

Combining Simulations and Real-robot Experiments for Bayesian Optimization of Bipedal Gait Stabilization

Diego Rodriguez*, André Brandenburger*, and Sven Behnke

Autonomous Intelligent Systems, Computer Science, Univ. of Bonn, Germany
 {rodriguez, behnke}@ais.uni-bonn.de, andre.brandenburger@uni-bonn.de
<http://ais.uni-bonn.de>

Abstract. Walking controllers often require parametrization which must be tuned according to some cost function. To estimate these parameters, simulations can be performed which are cheap but do not fully represent reality. Real-robot experiments, on the other hand, are more expensive and lead to hardware wear-off. In this paper, we propose an approach for combining simulations and real experiments to learn gait stabilization parameters. We use a Bayesian optimization method which selects the most informative points in parameter space to evaluate based on the entropy of the cost function to optimize. Experiments with the igus[®] Humanoid Open Platform demonstrate the effectiveness of our approach.

1 Introduction

Walking is a crucial task for legged robots. The state-of-the-art walking controllers and generators typically require a fine-tuned parametrization that due to its

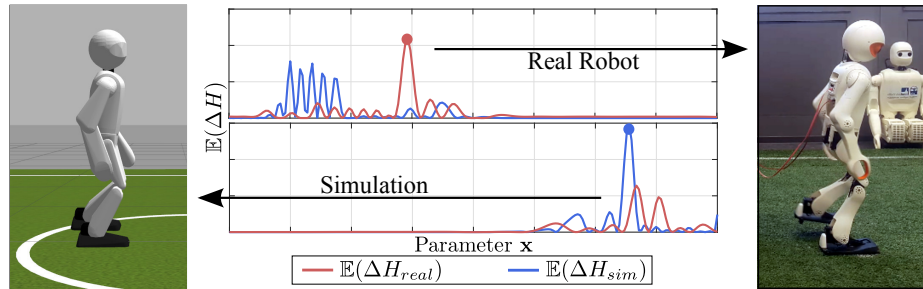


Fig. 1. Combining multiple sources of information for learning gait parameters $\mathbf{x}$. Based on the largest relative entropy $\mathbb{E}(\Delta H)$ of a cost function J , a walking sequence is performed in simulation or with the real robot. The number of walking sequences performed with the real robot is reduced by using simulations.

*Both authors contributed equally.

complexity is determined by experts. This puts a constraint on the applicability of these methods. They can be used mainly by the people who designed them. In recent years, learning approaches have been proposed in order to reduce the amount of work and expert knowledge required to tune these methods [7, 18]. This implies thus to perform experiments to estimate parameters. Consequently, sample-efficient learning approaches are required in order to reduce the hardware wear-off induced by the experiments. One way to reduce the need for real-robot experiments is to use simulations, which are cheaper to perform but do not fully represent reality. This point is particularly relevant for low-cost robots whose hardware is not as precise as expected. In this paper, we propose a method to combine simulations and real-robot experiments to optimize gait parameters, specifically to learn activation values of corrective actions that act on top of an open-loop bipedal gait generator. The optimization uses a state-of-the-art sample-efficient Bayesian method which selects the most informative points in parameter space to evaluate based on the entropy of a cost function.

2 Related Work

Several learning methods have been used to optimize manipulation or locomotion parameters [1, 7, 9, 12, 18, 20]. Most of the methods are based on Bayesian optimization—due to its high sample efficiency. Deisenroth et al. [9], for example, developed a Bayesian approach to tune a cart-pole system, whereas Berkenkamp et al. [6] proposed to use Bayesian optimization to safely tune robotic controllers for quadrotors. Moreover, Marco et al. [16] combined Bayesian optimization with optimal control to tune LQR regulators. In contrast to these approaches, Akrouir et al. [1] suggested to direct the optimization process by using a search distribution, however, the optimization loses expressibility on a global scope since it only optimizes locally. Even though these methods take advantage of the sample efficiency of the Bayesian optimization, the robotic hardware is worn off unnecessarily in the learning process, especially in the initial stages where the controllers do not possess any prior knowledge.

Specifically for the problem of gait parameter optimization, Calandra et al. [7] suggested a Bayesian optimization in order to replicate a given target trajectory for a bipedal robot. Based on only real experiments, the algorithm was able to find a stable gait. Because only the real hardware was used for all experiments, we expect a considerable wear-off of the robot. On the other hand, Heijmink et al. [12] proposed a method to learn gait parameters and impedance profiles in simulation for a quadruped robot. This was accomplished by using the PI^2 algorithm with a cost function consisting of speed tracking, energy consumption, joint limits, and torques. Although the results were validated in the real hardware, no transfer between simulation and the real hardware was addressed. Similarly, Hengst et al. [13], use a simulator to learn gait parameters that will be tested in the real robot. This is a reinforcement learning approach that learns the ankle joint position of the stance leg and the placement of the swing foot.

There exist several approaches that have transferred knowledge gathered in simulation to real robotic platforms. Farchy et al. [10], for example, learn several dynamic parameters of a simulator in order to get a similar performance compared with real-robot experiments. This approach was extended by Hanna and Stone [11] by learning the dynamics of the simulator using the differences of the actions between the real world and the simulation. The walk velocity of the NAO robot was increased by 43% starting from a state-of-the-art walk engine. Nevertheless, a human expert was required to select the appropriate parameters to be learned. Additionally, Cutler and How [8] uses a simulator to learn a nonparametric prior that will parametrize a learning algorithm that acts directly on the real platform, in other words, there is an one-step transfer from simulation to the real robot. In a recent work, Rai et al. [18] uses simulations to build a lower-dimensional space which is later used to learn gait parameters on the real hardware. Although this approach was able to achieve good results for a 9-dimensional controller with only 20 iterations, it requires—due to its informed kernel structure—a large amount of precomputed simulator data.

3 Preliminaries

3.1 Gaussian Process Regression

Given a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i) | i = 1, \dots, n\}$ of n observations, a Gaussian Process attempts to infer the relationship between the inputs and the targets given some *prior* knowledge. The observations are assumed to be corrupted by normally distributed noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$, such that

$$y = f(\mathbf{x}) + \epsilon \quad (1)$$

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}_i, \mathbf{x}_j)), \quad (2)$$

where $\mu(\mathbf{x})$ is the prior mean, which can be uniform, and $k(\mathbf{x}_i, \mathbf{x}_j)$ is the kernel also called the covariance function. Using a kernel allows us to transform the input space into a higher-dimensional feature space such that a non-linear map from the input vector $\mathbf{x}$ to the function value $f(\mathbf{x})$ can be inferred. The kernel models the uncertainty of the mean estimate and encodes how similar $f(\mathbf{x})$ is expected to be for two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$. A high value of $k(\mathbf{x}_i, \mathbf{x}_j)$ would mean that the posterior value of $f(\mathbf{x}_j)$ is significantly influenced by the value of $f(\mathbf{x}_i)$. Note that using kernels we do not need to know the shape of the corresponding feature space, because only the inner products in the input space are required.

3.2 Bayesian Optimization

Bayesian optimization is a gradient-free sample-efficient framework that optimizes a cost function $f(\mathbf{x})$ using statistical models. Its goal is to find a global optimum of a cost function which is typically expensive to evaluate. In our case, it would imply to wear-off the hardware of the robot by performing walking experiments.

This optimum is found by minimizing a posterior mean function. Often, the mean and covariance of $f(\mathbf{x})$ are described by a Gaussian Process. The points to evaluate $f(\mathbf{x})$ are selected through an *acquisition function*, which also trades off exploration and exploitation, i.e., to select promising points where the optimum might be and to reduce the uncertainty about $f(\mathbf{x})$.

A prominent example of those acquisition functions is *Entropy Search* (ES) [14]. ES is based on the expected change of entropy $\mathbb{E}[\Delta H(\mathbf{x})]$, such that the point to evaluate in the next iteration is the one that offers most information (highest entropy change). The location of the minimum is approximated by a non-uniform grid, which, upon convergence, will be peaked around the actual minimum. The acquisition function for ES is defined as:

$$\mathbf{x}_{t+1} = \arg \max_{\mathbf{x} \in X} (\mathbb{E}[\Delta H(\mathbf{x})]). \quad (3)$$

The approximations to make ES computationally tractable can be found in [14].

3.3 Multi-Fidelity Entropy Search

Marco et al. [15] extends the ES algorithm to integrate multiple sources of information (Fig. 2). The resulting method is called *Multi-Fidelity Entropy Search* (MF-ES) and typically trades off real experiments with simulations. MF-ES optimizes the cost function

$$J_{real}(\mathbf{x}) = J_{sim}(\mathbf{x}) + \epsilon_{sim}(\mathbf{x}) \quad (4)$$

over a parameter set $\mathbf{x} \in X$. The key idea is to model the cost on the physical system J_{real} as the cost in simulation $J_{sim}(\mathbf{x})$ plus a systematic error $\epsilon_{sim}(\mathbf{x})$. This error ϵ_{sim} can be a complex transformation, which is learned by the Bayesian optimization. MF-ES defines two kernel functions k_{sim} and k_{ϵ} , which model the cost on simulation and the difference to the real experiments, respectively. In this manner, the kernel is expressed as:

$$k(\mathbf{a}_i, \mathbf{a}_j) = k_{sim}(\mathbf{x}_i, \mathbf{x}_j) + k_{\delta}(\delta_i, \delta_j) k_{\epsilon}(\mathbf{x}_i, \mathbf{x}_j), \quad (5)$$

where $\mathbf{a} = (\delta, \mathbf{x})$ is an augmented vector in which δ indicates if a real experiment was performed and $k_{\delta}(\delta_i, \delta_j) = \delta_i \delta_j$ is a kernel indicator that equals one if both evaluations were performed with the real robot. Accordingly, two real experiments are expected to covary stronger than evaluations containing simulations.

Since k_{ϵ} is modeled inside the GP, we do not need to address explicitly the mapping between J_{sim} and J_{real} ; we only require assumptions about the difference between simulation versus real experiments in form of a mean and a covariance function. Additionally, δ has to be explicitly incorporated into the acquisition function of the optimization—otherwise only real experiments would be selected, because they deliver more information about the target function. This is done by introducing weight parameters w_i for both information sources. Thus, the acquisition function is expressed as

$$\mathbf{x}_{t+1} = \arg \max_{\mathbf{x} \in \mathbb{R}^d, i \in \{\text{sim}, \text{real}\}} \left(\frac{\Delta H_t(\mathbf{x})}{w_i} \right). \quad (6)$$

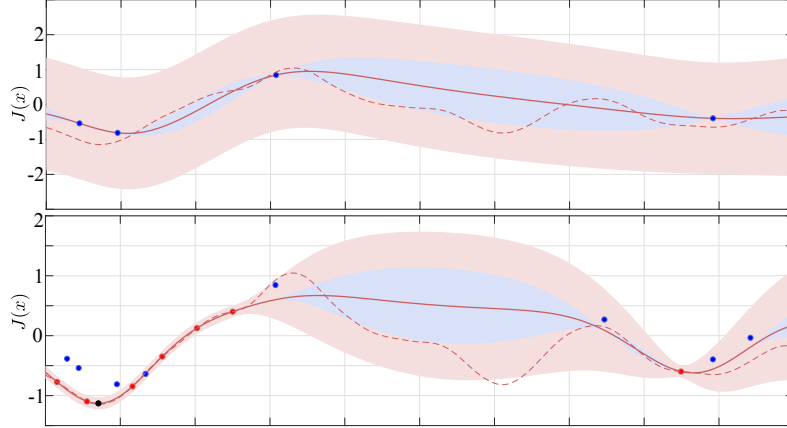


Fig. 2. Combination of simulation (blue dots) and real-robot (red dots) experiments in a synthetic example. The dashed red line represents the true cost function. Top: simulation experiments condition the mean posterior (red line) of the cost function of the physical system J_{real} and influence considerably the simulation uncertainty (blue shaded), but the uncertainty of the real system (red shaded) is only slightly affected. Bottom: this uncertainty is significantly reduced by real-robot experiments. Note that the difference of the simulation data points and J_{real} is captured by ϵ_{sim} (Eq. 4). The influence of simulation costs to the real system uncertainty is encoded in Eq. (5).

3.4 Bipedal Walking with Feedback Mechanisms

In this section, we briefly describe the gait we want to optimize [3]. The gait is based on an open-loop gait pattern generator as presented in [5]. It essentially produces high-dimensional trajectories for omnidirectional walking given a desired target velocity vector. The open-loop gait makes use of: the joint space, the inverse, and the abstract leg representations [17]. The abstract leg space is a representation of the leg pose, consisting of the leg extension, leg and foot angles, and is, in contrast to representations in Cartesian or joint space, designed for easy use for walking. The gait starts from a halt pose in the abstract space and incorporates several motion primitives such as leg lifting and leg swinging, also defined in the abstract space. The resulting pose is converted into the inverse space where more motion primitives are incorporated. Finally, the open-loop gait outputs a joint trajectory by converting the resulting inverse pose into the joint space.

In order to improve the stability and robustness of the robot, feedback mechanisms were added to the open-loop gait [3]. The orientation of the robot is represented by fused angles [2]. Using the deviations d_α and d_β of the fused roll α_B and fused pitch β_B from a desired orientation, e.g., from an upright torso pose, the activation value $\mathbf{u}$ of different corrective actions is calculated. The elements of $\mathbf{u}$ can be considered as the strength of corresponding corrective actions (feedback mechanisms) which are then applied to the open-loop gait in the abstract or inverse space. The corrective actions include: *arm swinging*, *hip movement*, *COM shifting*, *ankle tilting*, and *support foot tilting*. In order to obtain

the activation values $\mathbf{u}$, the deviations d_α and d_β are passed through a feedback pipeline composed of integrators, derivatives, mean filters and smooth deadband filters to produce a PID vector $\mathbf{e} \in \mathbb{R}^6$. This vector is then multiplied by a gain matrix $\mathbf{K}_a \in \mathbb{R}^{m \times 6}$ to generate the activation values of the corresponding m corrective actions.

4 Gait Parameter Learning

As explained in Section 3.4, the gait is composed of two main components: an open-loop central pattern generator and feedback mechanisms. In this paper, we address the problem of optimizing of the feedback mechanisms. Specifically, the activation gains $\mathbf{K}_a$ of the corrective actions will be optimized.

4.1 Cost Function

The fused angle deviations $d_\alpha = \alpha_{des} - \alpha$ (pitch) and $d_\beta = \beta_{des} - \beta$ (roll) give us an estimate about the unintended tilt of the robot induced by walking. These measurements are, however, very noisy. We apply thus a mean filter and a smooth deadband to d_α and d_β as proposed in [3]. In other words, we end up with the proportional part $e_{P\alpha}$ and $e_{P\beta}$ of the fused feedback vector $\mathbf{e}$. So, we define the stability criterion of the entire gait as the integral of $e_{P\alpha}$ and $e_{P\beta}$ along the gait duration T

$$\int_0^T \|e_{P\alpha}(\mathbf{x})\|_1 + \|e_{P\beta}(\mathbf{x})\|_1 dt. \quad (7)$$

This stability criterion will be part of our cost function. Additionally, we introduce a penalty term that smoothly regularizes the parameter $\mathbf{x}$. The penalty term is a logistic function of the form:

$$\nu(\mathbf{x}) = \frac{s}{1 + \exp(-\gamma(\|\mathbf{x}\|_2 - \lambda\|\mathbf{x}_{max}\|_2))}, \quad (8)$$

where $\mathbf{x}_{max}$ is the upper bound of $\mathbf{x}$, λ is a factor that affects the position of the transition, s represents the magnitude of the penalization and $\gamma \in \mathbb{R}$ controls the smoothness of the phase transition.

Since corrective actions in the sagittal plane are only activated by the fused angle pitch α and in the lateral plane by the fused angle roll β , we propose a cost function for the parameters $\mathbf{x}_l$ that have an effect on the lateral plane and another cost function for the parameters $\mathbf{x}_s$ that affect the sagittal plane:

$$J_\alpha(\mathbf{x}_l) = \int_0^T \|e_{P\alpha}(\mathbf{x}_l)\|_1 dt + \nu(\mathbf{x}_l) \quad (9)$$

$$J_\beta(\mathbf{x}_s) = \int_0^T \|e_{P\beta}(\mathbf{x}_s)\|_1 dt + \nu(\mathbf{x}_s). \quad (10)$$

In order to overcome the intrinsic error of the simulator, i.e., same parameters yield different results, we perform N evaluations in simulation with the same parameters and incorporate their mean into the cost function for the simulation:

$$\bar{J}_{sim,i}(\mathbf{x}) = \frac{1}{N} \left(\sum_{i=1}^N \int_0^T \|e_{Pi}(\mathbf{x})\|_1 dt \right) + \nu(\mathbf{x}), i \in \{\alpha, \beta\}. \quad (11)$$

For real-robot experiments, we set J_{real} as given by Eq. (9) and Eq. (10). In both (simulations and real-robot experiments), if the robot falls, a large cost is assigned to $\mathbf{x}$.

To define the kernel function of the optimization (Eq.(5)), we used the Rational Quadratic (RQ) kernel for k_{sim} and k_{ϵ} . The RQ kernel introduces three parameters (σ_k^2 , α and $\mathbf{l}$) to be tuned, also called hyperparameters. The $\mathbf{l}$ parameter roughly determines the distance of two points to significantly influence each other, the scale factor σ_k^2 determines the problem-specific signal variance, and α is a relative weight of large-scale and small-scale variations.

4.2 Termination Criteria

The most simple and frequently used criterion to stop global optimization algorithms is based on the number of iterations. Whereas this condition works fine for problems that are fast to compute, it loses applicability when iterations become more expensive. We propose a termination criterion which is based on entropy. We formulate a criterion that stops the algorithm as soon as the relative entropy $\mathbb{E}[\Delta H(\mathbf{x}_t)]$ reaches a defined value. Moreover, to ensure that outliers do not lead to a premature stop, we apply a saturated filter to the relative entropy (Fig. 3) defined for each iteration t with a velocity factor $0 < v < 1$ as:

$$(1 - v)\mathbb{E}[\Delta H(\mathbf{x}_{t-1})] + v\mathbb{E}[\Delta H(\mathbf{x}_t)]. \quad (12)$$

This criterion is applied after a minimum number of iterations since a bad prior mean can lead to a low relative entropy right after the first iteration. The entropy termination criterion is also combined with a maximum number of iterations criterion in case the relative entropy threshold is not reached mainly because the prior assumptions might not represent reality well enough.

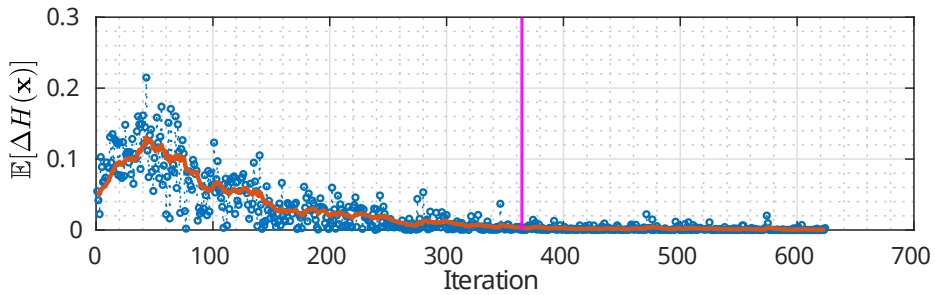


Fig. 3. Unfiltered (blue) and filtered with $v = 0.9$ (orange) relative entropy. The latter is used to terminate the optimization. Iterations after the magenta line can be skipped because of their very low information gain.

5 Evaluation

5.1 Experiment Setup

Robot Platform We test our approach on the igus® Humanoid Open Platform [4]. The robot has in total 20 degrees of freedom: 6 for each leg, 3 for each arm and 2 for the neck. The links of this platform are fully 3D printed. The robot is 92 cm tall and weighs 6.6 kg. The platform incorporates an Intel Core i7-5500U CPU running a 64-bit Ubuntu OS and a Robotis CM730 microcontroller board, which electrically interfaces with its Robotis Dynamixel MX actuators. A 3-axis accelerometer and gyroscope sensors are also contained in the CM730, for a total of 6 axes of inertial measurement.

Software Architecture Due to the limited computational power of the robot, the optimization and simulations are performed on a desktop PC with a Core i7-4890K CPU and 8 GB of RAM. The simulations are carried out in Gazebo 2.2 with a real-time factor of 1.5. We use ODE as the dynamics engine without constraint force mixing and an error reduction parameter of 0.2. The geometries of the joint links are approximated using convex hulls. The joints are controlled using the *ros_control* package. To avoid induced noise in the simulation, the simulator is reset after each performed trial. The Bayesian optimization run in Matlab 2017b. All the components required for the gait are implemented in C++ and executed on the robot’s computer. The interprocess communication is implemented using the ROS middleware.

Scenarios We propose two scenarios to evaluate our approach. We initially perform a 2D optimization to learn the P- and D-gain of the *Ankle Tilt* corrective action. In the second scenario, a 4D optimization to learn the P- and D- gains of the *Swing Arm* and *Ankle Tilt* actions is performed. In both scenarios, there are no other feedback mechanisms active. To avoid artifacts coming from the transient of a robot in a static configuration, all experiments start with the robot walking on the spot. The walking sequence is then defined as walking on the spot during three seconds and then walking forward with a speed of 0.3 m/s. We do not bias the optimizer specifying any initial values of the parameters. We compare the results of our algorithm against manually tuned parameters devised by experts. These parameters were used by the winning team NimbRo TeenSize at the RoboCup 2017 competition [19].

Parameters We parametrize the kernel function with $\mathbf{l} = \left(\frac{\mathbf{x}_{max}}{8}\right)$ and $\alpha = 0.25$ to produce a reasonable trade-off between exploration and exploitation. We use the same values of $\mathbf{l}$ and α for k_{sim} and k_{ϵ} . The standard deviation of k_{sim} is set to $\sigma_{sim} = 2.48$ and the standard deviation of k_{ϵ} is set to $\sigma_{\epsilon} = 2.07$ for the 2D optimization, whereas $\sigma_{sim} = 2.07$ and $\sigma_{\epsilon} = 1.79$ are set for the 4D optimization. The prior means are set to $\mu_{sim} = 53.3502$ and $\mu_{\epsilon} = -37.1385$. These values are chosen from initial experiments. Additionally, the penalty function $\nu(\mathbf{x})$ is

parametrized with $\lambda = 0.75$ and $s = 7.5$ to punish parameters larger than $\lambda \mathbf{x}_{max}$. The smoothness of the phase transition is set to $\gamma = 6$. The effort of the simulation and real-robot experiments are set to $w_{sim} = 10$ and $w_{real} = 50$, respectively, i.e., a real-robot walking sequence is five times as expensive as a simulated one.

5.2 Experimental Results

2D Optimization The algorithm performed in total 126 iterations, from which 20 walking trials were carried out with the real robot. This implies that approximately one real robot walking trial was required for every five simulations, which shows the applicability of the integration of simulation and real robot experiments for learning the activation value of the *Ankle Tilt* corrective action. The optimized values yielded a cost of 9.3, while the manually tuned ones resulted in a cost of 13.77. Thus, the optimization process found parameters that are approximately 32% better than the manually by-expert-tuned parameters. The resulting posterior is depicted in Fig. 4. The difference in the cost of the simulation compared to the cost with the real robot is mainly caused by the noise of the simulator, e.g., by modeling errors of the floor impacts.

Our approach was compared with a Random Search algorithm that greedily searches for the optimal value by corrupting the current best guess with uniform noise at each iteration. We applied this algorithm on the same experimental setup and a maximum of 25 real-robot experiments. With a cost of 11.04, the parameters of the random search outperformed the manually tuned ones. However, our method performed 20 real experiments and yielded 15% better results than random search. Additionally, the evaluations of the random search caused one fall of the robot which never occurred with our approach.

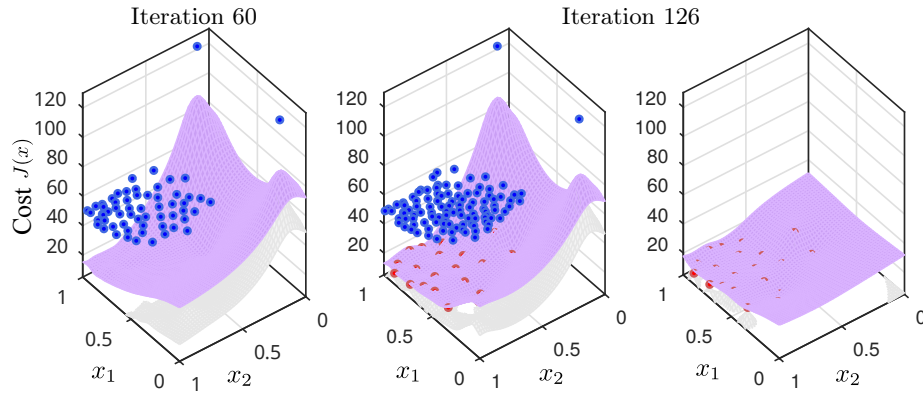


Fig. 4. Optimization of the parameters for the *Ankle Tilt* corrective action. The purple mesh resembles the posterior mean while the gray mesh shows the covariance. At the first iterations, the algorithm decided to perform only simulations (blue). Once enough information has been gathered in the simulation, real experiments (red) are carried out. To stress the contribution of the simulation, the rightmost plot shows the posterior considering only real-robot experiments.

4D Optimization For the 4D optimization, 301 iterations were carried out: 271 in simulation and 30 with the real robot, i.e., in average one real-robot experiment is required for every nine simulations. The resulting parameters were evaluated against manually tuned parameters, performing with the real robot 15 walking sequences for each set of parameters. The optimized parameters yielded an average cost of 10.38 and resulted, in comparison to the manually tuned gait with a mean cost of 16.28, in an improvement of 35%. In order to evaluate only the contribution of the stability in the cost, we subtract the penalty term of the cost; the resulting performance of the optimized parameters is 53% better than the manually tuned parameters.

Moreover, we compared the measured fused angle deviation (not fused angle feedback) and its integral during five trials. As expected, during the first three seconds (walking on spot) there are no significant differences. However, as soon as the robot starts walking forward, the deviations start to diverge and the difference between the set of parameters becomes apparent. In general, the optimized parameters reproducibly generate deviations of lower amplitude. The difference becomes more apparent observing the integral of the mean absolute deviation $\bar{D}_\alpha = \int_0^T \mathbb{E} [\|d_\alpha\|] dt$ depicted in Fig. 5.

A remarkable property of our approach is the fact that the real robot did not fall a single time during the optimization process, because parameters that resulted in a fall in simulation were ruled out without the need for real-robot experiments. Furthermore, the optimized gait looks qualitatively more stable and generally walks with a more upright torso compared to the manually tuned parametrization. A video of the gait with the optimized parameters is available online¹. The optimized gait was also tested in a very rough terrain, where the robot successfully traversed a series of debris (Fig. 5).

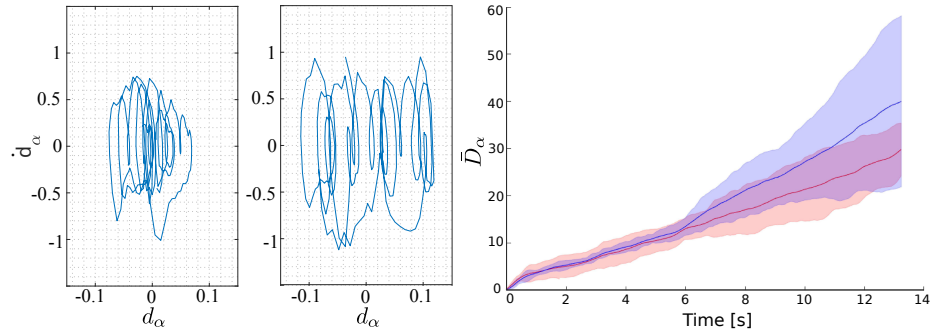


Fig. 5. Phase plots of the optimized (left) and manually (middle) tuned parameters for our gait. For clarity, only one walking sequence is displayed but all the evaluated sequences show a similar behavior. Right: integral of the mean absolute fused angle deviation $\bar{D}_\alpha$ of the optimized (red) and the manually tuned (blue) parameters. The shaded regions cover the values within two standard deviations ($\pm 2\sigma$).

¹http://www.ais.uni-bonn.de/videos/RoboCup_Symposium_2018

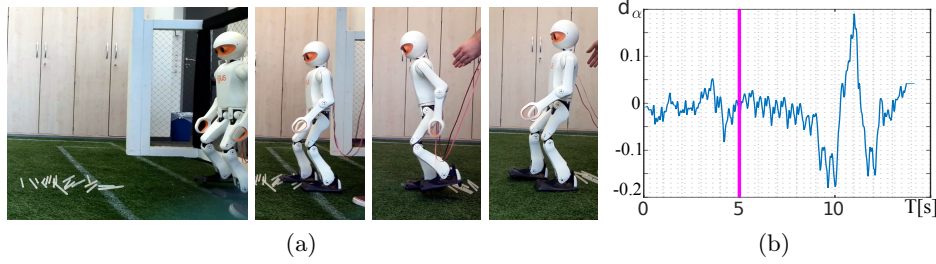


Fig. 6. Evaluation of the optimized gait parameters on very rough surfaces (artificial grass with small debris). a) Evaluation setup (leftmost) and pictures of the robot traversing the debris. b) The corresponding fused angle deviation over time. The magenta line indicates the moment of contact with the debris.

6 Conclusions

We presented an approach to trade off simulations and real-robot experiments for learning gait parameters based on a state-of-the-art Bayesian optimizer. We showed how the gait stability was improved with the parameters found by our approach. During the optimization process, the real robot did not fall a single time, which shows that the algorithm was successfully generalizing the information gathered from the simulation. This generalization also leads to a lower number of required physical experiments, which enables the applicability of our approach.

We observed a limitation of our method to be applied in higher dimensions. We hypothesize to solve this issue by using dimensionality reduction methods and by performing the optimization in a lower-dimensional space. Additionally, in the future, we also want to learn the hyperparameters of the kernel during optimization.

Acknowledgements

The authors would like to thank Alonso Marco from the Max Planck Institute for Intelligent Systems for providing the MF-ES implementation. This work was partially funded by grant BE 2556/13 of German Research Foundation.

References

1. Akrou, R., Sorokin, D., Peters, J., Neumann, G.: Local Bayesian optimization of motor skills. In: Proc. of the 34th Int. Conf. on Machine Learning (PMLR) (2017)
2. Allgeuer, P., Behnke, S.: Fused Angles: A representation of body orientation for balance. In: Int. Conf. on Intelligent Robots and Systems (IROS) (2015)
3. Allgeuer, P., Behnke, S.: Omnidirectional bipedal walking with direct fused angle feedback mechanisms. In: IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids) (2016)

4. Allgeuer, P., Farazi, H., Schreiber, M., Behnke, S.: Child-sized 3D Printed igus Humanoid Open Platform. In: Proc. of 15th IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids) (2015)
5. Behnke, S.: Online trajectory generation for omnidirectional biped walking. In: Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA) (2006)
6. Berkenkamp, F., Schoellig, A.P., Krause, A.: Safe controller optimization for quadrotors with gaussian processes. In: IEEE Int. Conf. on Robotics and Automation (ICRA) (2016)
7. Calandra, R., Gopalan, N., Seyfarth, A., Peters, J., Deisenroth, M.: Bayesian gait optimization for bipedal locomotion. In: Proc. of the 8th Int. Conf. on Learning and Intelligent Optimization (LION) (2014)
8. Cutler, M., How, J.P.: Efficient reinforcement learning for robots using informative simulated priors. In: IEEE Int. Conf. on Robotics and Automation (ICRA) (2015)
9. Deisenroth, M.P., Fox, D., Rasmussen, C.E.: Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2015)
10. Farchy, A., Barrett, S., MacAlpine, P., Stone, P.: Humanoid robots learning to walk faster: From the real world to simulation and back. In: Proc. of the Int. Conf. on Autonomous Agents and Multi-agent Systems (AAMAS) (2013)
11. Hanna, J.P., Stone, P.: Grounded action transformation for robot learning in simulation. In: Association for the Advancement of Artificial Intelligence Conf. (AAAI) (2017)
12. Heijmink, E., Radulescu, A., Ponton, B., Barasuol, V., Caldwell, D.G., Semini, C.: Learning optimal gait parameters and impedance profiles for legged locomotion. In: IEEE-RAS 17th Int. Conf. on Humanoid Robots (Humanoids) (2017)
13. Hengst, B., Lange, M., White, B.: Learning ankle-tilt and foot-placement control for flat-footed bipedal balancing and walking. In: 11th IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids) (2011)
14. Hennig, P., Schuler, C.: Entropy search for information-efficient global optimization. *Journal of Machine Learning Research* (2012)
15. Marco, A., Berkenkamp, F., Hennig, P., Schoellig, A.P., Krause, A., Schaal, S., Trimpe, S.: Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization. In: Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA) (2017)
16. Marco, A., Hennig, P., Bohg, J., Schaal, S., Trimpe, S.: Automatic LQR tuning based on gaussian process global optimization. In: IEEE Int. Conf. on Robotics and Automation (ICRA) (2016)
17. Missura, M., Behnke, S.: Self-stable omnidirectional walking with compliant joints. In: 8th Workshop on Humanoid Soccer Robots. IEEE-RAS Int. Conf. on Humanoid Robots (2013)
18. Rai, A., Antonova, R., Song, S., Martin, W.C., Geyer, H., Atkeson, C.G.: Bayesian optimization using domain knowledge on the ATRIAS biped. In: IEEE Int. Conf. on Robotics and Automation (ICRA) (2018)
19. Rodriguez, D., Farazi, H., Allgeuer, P., Ficht, G., Pavlichenko, D., Brandenburger, A., Kürsch, J., Behnke, S.: Advanced soccer skills and team play of RoboCup 2017 TeenSize winner NimbRo. In: RoboCup 2017: Robot World Cup XXI (2018)
20. Röfer, T.: Evolutionary gait-optimization using a fitness function based on proprioception. In: RoboCup 2004: Robot Soccer World Cup VIII (2005)