2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Learning Dynamic Tasks on a Large-scale Soft Robot in a Handful of Trials

Sicelukwanda Zwane[1], Daniel Cheney[2], Curtis C. Johnson[2], Yicheng Luo[1], Yasemin Bekiroglu[1,3], Marc D. Killpack[2], Marc Peter Deisenroth[1]

2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3503-7770-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/IROS58592.2024.10802122

***Abstract*— Soft robots offer more flexibility, compliance, and adaptability than traditional rigid robots. They are also typically lighter and cheaper to manufacture. However, their use in real-world applications is limited due to modeling challenges and difficulties in integrating effective proprioceptive sensors. Large-scale soft robots ($\approx$ two meters in length) have greater modeling complexity due to increased inertia and related effects of gravity. Common efforts to ease these modeling difficulties such as assuming simple kinematic and dynamics models also limit the general capabilities of soft robots and are not applicable in tasks requiring fast, dynamic motion like throwing and hammering. To overcome these challenges, we propose a data-efficient Bayesian optimization-based approach for learning control policies for dynamic tasks on a large-scale soft robot. Our approach optimizes the task objective function directly from commanded pressures, without requiring approximate kinematics or dynamics as an intermediate step. We demonstrate the effectiveness of our approach through both simulated and real-world experiments.**

I. Introduction

Elephant trunks, snakes, and certain invertebrates are capable of highly dynamic and fast motion while maintaining their flexibility, and compliance. However, despite being modeled after these biological entities, soft robots still struggle to perform highly dynamic control tasks. This is because it is difficult to derive accurate kinematic or dynamics models that describe their motion well enough to allow for the design of controllers that fully exploit their capabilities.

Large-scale soft robots (a few meters in length) are attractive because of their high force-to-weight ratio [1]. However, their increased inertia, surface area, and related gravitational effects also present further modeling difficulties.

Previous innovations in soft robot modeling, such as constant curvature [2], PDEs [3], finite elements [4], first-principles [5], splines [6], and deep learning [7] have allowed soft robots to perform well enough at static control tasks. However, they generally require the robot to move slower to maintain lower tracking error, and fail when the robot is carrying non-uniform loads or operating at higher velocities.

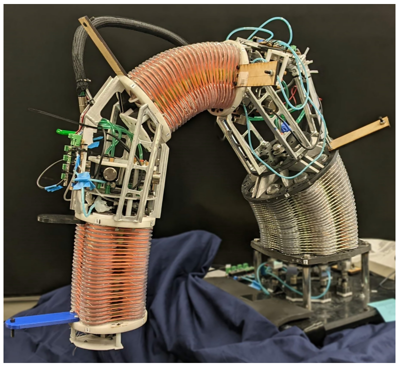


Fig. 1: The soft robot design used in this paper for hardware experiments. This is a 12 DOF pneumatically actuated, large-scale soft robot about 1.3 meters long. It consists of three continuum joints connected by two rigid links.

Modeling approaches with high data requirements (i.e., deep neural networks), are less suitable for soft robots as the data collection process may cause "wear and tear" on the "soft" parts of the robot. This may affect the robot's dynamics. Other effects such as temperature may also affect the robot's dynamics or contribute to measurement noise.

Given these modeling challenges, we adopt a *task-oriented*, Bayesian optimization-based control learning approach that does not require an explicit model of the soft robot. Our approach improves the efficiency of building a controller by significantly reducing the number of necessary hardware evaluations on the robot. Unlike existing soft robot control methods, our approach makes fewer platform-specific assumptions, making it easier to adapt for use on morphologically different soft robots.

Our contributions can be summarized as follows:

1) We present a Bayesian optimization-based approach to soft robot control tuning, capable of learning dynamic, high-dimensional behaviors directly from low-dimensional control parameterizations.
2) We learn successful control policies with remarkably few interactions with the soft robot, minimizing possible wear and tear on the robot hardware.
3) We evaluate on two challenging tasks: throwing and hammering, which require highly dynamic motions and complex velocity profiles for successful completion.
4) We demonstrate our approach on a real large-scale soft robot (shown in Figure 1), successfully learning con-

[1]UCL Centre for Artificial Intelligence, University College London, UK. Corresponding author email: sicelukwanda.zwane.20@ucl.ac.uk

[2] Department of Mechanical Engineering, Brigham Young University, Provo, Utah, UT 84602, USA

[3]Department of Electrical Engineering, Chalmers University of Technology, Göteborg SE-41296, Sweden

This work was partially supported by the National Science Foundation under Grant No. 1935312 and the Engineering and Physical Sciences Research Council (EPSRC) [EP/S021566/1]. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission. The codebase associated with this paper is available at https://github.com/Sicelukwanda/BayesOptSoftRobotControl

trol parameters for a "pseudo-throwing" task despite uncertain objective function observations.

II. Hardware Description

In this work, we consider the large-scale soft robot shown in Figure 1. It consists of three continuum joints connected together using rigid links. Each continuum joint is constructed with four plastic pressure chambers, which are individually pressure controlled. These chambers are arranged in an antagonistic pattern, such that each pair of pressure chambers causes a net bending torque along the longitudinal section of the robot. Each joint's bending stiffness is related to the material properties of the plastic pressure chambers. Although the robot is designed to resist elongation and twisting, these effects can still occur under high velocity or non-uniform loading conditions, introducing additional sources of uncertainty.

We control the robot by issuing a vector of 12 pressure commands (4 commands for each continuum joint).

III. Robot Control using Bayesian Optimization

In this section, we detail our approach for finding good policy parameters using Bayesian Optimization (BayesOpt).

A. Setting

We consider finite-episodic tasks, where the environment (robot and task objects) can be "reset" to the same starting conditions. To avoid having to find a state description and the need for a kinematic model for the large-scale soft robot, we consider an open-loop policy $\pi(\boldsymbol{\theta})$ with parameters $\boldsymbol{\theta} \in \mathbb{R}^D$. For each trial, the policy is expected to produce a sequence of controls $\boldsymbol{u} \in \mathbb{R}^U$ over the task horizon H. At the end of the trial, after executing all control commands $\boldsymbol{u}_{1:H} = \pi(\boldsymbol{\theta})$, we receive a single observation from a task utility function

$$J(\boldsymbol{\theta}) = \mathbb{E}\left[R(\boldsymbol{u}_{1:H}) \mid \boldsymbol{\theta}\right], \tag{1}$$

where $R(\cdot)$ may be any value indicating the performance over the entire action sequence $\boldsymbol{u}_{1:H}$. For example, in a throwing task, we may use a tape measure to record the final displacement of a projectile after sequentially executing individual control actions $\boldsymbol{u}_t$ on the soft robot.

B. Problem Formulation

The search for optimal policy parameters $\boldsymbol{\theta}^*$ with an unknown objective function $J(\boldsymbol{\theta})$ can be formulated as a black-box optimization problem

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}\in\mathbb{R}^D} J(\boldsymbol{\theta}). \tag{2}$$

Because J is unknown, we have to adopt a data-driven strategy to find $\boldsymbol{\theta}^*$. Moreover, this strategy has to be data-efficient since performing a large number of evaluations on the soft robot may cause changes to its dynamics. Although measuring such changes is not an easy task, we can directly observe that even the equilibrium configuration changes over the period of minutes if the robot is left in a specific configuration due to material properties. This variation in system dynamics and the need to minimize the number of hardware evaluations makes Bayesian optimization an effective approach to address these problems.

C. Bayesian Optimization based Control

BayesOpt solves problems of the form Equation (2) by learning a surrogate model or response surface of the objective function from data. Next, it optimizes an acquisition function $\alpha(\cdot)$ over the input space to find the next objective function input to evaluate. This input and resulting objective function observation are added to the dataset which is used to improve the surrogate model and the process is repeated.

In our robot control context, we apply this approach according to Algorithm 1, with a dataset consisting of the policy parameters and corresponding objective function observations $\mathcal{D} = \{\boldsymbol{\theta}_i, J(\boldsymbol{\theta}_i)\}_{i=1}^N$.

Algorithm 1 Bayesian Optimization for Robot Control

Input: Task objective function J, domain $\mathbb{R}^D$, initial observations $\mathcal{D}_0 = \{(\boldsymbol{\theta}_i, J(\boldsymbol{\theta}_i))\}_{i=1}^{N_{init}}$, where $\boldsymbol{\theta}_i \in \mathbb{R}^D$ and an acquisition function $\alpha(\boldsymbol{\theta})$.

```
Fit a GP on the initial dataset $\mathcal{D}_0$
for $n = 1, 2, \ldots, N$ do
    Select $\boldsymbol{\theta}_n$ by optimizing the acquisition function:
        $\boldsymbol{\theta}_n = \arg\max_{\boldsymbol{\theta}\in\mathbb{R}^D} \alpha(\boldsymbol{\theta} \mid \mathcal{D}_{n-1})$
    Get action sequence from the policy $\boldsymbol{u}_{1:H} = \pi(\boldsymbol{\theta}_n)$
    Execute action sequence $\boldsymbol{u}_{1:H}$ on the soft robot
    Record task performance $J(\boldsymbol{\theta}_n)$ for $\boldsymbol{u}_{1:H}$
    Update the dataset $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{(\boldsymbol{\theta}_n, J(\boldsymbol{\theta}_n))\}$
    Update the GP model with $\mathcal{D}_n$
end for
```

Output: The control parameters $\boldsymbol{\theta}^* \in \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N\}$ with the maximum objective function value $J(\boldsymbol{\theta})$.

The performance of BayesOpt greatly relies on the chosen surrogate model and acquisition function. We briefly explain these concepts in the following subsections and motivate our specific choices in this work.

1) Gaussian process Surrogate Model: For the dynamic, high-inertia tasks we consider in this work, we expect measurement noise in the objective function evaluations, i.e., the same set of policy parameters $\boldsymbol{\theta}$ may not yield exactly the same $J(\boldsymbol{\theta})$. This is especially true in the case of experiments on the physical soft robot. Due to this measurement noise, the optimum of Equation (1) may not exactly coincide with the optimum of the true objective, which corresponds to the best possible task-solving behavior [8]. This motivates the use of a probabilistic surrogate model, which can cope well with noisy observations. For this reason, we use a Gaussian process surrogate model.

A Gaussian process (GP) [9] has the capacity to learn an unknown non-linear function and express it as a posterior distribution over all possible functions under a given prior distribution and training dataset. Given a set of query control parameters $\boldsymbol{\theta}_*$, we can obtain the mean and covariance of the

GP predictive posterior in closed form as follows:

$$\mu(\boldsymbol{\theta}_*) = \boldsymbol{K}_*^T \left(\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y} \tag{3}$$

$$\mathrm{cov}(\boldsymbol{\theta}_*) = \boldsymbol{K}_{**} - \boldsymbol{K}_*^T \left(\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I}\right)^{-1} \boldsymbol{K}_*, \tag{4}$$

where $\boldsymbol{K}_{**} = k(\boldsymbol{\theta}_*, \boldsymbol{\theta}_*)$, $\boldsymbol{K}_* = k(\boldsymbol{\Theta}, \boldsymbol{\theta}_*)$, and $\boldsymbol{K} = k(\boldsymbol{\Theta}, \boldsymbol{\Theta})$ are matrices constructed using the GP kernel function $k(\cdot,\cdot)$ and training data $\boldsymbol{\Theta} = \begin{bmatrix}\boldsymbol{\theta}_1 & \cdots & \boldsymbol{\theta}_N\end{bmatrix}$, $\boldsymbol{y} = \begin{bmatrix}J(\boldsymbol{\theta}_1) & \cdots & J(\boldsymbol{\theta}_N)\end{bmatrix}$. $\boldsymbol{I}$ is the identity matrix, and σ_ε^2 is the measurement noise variance in the objective function observations $\boldsymbol{y}$.

The kernel function choice is particularly important in BayesOpt because it is directly responsible for optimization-relevant properties, such as the number and frequency of local minima, differentiability (smoothness), and convexity of the learned objective function. In this work, following the recommendations of [10], we use the automatic relevance determination (ARD) Matern52 kernel

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sigma_f^2 \left(1 + \sqrt{5d^2} + \frac{5d^2}{3}\right) \exp\left(-\sqrt{5d^2}\right), \tag{5}$$

where σ_f^2 is the signal variance and

$$d^2 = \sum_{i=1}^{D} \frac{(\theta_i - \theta_i')^2}{l_i^2}.$$

We train the GP on the latest set of observations $\mathcal{D}_n$ by maximizing the log marginal likelihood

$$\begin{aligned}\log p(\boldsymbol{y} \mid \{\boldsymbol{\theta}_n\}_{n=1}^N) = &-\frac{1}{2}\boldsymbol{y}^T \left(\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y} \\ &- \frac{1}{2}\log|\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I}| - \frac{N}{2}\log 2\pi\end{aligned} \tag{6}$$

during the GP update step (Line 8 of Algorithm 1). We use gradient-based (L-BFGS-B) multi-start optimization to optimize over the full set of GP parameters consisting of the likelihood variance σ_ε^2 and the Matern52 kernel hyperparameters which include the lengthscales $\boldsymbol{\Lambda} = [l_1, l_2, \ldots, l_D]$ and signal variance σ_f^2, training a new GP each time the dataset $\mathcal{D}$ is updated. In all BayesOpt experiments, we use a minimal initial dataset size of $D+1$, where D is the number of dimensions in $\boldsymbol{\theta}$.

2) Acquisition Functions: The acquisition function balances the trade-off between exploring input regions where there is high uncertainty about the objective function and "exploiting" input regions where the surrogate model predicts high objective function values. This balance is crucial to efficiently find the global optimum of the objective function with as few evaluations as possible. We consider two common acquisition functions, namely the Upper Confidence Bound (UCB)

$$\alpha(\boldsymbol{\theta}) = \mu(\boldsymbol{\theta}) + \sqrt{\kappa}\sigma(\boldsymbol{\theta}) \tag{7}$$

and the Expected Improvement (EI)

$$\alpha(\boldsymbol{\theta}) = \sigma(\boldsymbol{\theta})\left[v\Phi(v) + \phi(v)\right]; \qquad v = \frac{F - \mu(\boldsymbol{\theta})}{\sigma(\boldsymbol{\theta})}, \tag{8}$$

where F is the best objective function value so far. $\Phi(\cdot)$ and $\phi(\cdot)$ are the cumulative density function and probability density function of the normal distribution, respectively. UCB (Equation (7)) is the function representing some scalar (i.e., κ) number of standard deviations $\sigma(\boldsymbol{\theta})$ above the mean function $\mu(\boldsymbol{\theta})$ of the GP predictive posterior (Equation (3)) and $\sigma(\boldsymbol{\theta})$ is the corresponding standard deviation derived from predictive posterior at $\boldsymbol{\theta}$ (Equation (4)). As the name suggests, EI takes an expectation of improvements (positive changes in function value) over multiple surrogate model posterior samples at a given parameter configuration $\boldsymbol{\theta}$, for surrogate models with Gaussian likelihoods, this simplifies to the form presented in Equation 8. In this work, we employ the Log Expected Improvement (LEI) [11], an EI variant particularly well-suited for high-dimensional problems.

D. Few-parameter Control Policy

To perform policy evaluation (Line 4 of Algorithm 1), we require an efficient mapping from policy parameters $\boldsymbol{\theta} \in \mathbb{R}^D$ to a sequence of H, U-dimensional pressure commands $\{\boldsymbol{u}_t\}_{t=1}^H$. One potential method is to allocate a single "free parameter" to each degree of freedom. For a task horizon $H = 5$ on the soft robot, this translates to 60 parameters because we need a 12 DOF pressure command at each time step in the task horizon. However, using a GP as a surrogate model imposes restrictions both on the size of the training data $\mathcal{D}$ and the dimensionality of the parameter space $\mathbb{R}^D$. In fact, for computational feasibility, the recommended number of dimensions in $\boldsymbol{\theta}$ is $D < 20$ [12]. Various strategies have been presented in previous works to reduce the dimensionality of the control space when using GP-based BayesOpt. [8] discretized the continuous parameters of a bipedal robot's leg joints into three states—*flex*, *hold*, and *extend*—before fine-tuning a finite-state machine over these states using BayesOpt. Similarly, [13] confined the actuation of each leg of a spider robot to five discrete settings.

Adopting a parallel strategy, we discretize our soft robot's command pressures into P distinct values spanning from minimum to maximum pressures. Under the discretization P, we generate the set of all possible actions $\mathcal{A}_P = \{\boldsymbol{u}_i\}_{i \in \mathcal{I}}$ with $\mathcal{I} \subset \mathbb{N}$ serving as the index set. Our control policy, $\pi \circ g$, is defined in this indexed space, mapping parameters $\boldsymbol{\theta} \in \mathbb{R}^D$ to an action sequence within $\mathcal{A}_P$. Here, $g : \boldsymbol{\theta} \to \mathcal{I}^H$ is a transformation that converts the continuous parameters into H indices from the set $\mathcal{I}$. Unlike the "free parameter" setting, our index set control policy only requires $D = H$ parameters.

The problem of choosing a sequence of H action indices from a set of $M = |\mathcal{A}_P|$ actions has M^H possible solutions (i.e., action sequences). We condense the size of this search space by pairing pressure chambers such that we let $\rho_i = \rho_{\max} - \rho_j$ for a given antagonistic pair $(\rho_i, \rho_j) \in \boldsymbol{u}$ on our large scale soft robot (See Figure 1). This reduces the overall degrees of freedom from $U = 12$ to $U = 6$. However, evaluating all action sequences on the physical soft robot to find the most performant one is still impractical even for small values of P. As such, we employ BayesOpt to efficiently search for "objective-maximizing" policy parameters in a handful of trials.

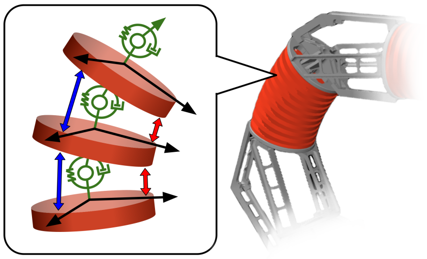


Fig. 2: The simulated version of the physical robot. Each continuum joint is approximated by individual disks that are connected by universal joints with springs and dampers connected in parallel, shown in green. Compressive antagonistic forces from pneumatic actuators are applied between disks, along the x and y axes, shown in red and blue, respectively.

IV. Simulation Experiments and Results

A. Simulation Description

We use MuJoCo [14] to simulate the physical soft robot for the simulation experiments. We approximate each continuum joint with a series of thin disks, as shown in Figure 2. We use the simulator to evaluate the performance of our BayesOpt pipeline, run ablation studies, and compare different policy optimization approaches.

Below, we detail dynamic tasks that leverage the flexible characteristics of the soft robot, but also showcase the difficulty of finding effective mappings between control inputs and effective performance.

1) Throwing Task: For this task, we attach a prismatic gripper to the simulated robot, allowing it to hold and release a 0.25 kg "life-size" object. We include an additional parameter $t_{\mathrm{R}} \in [1, \ldots, H]$, which corresponds to the release time of the cube and set up an objective function

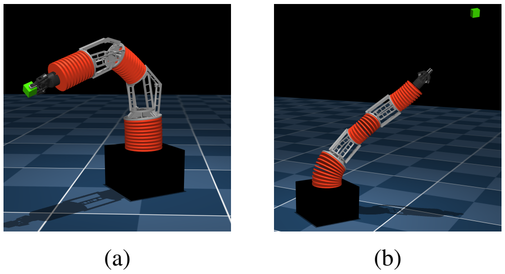
$$J(\boldsymbol{\theta}) = \sum_{t=1}^{t_{\mathrm{R}}} \|\dot{\boldsymbol{x}}_t\| + \mathbf{1}_H(t) \cdot D_{\text{cube}}, \tag{9}$$

where $\dot{\boldsymbol{x}}_t$ corresponds to the end-effector velocity and D_{cube} is the displacement in meters of the thrown object from the base of the robot. $\mathbf{1}_H(t)$ is an indicator function which returns 1 when $t = H$ and zero elsewhere.

The soft robot starts each trial with the object attached to the end-effector (Figure 3a), performs the prescribed pressure commands, and releases the object when time step $t = t_{\mathrm{R}}$ is reached (Figure 3b). We use BayesOpt to find policy parameters $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_H, t_{\mathrm{R}}]$ that maximize Equation (9).

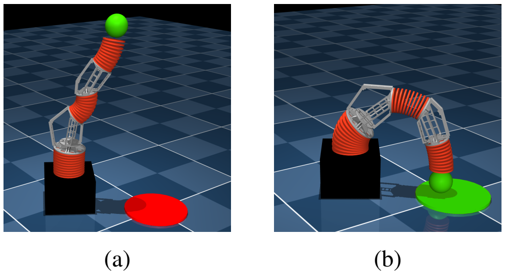
2) Hammering task: To set up a hammering task, we attach a 2kg mass to the tip of the robot to use as the "hammer" and place a force sensor (diameter = 0.5 m) near the simulated soft robot at a fixed Cartesian position $\boldsymbol{g} = [-0.5, 0.5, 0.0]$. To activate the sensor as shown in Figure 4b, the robot has to exert a minimum force of 5N. For this task, the objective function

$$J(\boldsymbol{\theta}) = \sum_{t=1}^{H} m\dot{\boldsymbol{x}}_t^z + F_t - \mathbf{1}_{t_0:H}(t) \cdot \|\boldsymbol{x}_t - \boldsymbol{g}\|_2 \tag{10}$$



(a) (b)

Fig. 3: Throwing task: The robot starts with the gripper holding the object 3a), generates high end-effector velocity, and has to release the object at the right time 3b) to throw successfully.



(a) (b)

Fig. 4: Hammering task. Starting from the equilibrium state (Figure 1), the robot has to "wind-up" enough energy 4a) to activate the touch force sensor 4b).

is a cumulative sum of the vertical momentum of the hammer $m\dot{\boldsymbol{x}}_t^z$ and the measured impact force F_t for $t < t_0$. When $t \geq t_0$, we add a Euclidean distance between the hammer and the force sensor, encouraging the robot to make contact with the sensor. We found $t_0 = 0.7H$ to work well in our experiments.

B. Experiment Setup

In this section, we introduce alternative global optimization strategies to compare with our BayesOpt approach: the Cross Entropy Method (CEM) [15] and Random Search. These methods are configured as follows:

1) CEM: We set the population size to 50 trials and the elite percentage to 20%. To determine the initial population distribution parameters, we randomly sample $D + 1$ trials, compute a sample mean and artificially set the variance to cover all possibilities for $\boldsymbol{\theta}$ in $\mathbb{R}^D$.

2) Random Search: Similar to CEM, Random Search searches through a population of 50 trials at each optimization step. Unlike BayesOpt which has access to a surrogate model for the objective, both CEM and Random Search have to evaluate all candidates in the population on the robot.

We focus on the pressure command discretization hyperparameter P. In general, higher P allows for more granular pressure commands and may result in the discovery of better control sequences $\boldsymbol{u}_{1:H}$. However, the size of the policy parameter search space M^H (where $M = |\mathcal{A}_P|$) also increases greatly. For all experiments, we set a task horizon of $H = 10$ over a period of 5 seconds. The transformation g

	# Trials	$P = 2$	$P = 5$	$P = 7$	$P = 10$
BayesOpt-UCB ($\kappa = 0.9$)	100	28.53 ± 4.03	25.96 ± 2.86	21.84 ± 7.43	21.24 ± 6.14
	500	$\mathbf{34.18 \pm 5.42}$	29.09 ± 3.66	26.02 ± 7.86	28.44 ± 6.0
BayesOpt-LEI	100	30.37 ± 6.86	$\mathbf{30.35 \pm 6.55}$	$\mathbf{32.21 \pm 3.71}$	$\mathbf{31.51 \pm 4.54}$
	500	$\mathbf{37.7 \pm 1.68}$	$\mathbf{38.29 \pm 2.58}$	$\mathbf{40.89 \pm 1.38}$	$\mathbf{40.17 \pm 1.83}$
CEM	100	28.94 ± 2.76	20.73 ± 0.92	19.72 ± 1.99	20.63 ± 2.06
	500	33.65 ± 1.79	24.03 ± 2.63	22.73 ± 2.79	22.87 ± 3.42
	10K	$\mathbf{41.94 \pm 1.9}$	$\mathbf{38.93 \pm 1.9}$	$\mathbf{36.94 \pm 2.76}$	$\mathbf{35.41 \pm 2.41}$
Random Search	100	27.44 ± 1.35	22.13 ± 1.76	19.17 ± 1.8	20.61 ± 3.25
	500	30.35 ± 1.84	23.84 ± 1.69	22.33 ± 1.07	22.27 ± 1.92
	10K	34.13 ± 1.47	28.83 ± 1.46	26.03 ± 0.49	26.63 ± 2.96

TABLE I: Simulated throwing performance — Best solution so far, averaged across 5 random seeds. The top three methods for each discretization setting P, are marked in bold.

	# Trials	$P = 2$	$P = 5$	$P = 7$	$P = 10$
BayesOpt-UCB ($\kappa = 0.9$)	100	1.14 ± 0.48	1.34 ± 0.62	2.59 ± 1.02	1.6 ± 1.29
	500	1.78 ± 0.4	$\mathbf{2.55 \pm 0.71}$	$\mathbf{2.72 \pm 1.06}$	2.28 ± 0.91
BayesOpt-LEI	100	$\mathbf{2.57 \pm 0.67}$	1.92 ± 0.94	$\mathbf{2.62 \pm 0.96}$	$\mathbf{2.84 \pm 0.83}$
	500	$\mathbf{3.39 \pm 0.37}$	$\mathbf{4.22 \pm 1.1}$	$\mathbf{3.53 \pm 0.19}$	$\mathbf{3.54 \pm 0.42}$
CEM	100	1.18 ± 0.47	1.14 ± 0.15	1.16 ± 0.18	1.16 ± 0.32
	500	1.49 ± 0.27	1.38 ± 0.27	1.3 ± 0.29	1.23 ± 0.31
	10K	2.17 ± 0.8	2.03 ± 0.6	2.34 ± 0.64	2.17 ± 0.54
Random Search	100	1.44 ± 0.62	1.14 ± 0.14	1.15 ± 0.07	1.08 ± 0.12
	500	2.03 ± 0.22	1.44 ± 0.35	1.21 ± 0.16	1.24 ± 0.13
	10K	$\mathbf{3.0 \pm 0.54}$	$\mathbf{2.44 \pm 0.23}$	2.37 ± 0.08	$\mathbf{2.28 \pm 0.3}$

TABLE II: Simulated hammering performance — Best solution so far, averaged across 5 random seeds. The top three methods for each discretization setting P, are marked in bold.

scales the parameter values to the range $[0, M]$, and "floors" to the nearest integer, yielding an index from the set $\mathcal{I}$.

C. Results

Table I and Table II present performance and sample-efficiency comparisons between BayesOpt, CEM, and Random Search. We observe that BayesOpt-UCB and BayesOpt-LEI perform comparably or surpass CEM and Random Search, particularly for limited-data scenarios (100 and 500 trials). For $P = 2$, all methods yield similar performance after 100 trials in the throwing task. This is possibly due to there being many sub-optimal solutions. However, the hammering task is less forgiving and mainly rewards contact between the robot and the force sensor. For this task, BayesOpt with LEI is superior. As P increases, we see the benefit of the LEI acquisition function being able to efficiently explore in high dimensions as highlighted in [11]. This suggests that LEI is well-suited for optimizing high-dimensional control problems in this domain. We found that UCB acquisition (with $\kappa = 0.9$) function also performs reasonably well across all settings of P for both tasks. The higher value for κ improved exploration but also increased the standard deviation in performance. A video showing the final policies for throwing and hammering in simulation can be seen at `https://youtu.be/OHcxJl7rC4E`.

D. Reinforcement Learning Experiments

The absence of rich state information makes applying traditional feedback control approaches, including reinforcement learning (RL), particularly challenging in this setting. Nonetheless, we investigated the use of the recent REDQ [16] agent, a data-efficient, model-free RL algorithm.

For the REDQ agent, we provided end-effector velocities, positions of task-relevant objects, and step-wise rewards derived from the objective function. Both the policy and critic use a two-layer neural network with a hidden size of 256. We use an ensemble size of 2 for the critic. We use a high Update-To-Data (UTD) ratio of 16, and, for fair comparison, provided 2500 transitions (250 trials) as pre-training data.

Our findings (Table III) indicate that REDQ struggles to learn effective policies for both dynamic tasks when given a limited data budget and restricted state information.

The results for simulated experiments show that BayesOpt is a viable strategy for controlling soft robots in dynamic task scenarios where the robot dynamics are complex. Its sample efficiency minimizes the required number of physical trials, preserving robot integrity and making BayesOpt highly attractive for real-world soft robot applications.

V. Real-robot Experiments

A. Hardware Setup

We simplified the hardware setup by not attaching a gripper or object, focusing on maximizing the tip velocity over a five-second trajectory instead. While this is not equivalent to throwing, the tip velocity should correlate with the velocity imparted to an object when thrown, allowing us to validate whether BayesOpt can find an effective policy on hardware. We measured the tip position and velocity using an HTC Vive Tracker attached via a blue 3D printed fixture (see Figure 1). For safety of the robot, we set nominal pressures for all twelve pressure commands to lift the robot tip away from the base. Any commands generated by BayesOpt or randomized policies were added to these nominal pressures. Additionally,

# Trials	Without Approximate Joints		With Approximate Joints	
	Throw	Hammer	Throw	Hammer
250	14.78 ± 2.94	0.12 ± 0.07	15.72 ± 2.77	0.24 ± 0.21
500	27.36 ± 0.86	1.41 ± 0.13	30.35 ± 1.81	2.1 ± 0.42
10K	33.02 ± 0.98	2.34 ± 0.06	37.72 ± 2.22	3.12 ± 0.07

TABLE III: Simulated results for the REDQ agent with and without the presence of approximate joint state information — Best solution so far, averaged across 5 random seeds.

BayesOpt only varied the pressure in steps of up to 30 kPa, even though the maximum available pressure was around 205 kPa. This constraint limited potential velocities but still allowed the robot to achieve tip velocities over 4 m/s.

Due to the complexity of running randomized trajectories at high speed on hardware, we only allowed BayesOpt to execute 12 random policies for each acquisition function (LEI and UCB), before executing 60 iterations in each case.

B. Hardware Experiment Results

To successfully maximize the robot tip velocity, BayesOpt has to find pressure commands within $\mathcal{A}_P$ that are temporally correlated, (i.e. higher velocities are only possible by storing energy and then releasing it over the five second window).

The UCB acquisition function produced a better overall policy than LEI, though both significantly increased tip velocity compared to random policies. Figure 5 shows the tip velocity over time for a random policy, the best LEI policy, and the best UCB policy, with the UCB policy achieving over 4 m/s. A video of the UCB optimization trial is available at `https://youtu.be/OHcxJl7rC4E`.

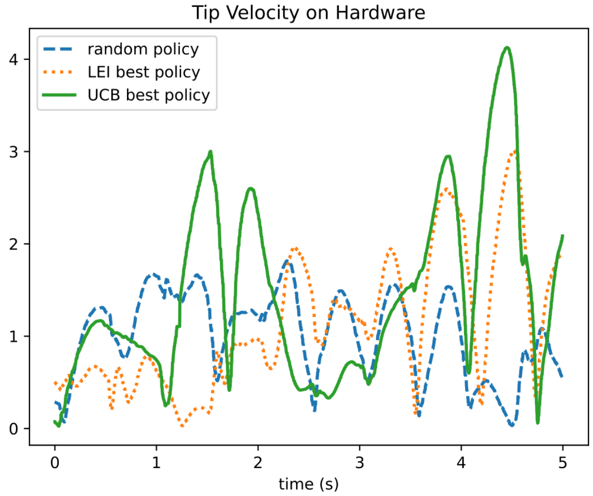


Fig. 5: Tip velocities that resulted from using BayesOpt-LEI, BayesOpt-UCB, and a random policy.

VI. Conclusion

In addition to increased load-bearing capacity, large-scale soft robots also offer inherent advantages in terms of flexibility, compliance, and impact-resistance for challenging tasks in complex environments. Although these properties make them capable of dynamic maneuvers, inherent modeling difficulties present hurdles for real-world deployment. In this work, we have demonstrated a Bayesian optimization (BayesOpt) approach to efficiently learn dynamic, high-dimensional behaviors from simple, low-dimensional controller parameterizations for a large-scale soft robot arm. We bypass explicit kinematic and dynamic modeling, and instead focus directly on optimizing task performance.

We compare against other control approaches which yield good performance when plenty of robot data is available. However, data-efficiency is a high priority in soft robots as the dynamics can exhibit variations and uncertainties over extended use. Our results show that BayesOpt excels in this low-data regime, where precise robot dynamics models are unavailable. Moreover, the simulation results translate well onto the physical soft robot, demonstrating that BayesOpt is able to maximize the tip velocity with only a handful of trials as training data.

References

[1] X. Li, H. Yue, D. Yang, K. Sun, and H. Liu, "A large-scale inflatable robotic arm toward inspecting sensitive environments: Design and performance evaluation," *IEEE Transactions on Industrial Electronics*, 2023.

[2] M. W. Hannan and I. D. Walker, "Kinematics and the implementation of an elephant's trunk manipulator and other continuum style robots," *Journal of Robotic Systems*, vol. 20, no. 2, pp. 45–63, 2003.

[3] D. Trivedi, A. Lotfi, and C. D. Rahn, "Geometrically exact models for soft robotic manipulators," *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 773–780, 2008.

[4] S. Grazioso, G. Di Gironimo, and B. Siciliano, "A geometrically exact model for soft continuum robots: The finite element deformation space formulation," *Soft robotics*, vol. 6, no. 6, pp. 790–811, 2019.

[5] C. C. Johnson, T. Quackenbush, T. Sorensen, D. Wingate, and M. D. Killpack, "Using first principles for deep learning and model-based control of soft robots," *Frontiers in Robotics and AI*, vol. 8, 2021.

[6] S. Luo, M. Edmonds, J. Yi, X. Zhou, and Y. Shen, "Spline-based modeling and control of soft robots," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE, 2020, pp. 482–487.

[7] R. L. Truby, C. Della Santina, and D. Rus, "Distributed proprioception of 3d configuration in soft, sensorized robots via deep learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3299–3306, 2020.

[8] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian optimization for learning gaits under uncertainty: An experimental comparison on a dynamic bipedal walker," *Annals of Mathematics and Artificial Intelligence*, vol. 76, pp. 5–23, 2016.

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

[10] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[11] S. Ament, S. Daulton, D. Eriksson, M. Balandat, and E. Bakshy, "Unexpected improvements to expected improvement for Bayesian optimization," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[12] P. I. Frazier, "A tutorial on Bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[13] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[14] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[15] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer, "The cross-entropy method for optimization," in *Handbook of statistics*. Elsevier, 2013, vol. 31, pp. 35–59.

[16] X. Chen, C. Wang, Z. Zhou, and K. Ross, "Randomized ensembled double q-learning: Learning fast without a model," *arXiv preprint arXiv:2101.05982*, 2021.