2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

Efficient Model Identification for Tensegrity Locomotion

Shaojun Zhu, David Surovik, Kostas Bekris and Abdeslam Boularias

Abstract— **This paper aims to identify in a practical manner unknown physical parameters, such as mechanical models of actuated robot links, which are critical in dynamical robotic tasks. Key features include the use of an off-the-shelf physics engine and the Bayesian optimization framework. The task being considered is locomotion with a high-dimensional, compliant Tensegrity robot. A key insight, in this case, is the need to project the space of models into an appropriate lower dimensional space for time efficiency. Comparisons with alternatives indicate that the proposed method can identify the parameters more accurately within the given time budget, which also results in more precise locomotion control.**

I. Introduction

This paper presents an approach for model identification by exploiting the availability of off-the-shelf physics engines used for simulating dynamics of robots and objects they interact with. There are many examples of popular physics engines that are becoming increasingly efficient [1]–[6]. These physics engines receive as input mechanical and mesh models of the robots in a particular scene, in addition to controls (force, torque, velocity, etc.) applied to them, and return a prediction of the robot's dynamical response.

The accuracy of the prediction depends on several factors. The first one is the limitation of the mathematical model used by the engine (e.g., the Coulomb approximation). The second factor is the accuracy of the numerical algorithm used for solving the equations of motion. Finally, the prediction depends heavily on the accuracy of the physical parameters of the robots, such as mass and friction. In this work, we focus on the last factor and propose a method to fine-tune the physical parameters used in the physics engine.

In the context of compliant locomotion systems, the Tensegrity robot in Fig. 1 is a structurally compliant platform that can distribute forces into linear elements as pure compression or tension [7]. This robot's tensile elements can be actuated, enabling it to effectively adapt to complex contact dynamics in unstructured terrains. A policy for a rolling locomotive gait of the platform has been learned from simulation [8].

Tensegrity robots are inherently high-dimensional, highly-dynamic systems, and providing a predictive model requires a physics-based simulator [9]. The accuracy of such a solution critically depends upon physical parameters of the robot, such as the density of its rigid elements and the elasticity of the tensile elements. While a manual process can be followed to tune a simulation to match the behavior of a real prototype [10], it is highly desirable to conduct this calibration using as few observed trajectories as possible.

This work was sponsored by NSF IIS-1734492, IIS-1723869, a NASA ECF award to Dr. Bekris and U.S. Army Research Lab Collaborative Agreement W911NF-10-2-0016. The authors are with the Department of Computer Science, Rutgers University, New Jersey, USA {shaojun.zhu, david.surovik, kostas.bekris, abdeslam.boularias}@cs.rutgers.edu

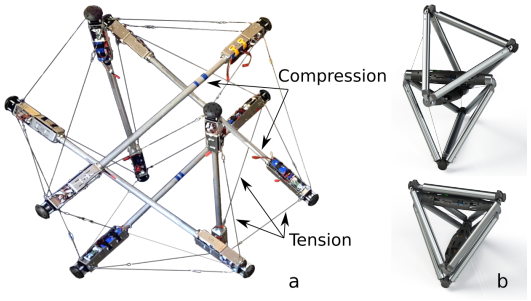

Fig. 1: Tensegrity robots: a) NASA SUPERball: a robotic icosahedron with 6 rods and 24 cables [13]. b) A duct climbing robot: 2 tetrahedral frames with 8 actuated cables [14].

In this work, trajectories generated by a simulation, that is manually tuned to a prototypical robotic platform, are used to identify the parameters of a physics engine for tensegrity modeling. Given the high-dimensionality of the parameter space, this is a challenging problem. This work proposes mapping the model parameters to a lower dimensional space of parameters. Methods used for dimensionality reduction include Random Embedding (REMBO) [11] as well as Variational Auto Encoder (VAE) [12].

Furthermore, this work proposes to tie the dimensionality reduction process with the task performance by first learning a simplified dynamics model, then utilizing it to train an auto-encoder in the parameter space. Bayesian optimization is then conducted in the encoded space, avoiding much of the burden of high dimensionality. The proposed method is able to efficiently identify the parameters that produce a simulation that most closely matches the observed ground-truth trajectories of this locomotive platform.

II. Foundations and Contributions

Two high-level approaches exist for learning robotic tasks with unknown dynamical models: model-free and model-based ones. Model-free methods search for a policy that best solves the task without explicitly learning the system dynamics [15]–[18]. A relative entropy policy search has been used [19] to successfully train a robot to play table tennis. The PoWER algorithm [20] is another model-free policy search approach widely used in robotics.

Model-free methods, however, do not easily generalize to unseen regions of the state-action space. To learn an effective policy, features of state-actions in learning and testing should be sampled from distributions that share the same support.

This is rather dangerous in robotics, as a poor performance during testing could lead to irreversible damages.

Model-based approaches explicitly learn the dynamics of the system and search for an optimal policy using standard simulation, planning, and actuation control loops for the learned parameters. There are many examples of model-based approaches for robotic manipulation [21]–[25], some of which have used physics-based simulation to predict the effects of pushing flat objects on a smooth surface [21]. A non-parametric approach was employed for learning the outcome of pushing large objects (furniture) [23]. A Markov Decision Process (MDP) has been applied to model interactions between objects; however, only simulation results on pushing were reported [24]. For general-purpose model-based reinforcement learning, the PILCO algorithm has been proven efficient in utilizing a small amount of data to learn dynamical models and optimal policies [26].

Bayesian Optimization (BO) is a popular framework for data-efficient black-box optimization [27]. In robotics, some recent applications include learning controllers for bipedal locomotion [28], gait optimization [29] and transfer policies from simulation to real world [30].

Traditional system identification builds a dynamics model by minimizing prediction error (e.g., using least squares) [31], [32]. There have been attempts to combine parametric rigid body dynamics models with nonparametric model learning for approximating the inverse dynamics [33]. In contrast to such methods, this work uses a physics engine, and concentrates on identifying mechanical properties instead of learning the models from scratch. Recent work also proposed model identification for predicting low dimensional physical parameters, such as either mass or friction [34], before searching for an optimal policy.

The present work is a model-based approach that utilizes a physics engine and concentrates on identifying only the mechanical properties of the objects instead of recreating the dynamics from scratch. Furthermore, it utilizes BO and identifies a dimensionality reduction process for dealing with high-dimensional model identification challenges efficiently.

III. Model Identification

This work proposes an online approach for robots to learn the physical parameters of their dynamics through minimal physical interaction. Because of the high dimensionality of the parameter space of the tensegrity robot, even efficient methods such as Bayesian optimization (BO) struggle to identify all parameters with sufficient accuracy.

This section introduces the overall framework of the model identification process. Dimensionality reduction methods, which decrease the search space of BO in order to achieve efficient optimization, are then covered in the next section.

For the tensegrity robot, the physical properties of interest correspond to the density, length, radius, stiffness, damping factor, pre-tension, motor radius, motor friction, and motor inertia of the various rigid and tensile elements and actuators which are modeled in the NASA's Tensegrity Robotics Toolkit (NTRT) [9]. In total, 15 different parameters are considered.

These physical properties are represented as a D-dimensional vector $\theta \in \Theta$, where Θ is the space of all possible values of the physical properties. Θ is discretized with a regular grid resolution. The proposed approach returns a distribution P on discretized Θ instead of a single point $\theta \in \Theta$. This is appropriate due to the fact that model identification is generally an ill-posed problem, where multiple models can explain an observed trajectory with equal accuracy. The objective is to preserve all possible explanations for the purposes of robust planning.

The online model identification algorithm (given in Algorithm 1) takes as input a prior distribution P_t, for time-step $t \geq 0$, on the discretized space of physical properties Θ. P_t is calculated based on the initial distribution P_0 and a sequence of observations $(x_0, \mu_0, x_1, \mu_1, \ldots, x_{t-1}, \mu_{t-1}, x_t)$. For the Tensegrity robot, x_t is a state vector concatenating the 3D centers of all rigid elements, i.e., the rods in the corresponding Figure 1, and μ_t is a vector of motor torques.

Input: State-action-state data $\{(x_i, \mu_i, x_{i+1})\}, i = 0, \ldots, t$;
Θ, a discretized space of physical properties;
Output: Probability distribution P over Θ ;
Sample $\theta_0 \sim \text{Uniform}(\Theta)$; $L \leftarrow \emptyset$; $k \leftarrow 0$;
repeat
  $l_k \leftarrow 0$;
  for $i = 0$ **to** t **do**
    Simulate $\{(x_i, \mu_i)\}$ using a physics engine with physical parameters θ_k and get the predicted next state $\hat{x}_{i+1} = f(x_i, \mu_i, \theta_k)$;
    $l_k \leftarrow l_k + \|\hat{x}_{i+1} - x_{i+1}\|_2$;
  end
  $L \leftarrow L \cup \{(\theta_k, l_k)\}$;
  Calculate $GP(m, K)$ on error function E, where $E(\theta) = l$, using data $(\theta, l) \in L$;
  $\theta_{k+1} = \arg\max_{\theta \in \Theta} EI(\theta)$;
  $k \leftarrow k + 1$;
until *Timeout*;

Algorithm 1: Model Identification with Bayesian Optimization

The process consists of simulating the effects of the controls μ_i on the robot in states x_i under various values of parameters θ and observing the resulting states $\hat{x}_{i+1}$, for $i = 0, \ldots, t$. The goal is to identify the model parameters that make the outcomes $\hat{x}_{i+1}$ of the simulation as close as possible to the real observed outcome x_{i+1}. In other terms, the following black-box optimization problem is solved:

$$\theta^* = \arg\min_{\theta \in \Theta} E(\theta) \stackrel{def}{=} \sum_{i=0}^{t} \|x_{i+1} - f(x_i, \mu_i, \theta)\|_2, \qquad (1)$$

wherein x_i and x_{i+1} are the observed states of the robot at times i and $i+1$, μ_i is the control that applied at time t, and $f(x_i, \mu_i, \theta) = \hat{x}_{i+1}$, the predicted state at time $t+1$ after simulating control μ_i at state x_i using physical parameters θ.

The proposed approach consists of learning the error function E from a sequence of simulations with different parameters $\theta_k \in \Theta$. To choose these parameters efficiently in

a way that quickly leads to accurate parameter estimation, a belief about the actual error function is maintained. This belief is a probability measure over the space of all functions $E : \mathbb{R}^D \rightarrow \mathbb{R}$, and is represented by a Gaussian Process (GP) [35] with mean vector m and covariance matrix K. The mean m and covariance K of the GP are learned from data points $\{(\theta_0, E(\theta_0)), \ldots, (\theta_k, E(\theta_k))\}$, where θ_k is a vector of physical properties of the object, and $E(\theta_k)$ is the accumulated distance between actual observed states and states that are obtained from simulation using θ_k. High-fidelity simulations are computationally expensive. It is therefore important to minimize the number of simulations, i.e., evaluations of function E while searching for the optimal parameters that solve Eq. 1. BO decides the location for next sample by optimizing the acquisition function. In our experiments, the expected improvement (EI) acquisition function [36] is used.

IV. Dimensionality Reduction

A. Random Embedding for Model Identification

For problems where space Θ of physical properties has a high dimension D, the method presented in Algorithm 1 is not practical because the number of elements in discretized Θ is exponential in dimension D. This is a common problem in global search methods [11]. In fact, it has been shown that BO techniques do not perform better than a random search when the dimension of the search space is too large (10 dimensions in the experiment in [37]). Therefore, Algorithm 1 cannot be directly used for robotic platforms with a large number of joints and parameters, such as the Tensegrity robot.

Dimensionality reduction is a popular solution to the problem of searching in high-dimensional spaces. This solution is particularly appealing in the context of this work because we are more interested in the accuracy of the predicted trajectory than in identifying the true underlying physical parameters. Mechanical models of motion tie together several parameters of an object. For example, in Coulomb's model, the mass and the friction of an object are used in a linear function to predict its planar sliding motion. Therefore, one can linearly map these two parameters to a single parameter and still make accurate predictions. Similarly, random embedding (REMBO) [11] is an effective dimensionality reduction technique in the context of BO.

B. Variational Auto Encoder for Model Identification

An autoencoder is a neural network that learns to reconstruct the input by going through a latent space that has a lower dimension than the original input space [38]. Autoencoders have been shown to be useful in unsupervised learning of low dimensional representations. A variational autoencoder (VAE) adds an additional constraint that the latent space follows a prior distribution, usually assumed to be Gaussian [12]. This constraint makes the model more useful as a generative model, as it also learns to generate output from the prior distribution in addition to reconstruction.

We adapt the VAE and combine it with the BO process, as shown in Fig. 2. First, the VAE is trained with randomly sampled physical parameters θ to learn a low dimension

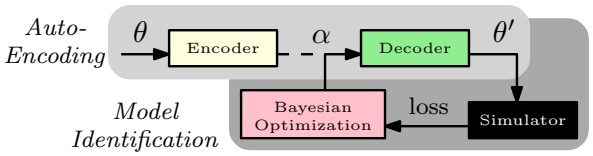

Fig. 2: The auto-encoder is trained first to learn the latent low-dimensional embedding. Then BO is performed in this low-dimensional space to identify the optimal model parameter. The decoder is used to reconstruct the original 15 dimensional parameter in order to perform physical simulation.

embedding α. The decoder is used to project the low dimensional α back to a value θ' in the original physical parameter space. Thus, the BO process as detailed in Algorithm 1 can be performed efficiently in the low-dimensional space.

C. Auto-Encoder with Learned Dynamics

The use of VAE for reconstructing parameters from a low dimensional space has some limitations. Specifically, we are more interested in the accuracy of the predicted trajectory than in identifying the true underlying physical parameters. Mechanical models of motion can tie together several parameters of a model. Thus, connecting the dimensionality reduction process directly with the task performance may further improve the performance when using the identified model on the task. This idea is similar to learning a locally linear dynamics model while aiming to maximize the performance of the controller [39].

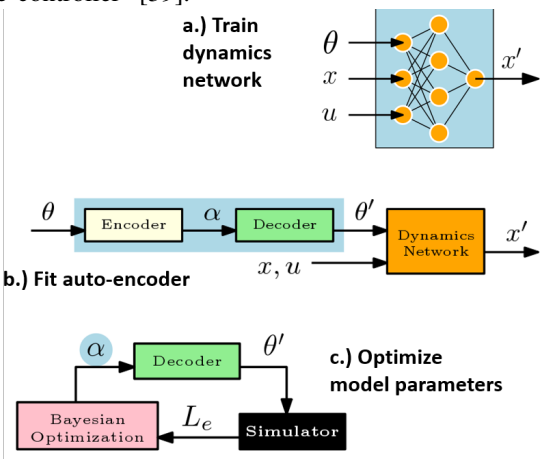

Fig. 3: The three-step process for model identification under reduced parameter dimensionality. For each step, the element being updated is highlighted in blue.

To provide intuition, we begin with an illustrative toy example of pushing a point-mass along a single dimension. We then show how the same approach can be applied to a much more complex system such as the Tensegrity robot.

1) Toy example: Pushing: Consider a cube of mass m resting on a surface, as in Fig. 4, which can be represented by a point that can only move along one axis. Assuming uniform and

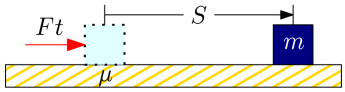

Fig. 4: 1-D point pushing example.

constant coefficient of kinetic friction μ between the cube and its resting surface, an impulse Ft is applied to the cube to cause displacement across some distance S. Applying Newton's Laws and Kinematic Equations, we have the following equations:

$$Ft = \mu m t_1 = m v_0 \tag{2}$$

$$S = v_0 t_1 / 2 \tag{3}$$

where v_0 is the initial velocity of the cube after the impulse and t_1 is the duration of the cube's movement. Solving the above equations, we have:

$$S(m, \mu) = (Ft)^2 / (2m^2 \mu) \tag{4}$$

We use this equation solely to generate training and testing data, in the same manner as a black-box simulator. This parallels our later use of a physics simulator for the Tensegrity robot without direct exposure of the differential equations of motion. Our goal is to identify m and μ, given only the initial impulse Ft and the displacement S, without explicitly solving the system's dynamics equations.

This problem, however, is ill-posed due to the fact that different values of m and μ can result in the same S for a given value of Ft. For example, assuming $Ft = 1$, both $m = 1, \mu = 0.32$ and $m = 0.8, \mu = 0.5$ will result in $S = 1.5625$. In other words, as long as the value of $m^2\mu$ is uniquely identified, so is the displacement S. Thus, if the task is to predict S, it is not necessary to individually identify m and μ; the scalar value $m^2\mu$ can still uniquely determine the system. The goal then is to identify this one-dimensional representation automatically.

First, we train a dynamics network to predict the displacement S given inputs m, μ, and Ft, as shown in Fig. 3a. In this case $\theta = [m, \mu]$ and $u = Ft$. The input state x is omitted as we assume the point is always at the origin before being pushed. Then, we use the resulting dynamics network to train the auto-encoder to reconstruct m and μ, as shown in Fig. 3b. During this step, the weights in the dynamics network are fixed. The encoder module is designed to receive input m, μ and output one-dimensional α, which is influenced by the previous observation. A unique aspect of this auto-encoder is that, instead of using reconstruction error of m and μ as the loss function, it passes the reconstructed parameters to the dynamics network and back-propagates the gradient of the displacement error. The goal of the auto-encoder is to reconstruct parameters resulting in similar dynamics, as predicting dynamics is the primary concern.

2) Tensegrity Robot: This procedure is next applied to the much more complex Tensegrity robot system. Here, we assume the existence of a low-dimensional representation of both the physical parameter space and the state-action space that, once identified, can sufficiently determine the system dynamics, similar to the role of $m^2\mu$ in Sec. IV-C.1.

One challenge in adapting this procedure is that the Tensegrity robot is inherently a high-dimensional, highly-dynamic system which makes learning a dynamics model extremely difficult. Instead, a simplified state-action model can be learned where instead of using the full state which is 126-dimensional, only the height of the center of mass of the robot is used as the state.[1] In our experiments, we found that using the full state as input state and the height of the center of mass as output results in better accuracy than using only the height of the center of mass as both input and output.

Thus, the simplified dynamics model takes as input model parameters θ, full state x, and action u and predicts the height of the center of mass x' at the next time-step, as shown in Fig. 3a. In this case, θ is the 15-dimensional parameter, x is the 126-dimensional full state, u is the 24-dimensional action and x' is the 1-dimensional height of the center of mass of the robot. Using the simplified dynamics model, we train an auto-encoder on top of it to learn the low-dimensional representation α of the original parameters θ.

Similar to the 1D pushing example, the auto-encoder in Fig. 3b is trained using loss function of the error between the new state simulated using original parameter θ and the new state predicted by the dynamics model using the reconstructed parameter θ'. Afterward, BO is performed on the low-dimensional parameters α, while using the decoder to project back to the original space for simulations with the physics engine, as shown in Fig. 3c.

V. Experimental Results

A. Toy example: 1-D Point Pushing

In this toy example, we use equation 4 as the "simulator" to generate data. 20,000 training data points are generated by randomly sampling m, μ, F and t. 2,000 additional data points are generated for testing. As a proof of concept, we only compare model identification using BO in the original 2D parameter space and in the 1D latent space in the auto-encoder.

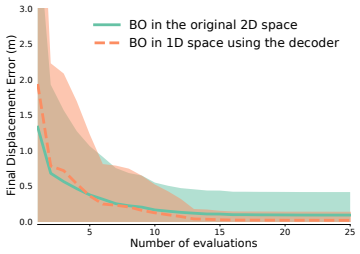

Fig. 5: 1-D point pushing result: Searching in the 1D space is more efficient than in the original 2D space.

The dynamics network has three hidden layers with 64, 128, and 64 hidden units and ReLU activation functions. The encoder network has two layers with 32, 1 units, and decoder has two layers with 1, 32 units. Both the encoder and the decoder only have ReLU on the first hidden layer.

The result is shown in Fig. 5. After training, the decoder is able to reconstruct m and μ which results in close to zero error in the final displacement. Comparing to optimizing in the original 2D parameter space, using the decoder to optimize in 1D space is both more efficient and achieves lower error.

[1]The selection of the state representation is not a focus of the paper. The search for optimal state representation is left for future work.

B. Tensegrity Robot

Setup: This experiment aims to identify the 15 parameters of the model of the Tensegrity SuperBall robot in NASA's Tensegrity Robotics Toolkit [9]. The complex dynamics and high dimensionality of the robot make this problem very hard.

The applied control law conducts an optimization procedure on the system's geometric configuration alone, without accounting for dynamics [40]. Under the assumption that the base triangle remains in full contact with the ground, this law commands a change in cable lengths that correspond to a desired shift in the system's center-of-mass. By displacing this value relative to the supporting base triangle, the system can be made dynamically unstable, causing a forward flop.

By using the controller above in simulation, 1200 trajectories are generated as training data by sampling the 15 parameters within the $\pm 10\%$ range of the ground truth. The assumption is that such error can appear during the robot modeling process and the proposed approach should be able to minimize such error. Examples of the trajectories can be found on `https://goo.gl/3LC92a`.

The proposed approach using BO with an auto-encoder and a dynamics network is compared with: Random Search in 15-D, BO in 15-D, REMBO [11] in 5-D, and BO with VAE in 5-D.

The encoder and decoder of the VAE used in the experiment are both two-layer neural networks. The input dimension of the encoder and the output dimension of the decoder is 15, which is the dimension of the parameter space. The latent space is 5 dimensional. Between them is one layer of 400 dimensions. This dimension is chosen through cross-validation by balancing accuracy and network complexity. The prior distribution of the latent space in the VAE is assumed to be $N(0,1)$. Based on the three-sigma rule, when sampling between $[-3,3]$, this interval should cover 99.7% of the latent space when the VAE is optimized. For REMBO, each time a random projection matrix is generated to project the parameters into $[0,1]$.

The dynamics network is a four-layer neural network with dimensions 128, 64, 32, and 1. The encoder and decoder with the dynamics network are also two-layer neural networks but with only 10 and 5 dimensions for each layer.

Results: Fig. 6 shows the average error between the trajectories using the model parameters identified by different methods and the trajectories generated from the ground-truth simulator. When optimizing in the original 15-dim. space, as a data-efficient global optimization method, BO outperformed random search. Further improvements are achieved by dimensionality reduction, making the search more efficient. BO with the autoencoder with the learned dynamics network in the 5D-space achieves the lowest trajectory error, outperforming the method using REMBO or VAE. This shows that a learned better latent embedding enables more efficient parameter search in the BO process.

Table I provides the errors for each of the final identified parameter. Interestingly, although achieved lowest trajectory error, BO with AE and Dynamics Net did not identify all parameters with lowest error. Specifically, it turns out that parameters like rod_length, rod_space, rod_length_mp, motor_radius, motor_friction, and motor_inertia are not actually used in the current model of the SuperBall simulation. Thus even methods like VAE may be able to get lower reconstructing error on the parameters themselves, BO with AE and Dynamics Net is able to achieve lower trajectory error as it ties the model identification process with the dynamics. Additionally, some parameters may have a stronger influence on the robot dynamics. An intelligent way to identify these parameters would be helpful to reduce the dimensionality of the parameter space and could be more informative than random embeddings. This will be a direction for future work.

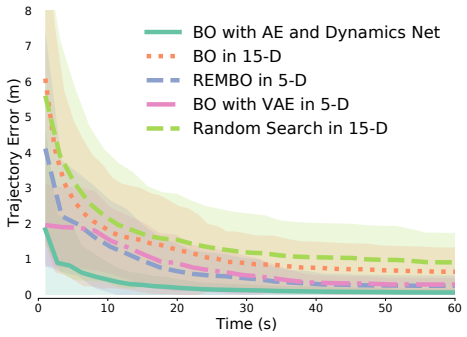

Fig. 6: Test trajectory errors of different methods for the Tensegrity robot as a function of time budget for the parameter optimization process. BO with the autoencoder with the learned dynamics network in the 5-dimensional space achieves the lowest trajectory error, outperforming random search and BO in the original 15 dimensional space, as well as REMBO or VAE in the 5-dimensional space.

VI. Conclusion

This work proposes an information and data efficient framework for identifying physical parameters critical for robotic tasks, such as compliant robot locomotion. The framework aims to minimize the error between trajectories observed in experiments and those generated by a physics engine. To solve high-dimensional challenges, this work integrates BO with a projection to a lower-dimensional space through random embedding or learning a latent embedding utilizing auto encoder. The evaluation of the proposed method against alternatives is favorable both in terms of identifying parameters more efficiently, as well as resulting in more accurate locomotion trajectories.

An interesting extension of this work would involve the identification of controls during the learning process that help in quickly minimizing the error. This can be a robust control process, which takes advantage of BO's output in terms of a belief distribution for the identified parameters, so as to minimize entropy and maximize the safety of the experimentation process. Furthermore, it is interesting to compare the generality of the learned models and resulting control schemes that utilize them against completely model-free and end-to-end approaches for reinforcement learning.

References

[1] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ODE and physx," in

TABLE I: Identified Prameter Error (%)

	Random Search in 15-D	BO in 15-D	REMBO in 5-D	BO with VAE in 5-D	BO with AE and Dynamics Net
density	5.01±2.86	4.10 ± 2.66	1.85±1.88	1.96 ± 0.46	**1.30**±0.05
radius	2.49±1.94	2.08 ± 1.73	1.86± 1.84	1.43±0.28	**0.30**±0.03
density_mp	5.40±2.96	5.19 ± 2.66	1.89± 1.86	2.38±0.39	**1.00**±0.18
radius_mp	4.78±2.78	5.36 ± 2.97	1.94± 1.94	2.00±0.55	**0.69**±0.43
stiffnessActive	4.49±2.68	4.44 ± 2.79	1.84± 1.90	**1.68±0.46**	1.71±0.03
damping	4.62±2.75	4.33 ± 2.78	1.81± 1.89	**2.02±0.44**	2.26±0.15
rod_length	5.05±2.75	4.72 ± 2.69	**1.90**± 1.88	2.04±0.31	6.25±0.59
rod_space	4.96 ±2.81	4.87 ± 2.74	1.88± 1.84	**1.68**±0.36	5.20±0.38
rod_length_mp	4.89 ± 2.81	5.22 ± 2.87	1.88± 1.96	**1.70**±0.58	4.06±0.23
pretension	5.10 ± 2.83	5.10 ± 3.01	1.93± 1.89	1.58±0.50	**1.38**±0.34
maxTens	4.99 ± 2.87	4.66 ± 2.80	1.86± 1.83	**1.85**±0.42	5.48±0.12
targetVelocity	4.85 ± 2.62	5.31 ± 3.01	1.84± 1.90	2.06±0.62	**0.49**±0.31
motor_radius	5.11 ± 2.90	4.38 ± 2.81	1.90± 1.91	**1.79**±0.66	4.9±0.23
motor_friction	5.10 ± 2.71	5.32 ± 3.17	1.89± 1.82	2.19±0.27	**0.65**±0.03
motor_inertia	4.78 ± 2.80	4.87 ± 3.01	1.83± 1.88	2.00±0.45	**1.86**±0.06

IEEE International Conference on Robotics and Automation, ICRA, 2015, pp. 4397–4404.

[2] "Bullet physics engine," [Online]. Available: www.bulletphysics.org.

[3] "MuJoCo physics engine," [Online]. Available: www.mujoco.org.

[4] "DART physics egnine," [Online]. Available: http://dartsim.github.io.

[5] "PhysX physics engine," [Online]. Available: www.geforce.com/hardware/technology/physx.

[6] "Havok physics engine," [Online]. Available: www.havok.com.

[7] K. Caluwaerts, J. Despraz, A. Iscen, A. Sabelhaus, J. Bruce, B. Schrauwen, and V. SunSpiral, "Design and control of compliant tensegrity robots through simulation and hardware validation," *Journal of The Royal Society Interface*, vol. 11, no. 98, 2014.

[8] X. Geng, M. Zhang, J. Bruce, K. Caluwaerts, M. Vespignani, V. SunSpiral, P. Abbeel, and S. Levine, "Deep reinforcement learning for tensegrity robot locomotion," *CoRR*, vol. abs/1609.09049, 2016.

[9] NTRT, "NASA tensegrity robotics toolkit (NTRT)," https://ti.arc.nasa.gov/tech/asr/intelligent-robotics/tensegrity/NTRT/.

[10] B. T. Mirletz, I.-W. Park, R. D. Quinn, and V. SunSpiral, "Towards bridging the reality gap between tensegrity simulation and robotic hardware," in *IEEE/RSJ IROS*, Hamburg, Germany, 2015.

[11] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. de Feitas, "Bayesian optimization in a billion dimensions via random embeddings," *Journal of Artificial Intelligence Research*, vol. 55, pp. 361–387, 2016.

[12] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.

[13] A. P. Sabelhaus, J. Bruce, K. Caluwaerts, P. Manovi, R. F. Firoozi, S. Dobi, A. M. Agogino, and V. SunSpiral, "System design and locomotion of superball, an untethered tensegrity robot," in *ICRA*. IEEE, 2015, pp. 2867–2873.

[14] J. Friesen, A. Pogue, T. Bewley, M. de Oliveira, R. Skelton, and V. Sunspiral, "Ductt: A tensegrity robot for exploring duct systems," in *ICRA*. IEEE, 2014, pp. 4222–4228.

[15] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[16] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, 1st ed. Athena Scientific, 1996.

[17] J. Kober, J. A. D. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *IJRR*, July 2013.

[18] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *NIPS*, 2014.

[19] J. Peters, K. Mülling, and Y. Altün, "Relative entropy policy search," in *Proceedings of the Twenty-Fourth AAAI*, 2010, pp. 1607–1612.

[20] J. Kober and J. R. Peters, "Policy search for motor primitives in robotics," in *NIPS*, 2009, pp. 849–856.

[21] M. Dogar, K. Hsiao, M. Ciocarlie, and S. Srinivasa, "Physics-Based Grasp Planning Through Clutter," in *R:SS VIII*, July 2012.

[22] K. M. Lynch and M. T. Mason, "Stable pushing: Mechanics, controllability, and planning," *IJRR*, vol. 18, 1996.

[23] T. Merili, M. Veloso, and H. Akin, "Push-manipulation of Complex Passive Mobile Objects Using Experimentally Acquired Motion Models," *Autonomous Robots*, pp. 1–13, 2014.

[24] J. Scholz, M. Levihn, C. L. Isbell, and D. Wingate, "A Physics-Based Model Prior for Object-Oriented MDPs," in *ICML*, 2014.

[25] J. Zhou, R. Paolini, J. A. Bagnell, and M. T. Mason, "A convex polynomial force-motion model for planar sliding: Identification and application," in *ICRA*, 2016, pp. 372–377.

[26] M. Deisenroth, C. Rasmussen, and D. Fox, "Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning," in *Robotics: Science and Systems (RSS)*, 2011.

[27] 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.

[28] R. Antonova, A. Rai, and C. G. Atkeson, "Sample efficient optimization for learning controllers for bipedal locomotion," in *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*. IEEE, 2016, pp. 22–28.

[29] 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, 2016.

[30] A. Marco, F. Berkenkamp, P. Hennig, A. P. Schoellig, A. Krause, S. Schaal, and S. Trimpe, "Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with bayesian optimization," in *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, 2017, pp. 1557–1563.

[31] J. Swevers, C. Ganseman, D. B. Tukel, J. De Schutter, and H. Van Brussel, "Optimal robot excitation and identification," *IEEE transactions on robotics and automation*, vol. 13, no. 5, pp. 730–740, 1997.

[32] L. Ljung, Ed., *System Identification (2Nd Ed.): Theory for the User*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999.

[33] D. Nguyen-Tuong and J. Peters, "Using model knowledge for learning inverse dynamics," in *ICRA*. IEEE, 2010, pp. 2677–2682.

[34] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," in *R:SS*, July 2017.

[35] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[36] J. Močkus, "On bayesian methods for seeking the extremum," in *Optimization Techniques IFIP Technical Conference*. Springer, 1975, pp. 400–404.

[37] M. Ahmed, B. Shahriari, and M. Schmidt, "Do we need "harmless" bayesian optimization and "first-order" bayesian optimization?" in *NIPS BayesOPT Workshop*, 2016.

[38] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[39] S. Bansal, R. Calandra, T. Xiao, S. Levine, and C. J. Tomlin, "Goal-driven dynamics learning via bayesian optimization," in *IEEE CDC*, 2017.

[40] Z. Littlefield, D. Surovik, W. Wang, and K. E. Bekris, "From quasi-static to kinodynamic planning for spherical tensegrity locomotion," in *International Symosium on Robotics Research (ISRR)*, Puerto Varas, Chile, 12/2017 2017.