Information-Efficient Model Identification for Tensegrity Robot Locomotion

Shaojun Zhu, David Surovik, Kostas Bekris, Abdeslam Boularias
Department of Computer Science, Rutgers University, New Jersey, USA
{shaojun.zhu, david.surovik, kostas.bekris, abdeslam.boularias}@cs.rutgers.edu

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 data-efficient adaptation of a black-box 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 system identification challenge into an appropriate lower dimensional space. 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.

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 (Erez, Tassa, and Todorov, 2015; Bul; MuJ; DAR; Phy; Hav). 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, friction and elasticity. In this work, we focus on the last factor and propose a method to improve the accuracy of the physical parameters used in the physics engine.

In the context of compliant locomotion systems, the Tensegrity robot of Figure 1 is a structurally compliant platform that can distribute forces into linear elements as pure compression or tension (Caluwaerts et al., 2014). This robot's tensile elements can be actuated, enabling it to effectively adapt to complex contact dynamics in unstructured terrains.

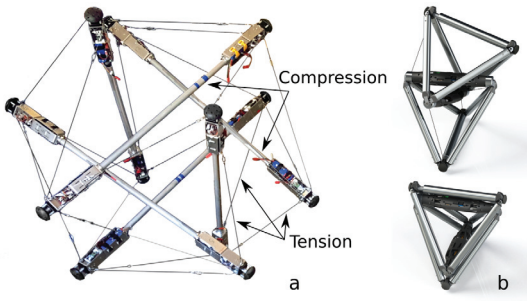

Figure 1: The Tensegrity robot (Caluwaerts et al., 2014).

A policy for a rolling locomotive gait of the platform has been learned from simulated data (Geng et al., 2016). Tensegrity robots are inherently high-dimensional, highly-dynamic systems, and providing a predictive model requires a physics-based simulator (NTRT). 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 (Mirletz et al., 2015), it is highly desirable to conduct this calibration using as few observed trajectories as possible. In this work, trajectories generated by a simulation 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 the mapping of the system identification process to a lower dimensional space of parameters. Methods used for dimensionality reduction include Random Embedding (REMBO) (Wang et al., 2016) as well as Variational Auto Encoder (VAE) (Kingma and Welling, 2014). A data-efficient Bayesian optimization technique is used for searching in the lower dimensional space, instead of the original high dimensional parameter space. 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 exciting locomotive platform.

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 (Sutton and Barto, 1998; Bertsekas and Tsitsiklis, 1996; Kober, Bagnell, and Peters, 2013; Levine and Abbeel, 2014). Model-free methods are accredited with the recent success stories of reinforcement learning in video games (Mnih et al., 2015). For robot learning, a relative entropy policy search has been used (Peters, Mülling, and Altün, 2010) to successfully train a robot to play table tennis. The PoWER algorithm (Kober and Peters, 2009) 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 poor performance in testing could lead to irreversible damage.

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 (Dogar et al., 2012; Lynch and Mason, 1996; Merili, Veloso, and Akin, 2014; Scholz et al., 2014; Zhou et al., 2016), some of which have used physics-based simulation to predict the effects of pushing flat objects on a smooth surface (Dogar et al., 2012). A nonparametric approach was employed for learning the outcome of pushing large objects (furniture) (Merili, Veloso, and Akin, 2014). A Markov Decision Process (MDP) has been applied to modeling interactions between objects; however, only simulation results on pushing were reported (Scholz et al., 2014). 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 (Deisenroth, Rasmussen, and Fox, 2011).

Bayesian Optimization is a popular framework for data-efficient black-box optimization (Shahriari et al., 2016). In robotics, some recent applications include learning controllers for bipedal locomotion (Antonova, Rai, and Atkeson, 2016), gait optimization (Calandra et al., 2016) and transfer policies from simulation to real world (Marco et al., 2017).

This work is based on a model-based approach, which instead of learning a dynamics model, it 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 Bayesian optimization and identifies a process for dealing with high-dimensional system identification challenges efficiently.

Proposed Approach

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 with efficient optimization method like Bayesian optimization (BO), it is still challenging to identify all the parameters efficiently. The overall framework of the model identification process is first introduced, then the approaches of dimensionality reduction to decrease the search space of BO in order to achieve efficient optimization are covered in detail.

Model Identification

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.

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 $\theta \in \Theta$ since model identification is generally an ill-posed problem. In other terms, there are multiple models that 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.

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) (Rasmussen and Williams, 2005) 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.

The probability distribution P on the identity of the best physical model θ^*, returned by the algorithm, is computed

Input: State-action-state data $\{(x_i, \mu_i, x_{i+1})\}$ for $i = 0, \ldots, t$
Θ, a discretized space of possible values of physical properties;
Output: Probability distribution P over Θ according to the provided data;
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 $(\theta, l) \in L$;
  Sample $E_1, E_2, \ldots, E_n \sim GP(m, K)$ in Θ;
  foreach $\theta \in \Theta$ **do**
    $P(\theta) \approx \frac{1}{n} \sum_{j=0}^{n} \mathbf{1}_{\theta = \arg\min_{\theta' \in \Theta} E_j(\theta')}$
  end
  $\theta_{k+1} = \arg\min_{\theta \in \Theta} P(\theta) \log\big(P(\theta)\big)$;
  $k \leftarrow k + 1$;
until *Timeout*;

Algorithm 1: Model Identification with Greedy Entropy Search

from the learned GP as

$$P(\theta) \stackrel{def}{=} P\big(\theta = \arg\min_{\theta' \in \Theta} E(\theta')\big) = \int_{E:\mathbb{R}^D \to \mathbb{R}} p_{m,K}(E) \Pi_{\theta' \in \Theta - \{\theta\}} H\big(E(\theta') - E(\theta)\big) \mathrm{d}E \quad (2)$$

where H is the Heaviside step function, i.e., $H\big(E(\theta') - E(\theta)\big) = 1$ if $E(\theta') \geq E(\theta)$ and $H\big(E(\theta') - E(\theta)\big) = 0$ otherwise, and $p_{m,K}(E)$ is the probability of a function E according to the learned GP mean m and covariance K. Intuitively, $P(\theta)$ is the expected number of times that θ happens to be the minimizer of E when E is a function distributed according to GP density $p_{m,K}$.

Distribution P from Equation 2 does not have a closed-form expression. Therefore, a *Monte Carlo* sampling is employed for estimating P. Specifically, the process samples vectors containing values that E could take, according to the learned Gaussian process, in the discretized space Θ. $P(\theta)$ is estimated by counting the fraction of sampled vectors of the values of E where θ happens to have the lowest value, as indicated in Algorithm 1.

Finally, the computed distribution P is used to select the next vector θ_{k+1} to use as a physical model in the simulator. This process is repeated until the entropy of P drops below a certain threshold, or until the algorithm runs out of the allocated time budget. The entropy of P is given as $\sum_{\theta \in \Theta} -P_{min}(\theta) \log\big(P_{min}(\theta)\big)$. When the entropy of P is close to zero, the mass of distribution P is concentrated around a single vector θ, corresponding to the physical model that best explains the observations. Therefore, the next vector θ_{k+1} should be selected such that the entropy of P would decrease after adding the data point $\big(\theta_{k+1}, E(\theta_{k+1})\big)$ to train the GP and re-estimate P using the new mean m and covariance K in Equation 2.

The Entropy Search method (Hennig and Schuler, 2012) follows this reasoning and use Monte Carlo again to sample, for each potential choice of θ_{k+1}, a number of values that $E(\theta_{k+1})$ could take according to the GP in order to estimate the expected change in the entropy of P and choose the parameter vector θ_{k+1} that is expected to decrease the entropy of P the most. The existence of a secondary nested process of Monte Carlo sampling makes this method impractical for online model identification. Instead, this work proposes a simple heuristic for choosing the next θ_{k+1}. In this method, called *Greedy Entropy Search*, the next θ_{k+1} is chosen as the point that contributes the most to the entropy of P, i.e.,

$$\theta_{k+1} = \arg\max_{\theta \in \Theta} -P(\theta) \log\big(P(\theta)\big).$$

This selection criterion is greedy because it does not anticipate how the output of the simulation using θ_{k+1} would affect the entropy of P. Nevertheless, this criterion selects the point that is causing the entropy of P to be high. That is, a point θ_{k+1} with a good chance $P(\theta_{k+1})$ of being the real model, but with a high uncertainty $P(\theta_{k+1}) \log\big(\frac{1}{P(\theta_{k+1})}\big)$.

Random Embedding for Model Identification in the High Dimensional Space

For problems where the 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 (Wang et al., 2016). In fact, it has been shown that Bayesian optimization techniques do not perform better than a random search when the dimension of the search space is too large (10 dimension in the experiment in (Ahmed, Shahriari, and Schmidt, 2016)). Therefore, Algorithm 1 cannot be directly used for robotic platforms with a large number of joints and parameters, such as the Tensegrity robot or compliant dexterous hands.

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 the motion of a sliding planar object. Therefore, one can map linearly these two parameters to a single parameter and still make accurate predictions of the motion.

Random embedding is an efficient and effective dimensionality reduction technique (Wang et al., 2016). Given a space of parameters Θ with dimension D, we generate a random matrix $A \in \mathscr{R}^{D \times d}$ that projects points from $\Theta \subset \mathbb{R}^D$ to a lower-dimensional space of parameters $\Omega \subset \mathbb{R}^d$ where $d < D$. Instead of discretizing Θ, we discretize Ω into a regular grid and map each point $\omega \in \Omega$ to a point θ in the

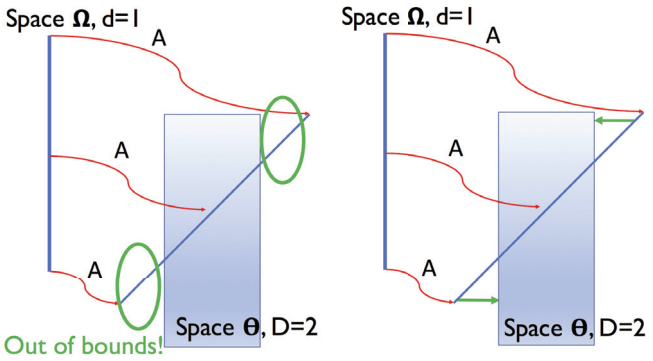

Figure 2: LEFT: A example of 1D-to-2D projection resulting in points outside the original domain. RIGHT: REMBO approaches this issue by projecting the point outside Θ to the nearest boundary point of Θ.

original high-dimensional space by using A, i.e. $\theta = A\omega$. One can show (Wang et al., 2016) that with probability one, $\min_{\theta\in\Theta} E(\theta) = \min_{\omega\in\Omega} E(A\omega)$ where E is the error function in Equation 1. Consequently, we run Algorithm 1 using discretized Ω as input instead of Θ. We project back the low-dimensional vectors $\omega \in \Omega$ to original parameter space Θ using $\theta = A\omega$ when we need to run the physical simulation to get the trajectory under a sampled value of ω.

However, For a randomly generated matrix A and point $\omega \in \Omega$, the corresponding high-dimensional vector $\theta = A\omega$ is not guaranteed to belong to Θ, but could instead lie anywhere within $\mathbb{R}^D$. The simulator may consider θ as invalid if it is outside of Θ as shown in Fig.2. Moreover, just doing a rejection sampling does not always work because most of the points could be rejected for being invalid in some cases. *Random EMbedding Bayesian Optimization (REMBO) (Wang et al., 2016)* addressed this issue simply by projecting the point outside Θ to the nearest boundary point of Θ.

Variational Auto Encoder for Model Identification in the High Dimensional Space

An auto encoder is a neural network that learns to reconstruct the input by going through a latent space, which is in a lower dimensional space than the original input space(Vincent et al., 2010). It has shown to be very useful in unsupervised learning of low dimensional representations. A variational auto encoder (VAE) adds an additional constraint that the latent space follows a prior distribution, usually assumed to be Gaussian (Kingma and Welling, 2014). This additional 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 Bayesian optimization process, as shown in Fig. 3. Firstly, the VAE is trained with randomly sampled physical parameter data θ to learn a low dimension embedding α. Once the VAE is optimized, the decoder part is used to project the low dimensional α back to the original physical parameter space θ. Thus, the Bayesian optimization process as detailed in Algorithm 1 can

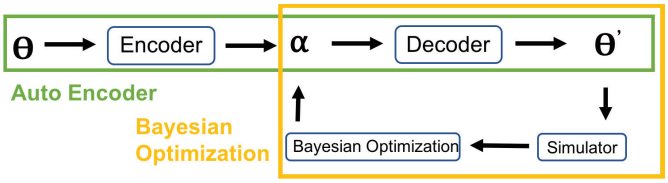

Figure 3: The auto encoder is trained first to learn the latent low dimensional embedding. Then Bayesian optimization is performed in this low dimensional space to search for the optimal parameter. The decoder is used to reconstruct the original 15 dimensional parameter in order to perform physical simulation.

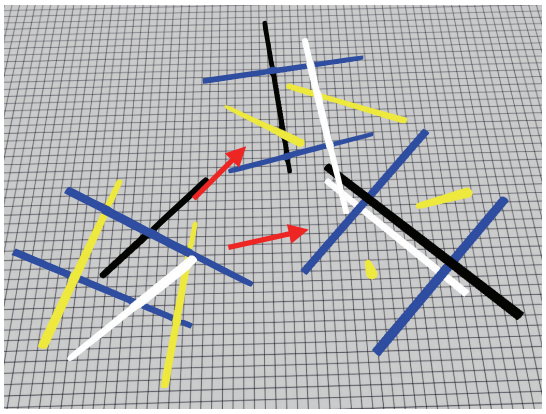

Figure 4: Simulation of the Tensegrity robot resulting in different states when executing the same control for different parameters.

be done efficiently in the low dimensional space. The decoder can be seen as a learned non-linear version of the projection matrix A in REMBO.

Experimental Results

Setup: This experiment aims to identify the 15 parameters of the T6 model of the Tensegrity SuperBall robot in NASA's Tensegrity Robotics Toolkit (NTRT). The complex dynamics and high dimensionality of the robot make this problem very hard. Fig. 4 shows an example of the different results of applying the same control to the robot with 1% difference in the rod length (one of the 15 parameters). In absence of access to the real robot, the default values of the T6 model in NTRT are used as ground-truth. The Guided Policy Search (GPS) algorithm (Levine and Abbeel, 2014) was used to discover fast trajectories of several flops through iterative exploration and refinement (GPS controller).

The Greedy Entropy Search (GES) method is compared against random search, where random values of the parameters are selected within the $\pm 10\%$ range. Nevertheless, it is well-known that Bayesian optimization in high dimensions is difficult due to the exponential growth of the search space. To deal with this issue, the two dimensionality reduction methods, REMBO and VAE are used to reduce the dimensionality of the parameter space from 15 to 5.

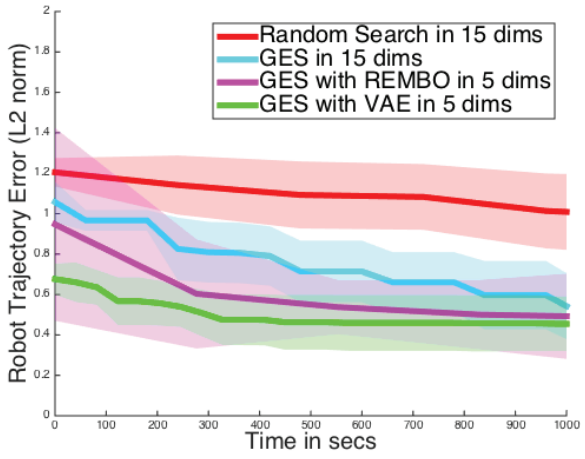

Figure 5: Test trajectory errors of different methods for the Tensegrity robot as a function of time budget for the parameter optimization process. Greedy Entropy Search in the 5-dimensional space using VAE achieves the lowest trajectory error, outperforming random search and Greedy Entropy Search in the original 15 dimensional space, as well as Greedy Entropy Search in the 5-dimensional space using REMBO.

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 $[0,1]$.

To train the VAE, 10,000 training trajectories are generated. These trajectories are generated by running the GPS controller in the simulator with different physical parameters and adding random noise of up to $\pm 10\%$ to the default parameter values. This means each trajectory is generated under slightly different physical parameters.

Results: Fig. 5 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, Bayesian optimization with Greedy Entropy Search outperformed random search. Further improvements are achieved by dimensionality reduction, making the search more efficient. Greedy Entropy Search in the 5-dimensional space using VAE achieves the lowest trajectory error, outperforming the method using REMBO. This shows that a learned better latent embedding enables more efficient parameter search in the Bayesian optimization process. A video showing examples of the Tensegrity robot locomotion can be found on https://youtu.be/lD31s0c_tqM.

Fig. 6 provides the errors for each of the parameter as a function of time budget for the parameter optimization process. Only the combination of Greedy Entropy Search with VAE achieves close to 1% error for all parameters. Some parameters may have 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.

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 minimize the number of needed experiments, a Greedy variant of Entropy Search is proposed, which is shown to be data efficient. To solve high-dimensional challenges, this work integrates Greedy Entropy Search with a projection to a lower-dimensional space through random embedding or learning a latent embedding utilizing variational 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 Bayesian Optimization'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 and control.

References

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

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

Bertsekas, D. P., and Tsitsiklis, J. N. 1996. *Neuro-Dynamic Programming*. Athena Scientific, 1st edition.

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

Calandra, R.; Seyfarth, A.; Peters, J.; and Deisenroth, M. P. 2016. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence (AMAI)* 76(1):5–23.

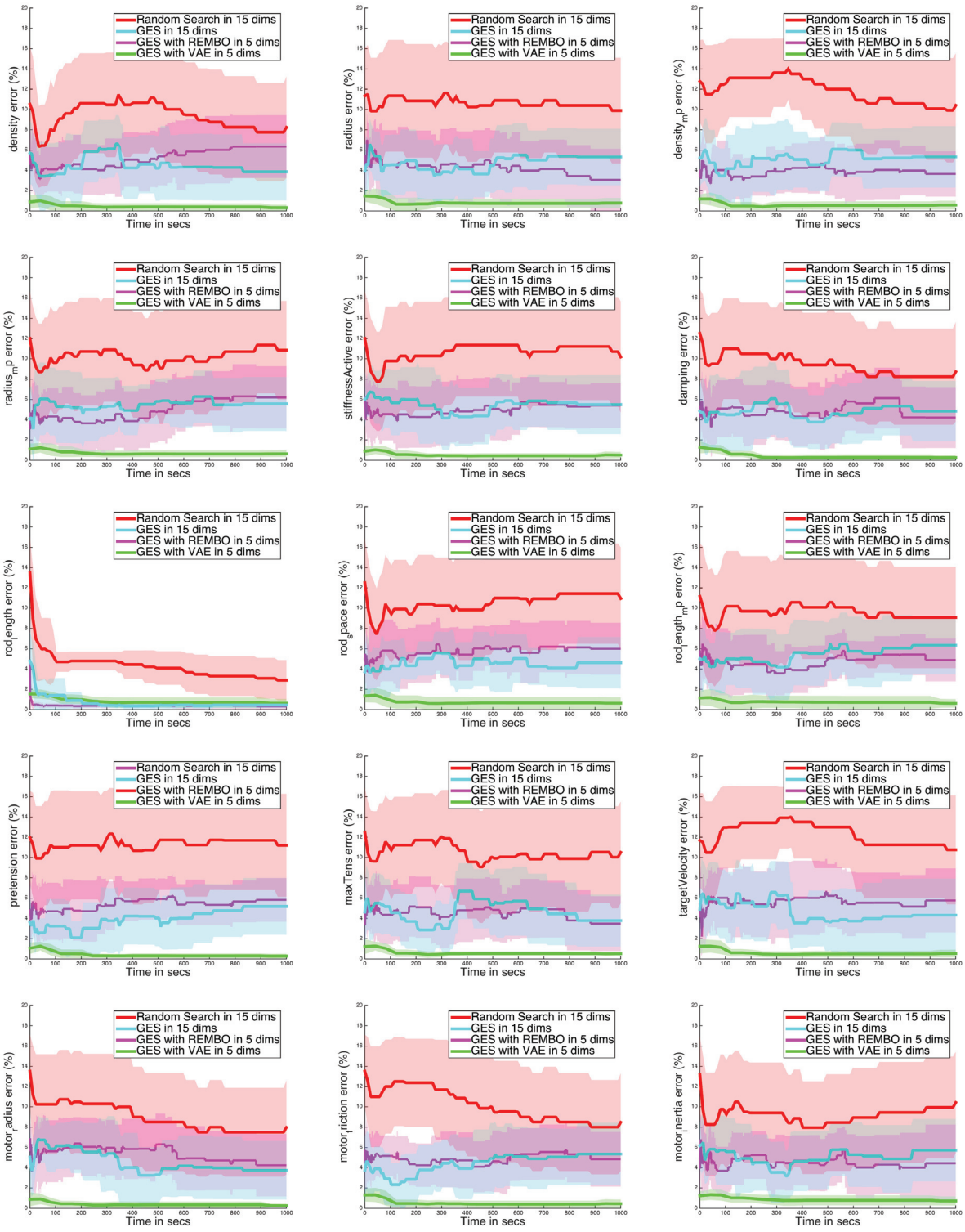

Figure 6: Each of the fifteen parameter error functions for the Tensegrity robot as a function of time budget for the parameter optimization process Greedy Entropy Search in the 5-dimensional space using VAE achieves the lowest error, which is less than 1% for all dimensions.

Caluwaerts, K.; Despraz, J.; Iscen, A.; Sabelhaus, A.; Bruce, J.; Schrauwen, B.; and SunSpiral, V. 2014. Design and control of compliant tensegrity robots through simulation and hardware validation. *Journal of The Royal Society Interface* 11(98).

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

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

Dogar, M.; Hsiao, K.; Ciocarlie, M.; and Srinivasa, S. 2012. Physics-Based Grasp Planning Through Clutter. In *Robotics: Science and Systems VIII*.

Erez, T.; Tassa, Y.; and Todorov, E. 2015. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ODE and physx. In *IEEE International Conference on Robotics and Automation, ICRA*, 4397–4404.

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

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

Hennig, P., and Schuler, C. J. 2012. Entropy Search for Information-Efficient Global Optimization. *Journal of Machine Learning Research* 13:1809–1837.

Kingma, D. P., and Welling, M. 2014. Auto-encoding variational bayes. In *ICLR*.

Kober, J., and Peters, J. R. 2009. Policy search for motor primitives in robotics. In *Advances in neural information processing systems*, 849–856.

Kober, J.; Bagnell, J. A. D.; and Peters, J. 2013. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*.

Levine, S., and Abbeel, P. 2014. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems (NIPS)*.

Lynch, K. M., and Mason, M. T. 1996. Stable pushing: Mechanics, control- lability, and planning. *IJRR* 18.

Marco, A.; Berkenkamp, F.; Hennig, P.; Schoellig, A. P.; Krause, A.; Schaal, S.; and Trimpe, S. 2017. 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*, 1557–1563.

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

Mirletz, B. T.; Park, I.-W.; Quinn, R. D.; and SunSpiral, V. 2015. Towards bridging the reality gap between tensegrity simulation and robotic hardware. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

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

NTRT. NASA tensegrity robotics toolkit (NTRT). https://ti.arc.nasa.gov/tech/asr/intelligent-robotics/tensegrity/NTRT/.

Peters, J.; Mülling, K.; and Altün, Y. 2010. Relative entropy policy search. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010)*, 1607–1612.

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

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

Scholz, J.; Levihn, M.; Isbell, C. L.; and Wingate, D. 2014. A Physics-Based Model Prior for Object-Oriented MDPs. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*.

Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R. P.; and de Freitas, N. 2016. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE* 104(1):148–175.

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

Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; and Manzagol, P.-A. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* 11(Dec):3371–3408.

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

Zhou, J.; Paolini, R.; Bagnell, J. A.; and Mason, M. T. 2016. A convex polynomial force-motion model for planar sliding: Identification and application. In *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016*, 372–377.