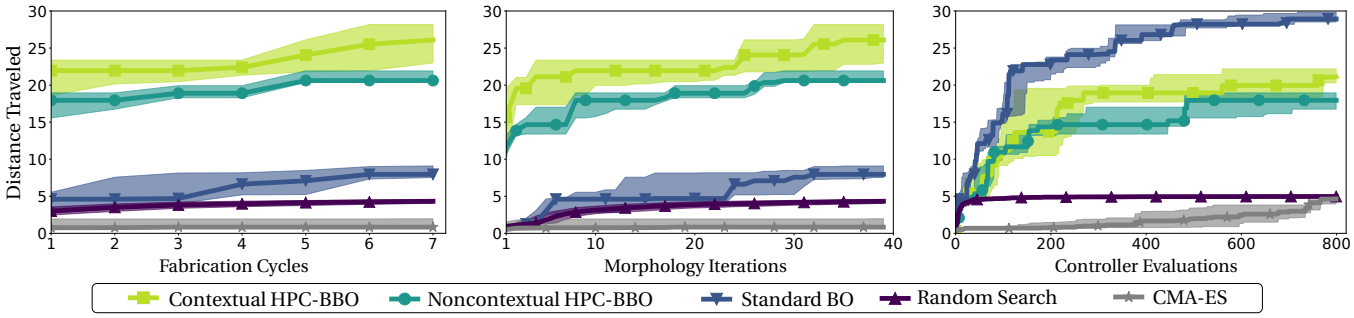

Fig. 3: Learning curves (median and 65th percentile of the best performance observed so far) for multiple algorithms w.r.t. (*left*) number of manufacturing batches, (*center*) number of morphologies evaluated, and (*right*) number of experiments performed. Each fabrication cycle corresponds to a wafer of 5 morphologies. Although standard BO performs better than our approach w.r.t. the pure number of evaluations, when considering the number of fabrication cycles and number of morphologies evaluated, our approach significantly outperforms all other baselines, with a 267% improvement over standard BO for the non-contextual version and a 360% improvement over standard BO for the contextual version. In the real fabrication process of the microrobot, where the fabrication cycle of a wafer takes about a month, this would correspond to 4 months for our approach and 21 months for standard BO to reach the same performance.

5 and run the controller optimizer for 50 iterations for each morphology evaluation. Videos and code for reproducing the experiments are available at `https://sites.google.com/view/learning-robot-morphology`

B. Comparison to Other Methods

We now compare the two variants of our approach (contextual HPC-BBO and non-contextual HPC-BBO) against three baselines: random search, covariance matrix adaptation evolutionary strategy (CMA-ES), and standard BO. Random search [39] samples the parameter space as a uniform distribution, establishing a baseline for our optimization task. Covariance matrix adaptation evolutionary strategy is a gradient-free algorithm to optimize non-convex functions [40]. Standard BO optimizes all parameters at once and, unlike our hierarchical approach, does not batch hardware evaluations.

Figure 3 shows the learning curves of the various methods w.r.t. different optimization desiderata: number of fabrication cycles, morphology iterations, and controller evaluations. The number of fabrication cycles is number of times a new silicon wafer has to be fabricated and is the most important statistic for comparison because it directly correlates to wall time. By generating a batch of multiple morphologies, HPC-BBO examines many different sets of hardware parameters at each fabrication cycle, whereas CMA-ES and standard BO only evaluate one new morphology every cycle. The fabrication process of each batch of morphologies takes 4-6 weeks in the real world. As a result, standard BO would need 21 months to reach the same performance that our approach would reach in 4 months (assuming that the convergence rate in simulation would translate to real world). HPC-BBO also outperforms standard BO and CMA-ES w.r.t. the number of morphology iterations. This metric is relevant because it shows that even if the batch size $K = 1$, HPC-BBO would still outperform standard BO (since the larger batch sizes decrease performance, as explained in Section IV).

Morphology	**1**	**2**	**3**	**4**
Controller 1	–	-18.73%	-7.35%	-46.29%
Controller 2	-74.79%	–	-78.38%	-29.34%
Controller 3	-75.52%	-13.26%	–	-56.19%
Controller 4	-88.43%	-65.83%	-89.04%	–

TABLE I: Changes in performance for a morphology/controller pair as a percentage of the reward, when changing the controller for a given morphology. The significant decreases in performance show that our hierarchical approach optimizes the best controller for each robot morphology instead of attempting to find a controller which works well for all morphologies.

HPC-BBO is significantly outperformed by standard BO when considering the number of controller evaluations, because standard BO is able to see far more morphologies as it is able to change both controller and morphology parameters at the same time. However, since even just 200 controller evaluations would take a decade and a half, comparing HPC-BBO to standard BO on this basis is misleading.

The importance of our hierarchical technique is highlighted by the result presented in Table I, which shows that the gaits we learn are specific to different morphologies. We take the controllers and morphologies of the best four performing robots across all experiments and show that recombining them produces suboptimal pairings. Highly performing morphologies can perform up to 90% worse when paired with a controller optimized for another morphology, even when the other morphology-controller pair also performs well. Importantly, this relationship is not symmetric: even though the controller optimized for morphology 1 only decreases the performance of morphology 3 by 8%, the controller for morphology 3 decreases the performance of morphology 1 by 75%.

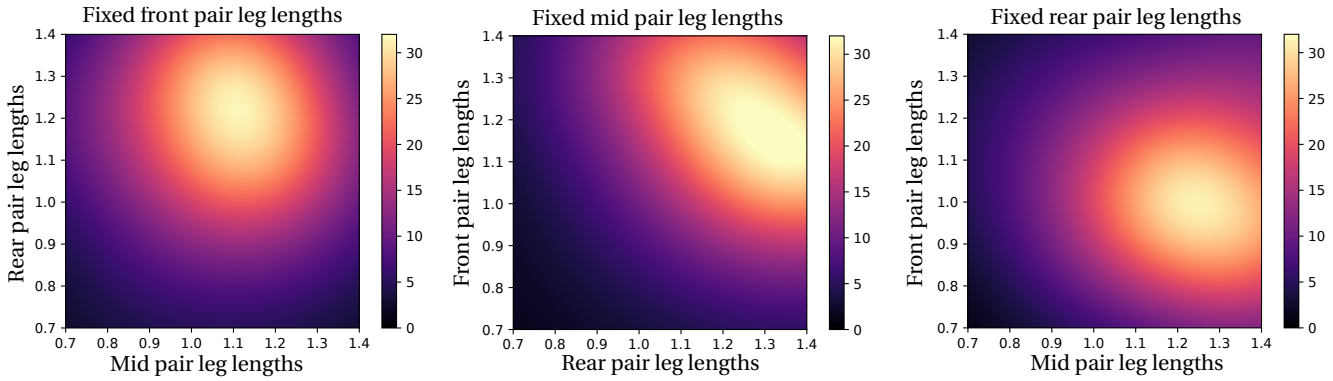

Fig. 4: Visualization of the learned GP model for the hardware parameter space from contextual HPC-BBO. The colormap indicates the distance walked, where 30 is far and 0 is stationary. We see from the left map that fixing the length of the front pair of legs to be equal to that of a specific high performing morphology, it is best paired with medium length middle legs and long rear legs. When we fix the middle leg as seen in the center map, it is best paired with long rear legs and long front legs. The right map indicates that when we fix the rear leg, it is best paired with long middle legs and short front legs.

Even the controller that least decreases the performance of morphologies it was not optimized for, still decreases the performance of the morphology which has the least change in performance with different controllers. This means that even if standard BO or CMA-ES were to find a controller that performs well across most morphologies, they are not guaranteed to find the optimal controller/morphology pair.

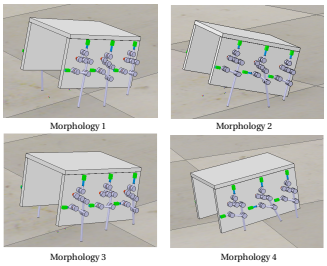

Fig. 5: The high-performing morphologies from Table I. While morphology 4 has all short legs, the other morphologies have varied lengths and a tilted stance. This may explain why other controllers perform so poorly when paired with morphology 4.

From Figure 5, we can see how the tilted stances of the morphologies obtained with HPC-BBO resemble the inclined poses adopted by 6-legged insects in nature [41].

C. Contextual vs Non-contextual HPC-BBO

We evaluate both the contextual and noncontextual variants of HPC-BBO. Contextual HPC-BBO consistently outperforms noncontextual HPC-BBO. As seen in Figure 3, the performance gap in distance traveled is largest at the beginning and towards the end w.r.t. the number of fabrication cycles and morphology iterations. Contextual HPC-BBO outperforms non-contextual HPC-BBO at the beginning of optimization because it is able to efficiently use data accumulated in previous morphology contexts to produce competitive gaits in unseen morphology contexts. This gap becomes less evident when the number of controller evaluations increases, as seen in the rightmost graph of Figure 3. As non-contextual HPC-BBO trains on more morphologies, it picks better ones, which lessens the importance of generalizing across different contexts. This is because a higher number of controller evaluation iterations devoted to optimizing the software parameters allows the non-contextual optimizer to catch up to the contextual optimizer. With more iterations to optimize software parameters, the information of the contextual information from previous iterations is washed out, reducing the advantage the contextual optimizer has with fewer iterations. The practical implication is that a shorter period of time to optimize software favors the contextual approach, whereas a longer one reduces its advantage.

VII. Conclusions

In this paper, we studied how to automatically optimize the design of robot morphologies and controllers in a data-efficient manner. To achieve this goal, we introduced a novel algorithm, hierarchical process constrained batch Bayesian optimization (HPC-BBO), and validated our approach in simulation. Results on a simulated hexapod microrobot show that HPC-BBO significantly outperforms all other baselines and other state-of-the-art learning methods, with a performance improvement of 360% over standard Bayesian optimization. By exploiting the hierarchical relationship between morphology and controller, we demonstrate that HPC-BBO can produce high-performing morphologies/controllers in a data-efficient manner. Moreover, HPC-BBO can exploit the simultaneous fabrication of multiple robot morphologies. As a result, HPC-BBO achieve the same performance of standard BO in a fifth of the time (4 months compared to 21 months).

The proposed approach is a first step towards the grand goal of allowing robots that can not only quickly learn suitable controllers from experience, but also to adapt their hardware based on the needs dictated by their environment and goals.

An exciting future direction is to "open the black-box" by replacing Bayesian optimization with model-based reinforcement learning [42], to allow for more complex controllers. Additionally, we aim to apply the proposed approach to the design of real-world micro-robots.

REFERENCES

[1] J.-B. Lamarck, *Philosophie Zoologique*. Paris: Dentu et L'Auteur, 1809.

[2] E. V. Koonin and Y. I. Wolf, "Is evolution Darwinian or/and Lamarckian?" *Biology direct*, vol. 4, no. 1, p. 42, 2009.

[3] H. Lipson and J. B. Pollack, "Automatic design and manufacture of robotic lifeforms," *Nature*, vol. 406, no. 6799, p. 974, 2000.

[4] M. Jelisavcic, R. Kiesel, K. Glette, E. Haasdijk, and A. E. Eiben, "Analysis of lamarckian evolution in morphologically evolving robots," *Proceedings of the European Conference on Artificial Life*, pp. 214–221, 2017.

[5] D. S. Contreras, D. S. Drew, and K. S. Pister, "First steps of a millimeter-scale walking silicon robot," in *International Conference on Solid-State Sensors, Actuators and Microsystems (TRANSDUCERS)*. IEEE, 2017, pp. 910–913.

[6] R. Martinez-Cantin, "Funneled Bayesian optimization for design, tuning and control of autonomous systems," *IEEE Transactions on Cybernetics*, 2018.

[7] D. J. Lizotte, T. Wang, M. Bowling, and D. Schuurmans, "Automatic gait optimization with Gaussian process regression," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007, pp. 944–949.

[8] M. Tesch, J. Schneider, and H. Choset, "Using response surfaces and expected improvement to optimize snake robot gait parameters," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 1069–1074.

[9] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian optimization for learning gaits under uncertainty," *Annals of Mathematics and Artificial Intelligence (AMAI)*, vol. 76, no. 1, pp. 5—23, 2015.

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

[11] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, 2007.

[12] J. C. Bongard, "Evolutionary robotics," *Communications of the ACM*, vol. 56, no. 8, pp. 74–83, 2013.

[13] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, p. 503, 2015.

[14] J. Bongard, V. Zykov, and H. Lipson, "Resilient machines through continuous self-modeling," *Science*, vol. 314, no. 5802, pp. 1118–1121, 2006.

[15] J. W. Romanishin, K. Gilpin, and D. Rus, "M-blocks: Momentum-driven, magnetic modular robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 4288–4295.

[16] T. F. Nygaard, C. P. Martin, E. Samuelsen, J. Torresen, and K. Glette, "Real-world evolution adapts robot morphology and control to hardware limitations," *arXiv preprint arXiv:1805.03388*, 2018.

[17] T. Geijtenbeek, M. van de Panne, and A. F. van der Stappen, "Flexible muscle-based locomotion for bipedal creatures," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 206:1–206:11, Nov. 2013.

[18] J. C. Bongard and R. Pfeifer, "Evolving complete agents using artificial ontogeny," in *Morpho-functional Machines: The new species*. Springer, 2003, pp. 237–258.

[19] P. Funes and J. Pollack, "Evolutionary body building: Adaptive physical designs for robots," *Artificial Life*, vol. 4, no. 4, pp. 337–357, 1998.

[20] K. Sims, "Evolving virtual creatures," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM, 1994, pp. 15–22.

[21] N. Cheney, J. Clune, and H. Lipson, "Evolved electrophysiological soft robots," in *ALIFE*, vol. 14, 2014, pp. 222–229.

[22] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane, "Joint optimization of robot design and motion parameters using the implicit function theorem," in *Robotics: Science and Systems*, 2017.

[23] M. Joachimczak, R. Suzuki, and T. Arita, "Artificial metamorphosis: Evolutionary design of transforming, soft-bodied robots," *Artificial life*, vol. 22, no. 3, pp. 271–298, 2016.

[24] H. Lipson, V. Sunspiral, J. Bongard, and N. Cheney, "On the difficulty of co-optimizing morphology and control in evolved virtual creatures," in *Proceedings of the European Conference on Artificial Life 13*. MIT Press, 2016, pp. 226–233.

[25] K. S. Luck, J. Campbell, M. A. Jansen, D. M. Aukes, and H. B. Amor, "From the lab to the desert: fast prototyping and learning of robot locomotion," *Robotics: Science and Systems (RSS)*, 2017.

[26] A. M. Hoover, E. Steltz, and R. S. Fearing, "Roach: An autonomous 2.4 g crawling hexapod robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 26–33.

[27] K. Wampler and Z. Popović, "Optimal gait and form for animal locomotion," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 60:1–60:8, July 2009.

[28] J. Yu, M. Tan, J. Chen, and J. Zhang, "A survey on cpg-inspired control models and system implementation," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 3, pp. 441–456, 2014.

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

[30] L. Righetti and A. Ijspeert, "Design methodologies for central pattern generators: an application to crawling humanoids," in *Robotics: Science and Systems (RSS)*, 2006, pp. 191–198.

[31] A. Crespi, D. Lachat, A. Pasquier, and A. J. Ijspeert, "Controlling swimming and crawling in a fish robot using a central pattern generator," *Autonomous Robots*, vol. 25, no. 1-2, pp. 3–13, 2008.

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

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

[34] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced lectures on machine learning*. Springer, 2004, pp. 63–71.

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

[36] J. H. Metzen, A. Fabisch, and J. Hansen, "Bayesian optimization for contextual policy search," in *IROS Workshop on Machine Learning in Planning and Control of Robot Motion*, 2015.

[37] P. Vellanki, S. Rana, S. Gupta, D. Rubin, A. Sutti, T. Dorin, M. Height, P. Sanders, and S. Venkatesh, "Process-constrained batch bayesian optimisation," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 3414–3423.

[38] T. Desautels, A. Krause, and J. W. Burdick, "Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3873–3923, 2014.

[39] S. H. Brooks, "A discussion of random methods for seeking maxima," *Operations Research*, vol. 6, no. 2, pp. 244–251, 1958.

[40] N. Hansen, "The cma evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016.

[41] L. M. Theunissen, H. H. Bekemeier, and V. Dürr, "Comparative whole-body kinematics of closely related insect species with different body morphology," *Journal of Experimental Biology*, vol. 218, no. 3, pp. 340–352, 2015.

[42] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *Advances in Neural Information Processing Systems (NIPS)*, 2018.