Deep Kernels for Optimizing Locomotion Controllers

Rika Antonova∗
Robotics, Perception and Learning, CSC
KTH Royal Institute of Technology
Stockholm, Sweden
antonova@kth.se

Akshara Rai∗
Robotics Institute
School of Computer Science
Carnegie Mellon University, USA
arai@andrew.cmu.edu

Christopher G. Atkeson
Robotics Institute
School of Computer Science
Carnegie Mellon University, USA
cga@cs.cmu.edu

Abstract: Sample efficiency is important when optimizing parameters of locomotion controllers, since hardware experiments are time consuming and expensive. Bayesian Optimization, a sample-efficient optimization framework, has recently been widely applied to address this problem, but further improvements in sample efficiency are needed for practical applicability to real-world robots and high-dimensional controllers. To address this, prior work has proposed using domain expertise for constructing custom distance metrics for locomotion. In this work we show how to learn such a distance metric automatically, without relying on domain experts. We use a neural network to learn an informed distance metric from data obtained in high-fidelity simulations. We conduct experiments on two different controllers and robot architectures. First, we demonstrate improvement in sample efficiency when optimizing a 5-dimensional controller on the ATRIAS robot hardware. We then conduct simulation experiments to optimize a 16-dimensional controller for a 7-link robot model and obtain significant improvements even when optimizing in perturbed environments. This demonstrates that our approach is able to enhance sample efficiency for two different controllers, hence is a fitting candidate for further experiments on hardware in the future.

1 Introduction

Bayesian Optimization (BO) is rapidly becoming a popular approach for optimizing controllers in robotics. It offers sample-efficient, black-box and gradient-free optimization, well suited for many problems in the field. Recently, some success has also been achieved when optimizing controllers directly on hardware [1], [2], [3]. Hence, this sample-efficient optimization framework has the potential to alleviate the need for manual tuning by experts, to a large extent. However, for high-dimensional controllers and challenging cost functions the performance of conventional BO often degrades. Without an informative prior, the number of data points required could be prohibitively expensive for hardware-only optimization. Hence, it seems ideal to exploit simulation to speed up learning, as proposed in [4] and [5]. These prior approaches, however, need extensive expert domain knowledge to define the problem-specific informed distance metric.

In this work we demonstrate how to construct an informed metric automatically, without relying heavily on domain experts. We propose to learn a distance metric with a neural network, utilizing data obtained from a high-fidelity simulator. This involves first running short simulations of a locomotion controller on a large grid of control parameters and recording the behavior of each set of parameters. The neural network then learns a mapping between input controller parameters and simulation output/behavior. We propose two ways of defining the target to be learned by the network. The first approach is based on the cost function that is to be optimized with BO on hardware, or a perturbed simulator. The second is cost-agnostic: learning to reconstruct a summary of robot trajectories obtained from simulation. This provides a useful re-parameterization: controller parameters that produce similar walking trajectory summaries are closer in this re-parameterized space.

∗Both of these authors contributed equally
In our first set of experiments we optimize a 5-dimensional controller on the ATRIAS robot hardware (Figure 1). We demonstrate that using cost-based kernel obtained with our approach outperforms using an uninformed kernel for BO. The setting we consider for ATRIAS experiments yields a proof-of-concept rather than a large-scale optimization problem. Nonetheless, we believe that its an important step towards optimizing locomotion policies for complex humanoid robots on hardware.

Prior Bayesian Optimization studies often used simpler robots. For example, [5] use snake robots, [3] use a hexapod, which are statically stable robots. [1] use a smaller biped with a finite-state-machine controller, which is not widely used. In contrast, ATRIAS is a complex humanoid system, which is not statically stable and likely to fall with unstable controllers. Moreover, our control framework is in line with most state-of-the-art robot controllers [6], [7]. Hence, results on our testbed can be transferred to other systems.

Our second set of experiments is on the Neuromuscular model [8]. We optimize a 16-dimensional controller for a 7-link robot model in simulation. Our approach of reconstructing trajectory summaries again yields a significant improvement over using uninformed kernels for BO. This is the case for both a smooth and a challenging non-smooth cost suggested in prior literature [9]. Hence the proposed approach offers a promising way to construct cost-agnostic kernels for BO automatically.

2 Background

2.1 Optimizing Bipedal Locomotion Controllers

Approaches to optimizing locomotion controllers range from manual tuning to fully automatic optimization. For complex controllers fully manual tuning is sometimes infeasible or excessively time consuming. In such cases, approaches like CMA-ES have been used to find points yielding good performance in simulation first [9]. A domain expert could then use such points as starting points to later manually adjust the parameters such that they are effective on hardware. Recently there has been significant interest in developing methods for automatic parameter optimization. Bayesian Optimization has been suggested as one of the promising approaches due its sample efficiency [1]. However, it still can take 30-40 samples to optimize a 4 dimensional controller.

Recent works proposed using simulation to aid learning on hardware, for example [2], [3], [4]. [2] propose adding noisy evaluations from simulation to the posterior used in Bayesian Optimization directly. The limitation is the need to carefully balance the influence of the samples obtained from simulation versus hardware. [3] tabulate best performing points versus their average score on a behavioural metric – average contact time of their hexapod system in simulation. This metric guides a trial-and-error learning method to quickly find behaviours that can compensate for damage of the robot. The search is conducted in behaviour space, and limited to pre-selected “successful” points from simulation. This helps make their search faster and potentially safer. However, if an optimal point was not pre-selected, BO cannot sample it during optimization, losing global optimality guarantees. “Best points” are cost-specific (the map needs to be re-generated for each cost) and problem specific, so expert-knowledge would be needed to apply the method to other systems. [4] propose a new distance metric using domain knowledge about bipedal locomotion. Short simulations are used to compute this metric for a large number of points (sets of control parameters), thus distinguishing points based on their behaviour in simulation, rather than the Euclidean distance between them. The method generalizes to different costs and maintains global optimality guarantees of BO. However, the distance metric is specifically designed for bipedal locomotion. Further domain-specific expertise would be needed to adapt this approach to other settings.

Another recent direction for learning locomotion controllers utilized deep neural networks. [10] formulate the problem of learning locomotion gaits as actor-critic Reinforcement Learning with neural networks as function approximators for policy and value functions. Such approaches, however, are
not data-efficient enough to support learning optimal parameters for locomotion controllers on real hardware. So in our work we are interested in combining sample efficiency of an approach like Bayesian Optimization with the flexibility and scalability of deep neural networks.

2.2 Background on Bayesian Optimization

Bayesian Optimization is a framework for sample-efficient global search ([11] gives a recent overview). The goal is to find \( x^* \) that optimizes a given objective function \( f(x) \), while executing as few evaluations of \( f \) as possible. In order to select the most promising points to evaluate next, an “acquisition” function is defined. One example is Expected Improvement (EI) function that selects \( x \) to maximize expected improvement over the value of the best result obtained so far [12]. EI requires defining the prior/posterior mean and variance of \( f \), and Gaussian Process is frequently used for this:

\[
f(x) \sim \mathcal{GP}(\mu(x), k(x_i, x_j)),
\]

Here \( \mu \) is a mean function and \( k \) defines a kernel. \( k(x_i, x_j) \) encodes the similarity of two inputs \( x_i, x_j \). The value of \( f(x_i) \) has a significant influence on the posterior value of \( f(x_j) \) if \( x_i, x_j \) have high similarity according to the kernel. Squared Exponential kernel is widely used:

\[
k_{SE}(x_i, x_j) = \sigma^2 \exp \left( -\frac{1}{2l^2} \|x_i - x_j\|^2 \right),
\]

where hyperparameters: \( \sigma^2, l^2 \) are variance and a vector of length scales respectively. It is customary to adjust these automatically during optimization to learn the overall variance and how quickly \( f \) varies in each input dimension.

Gaussian Process conditioned on evidence represents a posterior distribution for \( f \). After evaluating \( f \) at points \( x_1, \ldots, x_s \) the predictive posterior \( P(f_{t+1}|x_1, t, y, x_{t+1}) \sim \mathcal{N}(\mu_t(x_{t+1}), cov_t(x_{t+1})) \) can be computed in closed form with mean and covariance:

\[
\mu_t(x_{t+1}) = k^T [K + \sigma^2_{\text{noise}}]^{-1} y \quad \text{and} \quad cov_t(x_{t+1}) = k(x_{t+1}, x_{t+1}) - k^T [K + \sigma^2_{\text{noise}}]^{-1} k,
\]

where \( k \in \mathbb{R}^s \), with \( k_i = k(x_{t+1}, x_i) \); \( K \in \mathbb{R}^{s \times s} \) with \( K_{ij} = k(x_i, x_j) \); \( I \) is an identity \( \mathbb{R}^{s \times s} \), and \( y \) is a vector of values obtained after evaluating \( f(x_1), \ldots , f(x_s) \), assuming Gaussian noise with variance \( \sigma^2_{\text{noise}} \); \( y_i = f(x_i) + \epsilon \mathcal{N}(0, \sigma^2_{\text{noise}}) \). More details can be found in [13].

3 Problem Formulation

In this work we aim to automatically optimize parameters of controllers for bipedal locomotion with respect to some commonly used cost functions. We assume that for a \( d \)-dimensional controller there is a bounded region of interest (a hypercube) defined by low/high limits on the values of controller parameters: \( x \in [x_{low}, x_{high}] \subset \mathbb{R}^d \). Some parts of this region contain points corresponding to parameter sets of the controller that yield the desired walking behavior. Such regions might comprise a large part of the space with numerous local optima, or might comprise only a small part of the space (e.g. less than 1%). In other words: we do not impose any overly restrictive assumptions on the space of controller parameters, local/global optima, or structure and properties of the cost functions of interest.

The first setting we consider is the case of optimizing 5-dimensional parameters for Raibert locomotion controller of the ATRIAS robot similar to [14], [15] and [16]. This controller has a Raibert-like foot placement policy [17]. It uses a linear feedback law operating on horizontal speed and displacement of the center of mass (CoM) to determine a desired foot touch down point:

\[
x_p = k(v - v_{tgt}) + C \cdot d + 0.5 \cdot v \cdot T
\]

Here, \( x_p \) is the desired location for the end of swing; \( v \) is the speed of the CoM; \( k \) is a feedback term that regulates \( v \) towards the target speed \( v_{tgt} \); \( C \) is a constant and \( d \) is the distance between the stance leg and the CoM; \( T \) is the step time and the term \( 0.5 \cdot v \cdot T \) is a feedforward term, similar to [17]. The swing foot trajectory is defined as a 5th order spline ending at \( x_p \).

In stance, we regulate both the torso pitch and CoM height to maintain constant desired values:

\[
F_x = K_{p\theta}(\theta_{des} - \theta) + K_{dz}(\dot{\theta}_{des} - \dot{\theta}) \quad F_z = K_{pz}(z_{des} - z) + K_{dz}(\dot{z}_{des} - \dot{z})
\]

These desired forces are sent to an inverse dynamics solver to return the corresponding joint torques that produce these desired ground reaction forces.
Our 5-dimensional controller consists of \([k, C, T, K_{pt}, K_{dt}]\). Other parameters can also be included, but the performance is not sensitive to them. A significant part of the parameter space of the 5-dimensional Raibert controller contains well-performing points (1/6 on hardware, 1/4 in simulation). To demonstrate applicability to a more challenging setting and a higher-dimensional controller we also experiment with a Neuromuscular model for control [9]. To facilitate comparison of our results with prior work in [4], we optimize over a 16-dimensional subspace of controller parameters. Since the Neuromuscular model has not yet been fully adapted to work on ATRIAS in hardware, for this setting we evaluate our work on a 7-link planar model [18]. We collect all the training data from simulations run on flat ground for a pre-defined grid of model parameters. We then generate a set of model disturbances for each link of the robot, perturbing the mass, inertia and center of mass location up to \( \pm 15\% \) of the original value. In addition, instead of walking on flat ground, we use a set of randomly generated rough ground profiles with step height of up to \( \pm 6 \) cm. We conduct the evaluation of our approaches on these perturbed models of the robot to create a simulated mismatch between simulation and hardware.

4 Proposed Approach

The aim of our approach is to automatically learn an informed kernel for optimizing bipedal locomotion controllers with Bayesian Optimization. We run short simulations for a range of parameter sets and record the resulting costs from the same cost function as that used in Bayesian Optimization. Costs obtained during short simulations serve as approximate indicators of the quality of the controller parameters. Our idea is to use these costs obtained from simulation to generate an informed distance metric for the Bayesian Optimization kernel.

4.1 Regression with Implicitly Asymmetric Loss

We consider a cost function focused on matching the desired walking speed and heavily penalizing falls:

\[
cost_{atrias} = \begin{cases} 
100 - x_{fall}, & \text{if fall} \\
10 \cdot \left\| v_{tgt} - v_{actual} \right\|^2, & \text{if walk}
\end{cases}
\]  

(1)

where \( x_{fall} \) is the distance travelled before falling, \( v_{tgt} \) is the target velocity and \( v_{actual} \) is the vector containing actual velocities of the robot. This kind of cost function is of interest because it helps easily distinguish points that walk from points that fall. Similar costs have been considered in prior work when optimizing locomotion controllers [9, 4].

Figure 2 shows a scatter plot of applying cost from Equation 1 to simulations of Raibert controller for the ATRIAS robot as introduced in Section 3. For visualization we restrict attention to a 2-dimensional subspace of the parameter space. We pick a well-performing set of parameters (in 5D), then vary the first two dimensions to obtain a 2D subspace. The challenge comes from the fact that the boundary between the well-performing (blue) and poorly performing (yellow) parameters is discontinuous. This is a typical landscape for bipedal systems, where a controller that makes the robot fall is much worse than one than walks, and the boundary is extremely sharp. While there can be variations to how costs are structured among stable walking points – efficiency vs speed vs distance covered, parameters that fall are much worse. Fitting such cost function with regression could be difficult. Learning to reconstruct the boundary exactly using the training set might result in overfitting and poor performance on the test set. Trying to fix this by applying regularization is likely to result in high loss and uncertainty about the points close to the boundary. This is particularly problematic if poorly performing points lie close to some of the most promising regions of the parameter space, which is the case for the setting we consider.

We propose to use a transformation of the cost as the target for the supervised learning. Our approach is to train a deep neural network to reconstruct a reflected shifted softplus function of the cost:

\[
\text{score}_{NN} = \xi \left( \text{cost}_{walk} - f_{sim}(x) \right)
\]  

(2)
Here $\zeta$ is a softplus function: $\zeta(u) = \ln(1 + e^u)$, $\text{cost}_{walk}$ is the average cost for the parameter sets that walk during short simulations, $f_{sim}(x)$ is the cost computed by the simulator for vector $x$ of controller parameter values. Using this transformation yields a “score” function such that parameter sets which produce poor results in simulation are mapped to values close to zero. With this, the differences in scores of the poorly performing parameter sets become small or zero. In contrast, the differences in scores of the parameter sets yielding potentially promising results remain proportional to the difference in the corresponding costs. Figure 3 gives a visualization of this transformation.

Cost transformation in Equation 2 serves to essentially create an asymmetric loss for neural network training. This asymmetric loss is minimized when the promising (low-cost, high-score) points are reconstructed correctly. For the poorly performing (high-cost, low-score) points, it only matters that the output of the neural network is close to zero. Such asymmetric loss can be interpreted as implementing a hybrid of regression and “soft” classification. The regression aspect aims to fit the promising points which correspond to parameters yielding walking behaviors. The “soft” classification causes an increase in the loss only if a poorly performing point is “mis-classified” as well-performing.

We also apply L1 loss instead of the usual L2 loss when training the neural network. With this, errors in reconstructing points on the boundary contribute only linearly to the overall loss. This helps achieve a better fit of the stable parts of the parameter space, instead of focusing on the boundary.

We utilize the reconstructed transformed costs to define $\text{asymNN}$ kernel for Bayesian Optimization:

$$k_{\text{asymNN}}(x_i, x_j) = \sigma_k^2 \exp \left( -\frac{1}{2\tau} \| \text{score}_{NN}(x_i) - \text{score}_{NN}(x_j) \|^2 \right),$$

with hyperparameters $\sigma_k, l$ as described in Section 2.2. The proposed approach is able to clearly separate the unpromising part of the parameter space. Under the resulting distance metric poorly performing sets of parameters are close together and far from well-performing ones.

### 4.2 Reconstructing Cost-agnostic Trajectory Summaries

While learning a distance metric from the cost could be effective for a wide variety of problems, frequently there is a need for a cost-agnostic approach. Such cases arise when the data from the simulator is computationally expensive to collect and we need to change the cost function. Different tasks might call for slightly different cost functions. For example, high energy consumption could be penalized if energy use needs to be restricted; robustness of the walk could be emphasized if only stable walking is acceptable; if achieving the desired speed is the most important factor – then the cost might instead only reflect how well the desired speed is maintained. For such cases we propose to train a neural network to reconstruct summaries of trajectories that are cost-agnostic, then utilize these trajectory summaries for constructing kernel distance metric.

In most cases, a summary of trajectory information contains all the pertinent information. Hence, we focus on collecting the following aspects of the simulated trajectories: walking time (time before falling), energy used during walking, position of the torso, angle of the torso, coordinates of the center of mass at the end of the short simulation runs. These summaries of simulated trajectories are collected for a range of controller parameters and comprise the training set for the neural network to fit (input: $x$ – a set of control parameters; output: $\text{traj}_x$ – the corresponding trajectory summary obtained from simulation). The outputs of the (trained) neural network offer the reconstructed/approximate trajectory summaries: $f_{NN}(x) = \hat{\text{traj}}_x$, where $x$ is the input controller parameters, and $\hat{\text{traj}}_x$ is the corresponding reconstructed trajectory summary. These are then used to define a distance metric for the kernel for Bayesian Optimization:

$$k_{\text{trajNN}}(x_i, x_j) = \sigma_k^2 \exp \left( -\frac{1}{2\tau} \| f_{NN}(x_i) - f_{NN}(x_j) \|^2 \right).$$

The general concept of utilizing trajectory data to improve sample efficiency of BO has been proposed before, for example in [19]. However, prior work assumed obtaining trajectory data is possible every time kernel values $k(x, x_j)$ need to be evaluated. This is not the case in our setting. Trajectory summaries are initially obtained via costly high-fidelity simulations, and it would be infeasible to compute trajectory information via simulation during BO. Hence, our approach is to train a neural
network to learn reconstructing trajectory summaries first. Running a forward pass of the neural network is a relatively inexpensive operation, hence reconstructed/approximate trajectory summaries can be quickly obtained during BO whenever $k_{\text{trajNN}}(x_i, x_j)$ needs to be computed. Note that the trajectory information we extract is generic and can be applied to other problems without requiring in-depth domain knowledge. When applying this approach to a new domain, the strategy would be to include trajectory information used to compute cost functions that are of interest/relevance in the domain. For example, for a manipulator, the coordinates of end-effector(s) could be recorded at relevant points. In contrast, information extracted in [4], [3] is very specific to locomotion.

5 Experimental Results

In this section we describe our experiments with cost-based and trajectory-based kernels. We first consider the setting of optimizing a 5-dimensional controller for the ATRIAS robot. We show that the cost-based kernel is able to improve sample efficiency over standard Bayesian Optimization. We present hardware experiments to demonstrate that our kernel allows obtaining a set of parameters close to optimal on the second trial. We then discuss simulation experiments with a 16 dimensional controller that utilizes a Neuromuscular model [9]. These experiments show that our trajectory-based kernel is able to significantly outperform standard Bayesian optimization for a higher-dimensional controller even when a sharply discontinuous cost is used during optimization.

5.1 Experiments with Raibert controller on the ATRIAS robot

For our experiments on the ATRIAS robot we used a high-fidelity ATRIAS simulator [16] to generate the kernel. We did an initial analysis of the performance of our approach in simulation, followed by hardware experiments. As described in section 4, we constructed a distance metric for a kernel used in Bayesian Optimization by training a neural network to reconstruct cost obtained from short simulations. We created a sobol grid on the input parameter space with 20K points and ran short 3.5 second simulations on each of the corresponding 20K parameter sets to compute the costs. We then used a fully connected network with 4 hidden layers (128, 64, 16, 4 units) with L1 loss to reconstruct the transformation of the cost described in section 4.1.

In Figure 4 we first compare the performance of BO that used our neural network kernel ($\text{asymNN}$) versus using a standard Squared Exponential kernel ($\text{SE}$) in simulation. For these experiments we used the cost in Equation 1, Section 4.1 with a target velocity of $1 \text{m/s}$. Simulations with cost less than 50 yielded walking behavior, while cost less than 20 resulted in a stable walk close to the desired speed. BO with $\text{asymNN}$ kernel was able to reliably find points corresponding to stable walking behavior in only 8 trials. In contrast, BO with $\text{SE}$ kernel did not find stable walking solutions in the first 20 trials reliably. We also compare with a recently proposed Determinants of Gait ($\text{DoG}$) kernel [4] that utilized domain knowledge to construct an informed kernel for BO of locomotion controllers. $\text{asymNN}$ is able to closely match the performance of $\text{DoG}$ in this setting after 8 trials.

After experiments in simulation suggested that $\text{asymNN}$ kernel can yield a significant improvement in sample efficiency of BO, we conducted a set of experiments on the ATRIAS robot. We completed 6 sets of runs of BO: 3 using $\text{asymNN}$ kernel and 3 using a standard $\text{SE}$ kernel with 10 trials each, leading to a total of 60 hardware experiments. Since ATRIAS walks around a rather short boom in 2D, walking at high speeds needs a lot of torque from the robot motors. This means higher lateral forces between the robot and the boom, which do not affect our direction of motion but can lead to a lot of internal forces, eventually breaking the robot. So, in our first attempt, we tried to start with lower speeds of $0.4 \text{m/s}$ so that we could do hardware experiments and analyze the validity of our approach on hardware without breaking the robot too often. Setting a lower target speed yielded a
problem with more stable walking points: they comprised approximately \( \frac{1}{6} \) of the parameter space. So we anticipated it would be challenging to improve over BO with SE kernel, since it was already able to find stable walking solutions after only 3-4 trials. Another feature of the asymNN kernel was that it was biased towards sampling points that walk in simulation. Hence, out of 10 trials, it is more likely to sample stable points even in hardware (Figure 6b), as compared to SE. This is desirable as stable points are less likely to break the robot.

Figure 6a shows the performance of BO with SE versus asymNN kernel. SE obtains a stable walking solution on the 3rd trial in one run, and on the 4th trial in the two other runs. asymNN kernel is able to find the best-performing set of parameters on the second trial in each of the 3 runs. This confirms that using asymNN kernel offers an improvement over using SE kernel in this setting. We suggest that asymNN reliably selects an excellent point on the 2nd trial because such points lie far from poorly performing subspace of parameters (under the distance metric constructed with asymNN).

While in our hardware setup most methods are likely to sample walking points within 10 trials, we believe our experimentation is an important step towards optimizing locomotion policies for complex humanoid robots. Bayesian Optimization studies in the past have also used real robot hardware, for example, [1], [3] and [5]. However, [5, 20, 3] used snake robots, which are statically stable, making discontinuities in the cost function landscape less likely and in turn making the optimization easier. On the other hand, ATRIAS is a complex humanoid system which is likely to fall with unstable controllers. [1] use a walking robot similar to ours. However, their controller parametrization is very different, and not widely used, unlike our inverse dynamics and force-based controller which is more modern and state-of-the-art [7], [21], [6]. Hence, even if our policy parametrization was chosen so that steady walking points could be found in a few trials, our testbed is fairly complex and our problem formulation is widely applicable.

5.2 Experiments with the Neuromuscular Model

In section 4.2 we introduced a cost-agnostic approach for constructing an informed kernel from simulations. Our approach is to train a neural network to reconstruct trajectory information. Here we describe our experiments with 16-dimensional controller of the Neuromuscular model [4]. We created a grid of 100K points in the input parameter space and ran short 5 second simulations on each of the corresponding 100K parameter sets to collect the trajectory summaries. We then used a fully connected network with 4 hidden layers (512, 128, 32 units) with L1 loss to reconstruct the summaries of the trajectories (as described in section 4.2). This transformation induced by the neural network was used as a re-parameterization from the input space of 16-dimensional controller parameters to 8-dimensional space of trajectory summaries. These 8-dimensional outputs of the neural network define kernel distances in the informed kernel (trajNN). All experiments were conducted on perturbed models, as described in Section 3.

Figure 7 illustrates our experiments comparing using trajNN versus using Squared Exponential (SE) kernel for BO. To analyze the performance of trajNN on different cost functions we conducted experiments on two different costs suggested in prior literature. The first cost promotes walking further and longer before falling, while penalizing deviations from the target speed [4]:

\[
\text{cost}_{\text{smooth}} = 1/(1 + t) + 0.3/(1 + d) + 0.01(s - s_{\text{tgt}}),
\]

where \( t \) is seconds walked, \( d \) is the final hip position, \( s \) is mean speed and \( s_{\text{tgt}} \) is the desired walking speed (1.3m/s in our case). The second cost function is a simplified version of the cost used in [9].
penalizes falls explicitly, and encourages walking at desired speed and with lower cost of transport:

$$\text{cost}_{\text{non-smooth}} = \begin{cases} 
300 - x_{\text{fall}}, & \text{if fall} \\
100||v_{\text{avg}} - v_{\text{tgt}}|| + c_{tr}, & \text{if walk}
\end{cases}$$

where $x_{\text{fall}}$ is the distance covered before falling, $v_{\text{avg}}$ is the average speed of walking, $v_{\text{tgt}}$ is the target velocity, and $c_{tr}$ captures the cost of transport.

Figure 7a shows that trajNN offers a significant improvement in sample efficiency when using cost_{smooth} during optimization. Points with cost less than 0.2 correspond to robust walking behavior. With trajNN, more than 90% of runs obtain walking solutions after only 25 trials. In contrast, using SE requires more than 90 trials for such success rate. The performance of trajNN matches that of a DoG kernel from prior work [4]. This is notable, since trajNN is learned automatically, whereas DoG kernel is constructed using domain expertise. Figure 7b shows that trajNN also provides a significant improvement when using the second cost. Points with cost less than 100 correspond to walking. With trajNN, 70% of the runs find walking solutions after 100 trials. In contrast, optimizing non-smooth cost is very challenging for BO with SE kernel: a walking solution is found only in 1 out of 50 runs after 100 trials.

The difference in performance on the two costs is due to the nature of the two costs. If a point walks some distance $d$, Equation 3 penalizes points according to $1/d$ and Equation 4 penalizes them according to $-d$. This results in a much steeper fall in cost with the first cost, and BO starts to exploit around points that walk some distance, quickly finding points that walk forever. However, with the second cost, BO continues to explore, and sometimes does not find walking points even in 100 trials. Exploitative methods might be better suited for higher dimensional problems, as compared to exploratory methods, in our experience.

trajNN kernel might be preferable to a cost-based kernel not only for settings with multiple costs. For some higher dimensional problems reduction to a 1-dimensional space in the kernel could be undesirable. So, while 1-dimensional cost-based kernel could yield highly sample-efficient optimization for lower dimensional problems, a higher-dimensional kernel like trajNN could provide more flexibility without compromising sample-efficiency for higher dimensional problems.

6 Conclusion

In this work we proposed constructing custom kernels for Bayesian Optimization of locomotion controllers without relying heavily on domain experts. We optimized a 5-dimensional controller on the ATRIAS robot and showed that our cost-based kernel offered an improvement over using an uninformed kernel. We also proposed a cost-agnostic alternative: an informed metric based on learning to reconstruct trajectory information. Experiments with a 16-dimensional Neuromuscular controller in simulation showed that our cost-agnostic approach offers a significant improvement over using an uninformed kernel. Hence this is a promising approach for automatically constructing cost-agnostic kernels for Bayesian Optimization.
References

[1] Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian Optimization for Learning Gaits Under Uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1-2):5–23, 2016.

[2] Alonso Marco, Felix Berkenkamp, Philipp Hennig, Angela P Schoellig, Andreas Krause, Stefan Schaal, and Sebastian Trimpe. Virtual vs. Real: Trading Off Simulations and Physical Experiments in Reinforcement Learning with Bayesian Optimization. *arXiv preprint arXiv:1703.01250*, 2017.

[3] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots That Can Adapt Like Animals. *Nature*, 521(7553):503–507, 2015.

[4] Rika Antonova, Akshara Rai, and Christopher G Atkeson. Sample Efficient Optimization for Learning Controllers for Bipedal Locomotion. In *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pages 22–28. IEEE, 2016.

[5] Matthew Tesch, Jeff Schneider, and Howie Choset. Using Response Surfaces and Expected Improvement to Optimize Snake Robot Gait Parameters. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1069–1074. IEEE, 2011.

[6] Siyuan Feng, Eric Whitman, X Xinjilefu, and Christopher G Atkeson. Optimization-based Full Body Control for the DARPA Robotics Challenge. *Journal of Field Robotics*, 32(2):293–312, 2015.

[7] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based Locomotion Planning, Estimation, and Control Design for the Atlas Humanoid Robot. *Autonomous Robots*, 40(3):429–455, 2016.

[8] Hartmut Geyer and Hugh Herr. A Muscle-reflex Model that Encodes Principles of Legged Mechanics Produces Human Walking Dynamics and Muscle Activities. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(3):263–273, 2010.

[9] Seungmoon Song and Hartmut Geyer. A Neural Circuitry that Emphasizes Spinal Feedback Generates Diverse Behaviours of Human Locomotion. *The Journal of Physiology*, 593(16):3493–3511, 2015.

[10] Xue Bin Peng, Glen Berseth, and Michiel van de Panne. Terrain-adaptive Locomotion Skills Using Deep Reinforcement Learning. *ACM Transactions on Graphics (TOG)*, 35(4):81, 2016.

[11] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[12] J Mockus, V Tiesis, and A Zilinskas. Toward Global Optimization, volume 2, chapter Bayesian Methods for Seeking the Extremum. 1978.

[13] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning* (Adaptive Computation and Machine Learning). The MIT Press, 2005. ISBN 026218253X.

[14] William C Martin, Albert Wu, and Hartmut Geyer. Experimental Evaluation of Deadbeat Running on the ATRIAS Biped. *IEEE Robotics and Automation Letters*, 2(2):1085–1092, 2017.

[15] Christian Hubicki, Andy Abate, Patrick Clary, Siavash Rezazadeh, Mikhail Jones, Andrew Peekema, Johnathan Van Why, Ryan Domres, Albert Wu, William Martin, et al. Walking and Running with Passive Compliance: Lessons from Engineering a Live Demonstration of the ATRIAS Biped. *IEEE Robotics and Automation Magazine*, 2(4.1):4–1, 2016.
[16] William C Martin, Albert Wu, and Hartmut Geyer. Robust Spring Mass Model Running for a Physical Bipedal Robot. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 6307–6312. IEEE, 2015.

[17] Marc H Raibert. *Legged Robots that Balance*. MIT press, 1986.

[18] Nitish Thatte and Hartmut Geyer. 16D Simulator for Neuromuscular Models for Biped Locomotion. Code available from https://github.com/nthatte/Neuromuscular-Transfemoral-Prosthesis-Model, 2011.

[19] Aaron Wilson, Alan Fern, and Prasad Tadepalli. Using Trajectory Data to Improve Bayesian Optimization for Reinforcement Learning. *The Journal of Machine Learning Research*, 15(1):253–282, 2014.

[20] Daniel J Lizotte, Tao Wang, Michael H Bowling, and Dale Schuurmans. Automatic Gait Optimization with Gaussian Process Regression. In *IJCAI*, volume 7, pages 944–949, 2007.

[21] Alexander Herzog, Nicholas Rotella, Sean Mason, Felix Grimminger, Stefan Schaal, and Ludovic Righetti. Momentum Control with Hierarchical Inverse Dynamics on a Torque-controlled Humanoid. *Autonomous Robots*, 40(3):473–491, 2016.