Automatic Parameter Optimization for a Dynamic Robot Simulation

Tim Laue[1] and Matthias Hebbel[2]

[1] Deutsches Forschungszentrum für Künstliche Intelligenz GmbH,
Sichere Kognitive Systeme, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany
tim.laue@dfki.de
[2] Robotics Research Institute, Section Information Technology,
Dortmund University of Technology, Otto-Hahn-Str. 8, 44221 Dortmund, Germany
matthias.hebbel@uni-dortmund.de

Abstract. One common problem of dynamic robot simulations is the accuracy of the actuators' behavior and their interaction with the environment. Especially when simulating legged robots which have optimized gaits resulting from machine learning, manually finding a proper configuration within the high-dimensional parameter space of the simulation environment becomes a demanding task. In this paper, we describe a multi-staged approach for automatically optimizing a large set of different simulation parameters. The optimization is carried out offline through an evolutionary algorithm which uses the difference between the recorded data of a real robot and the behavior of the simulation as fitness function. A model of an AIBO robot performing a variety of different walking gaits serves as an example of the approach.

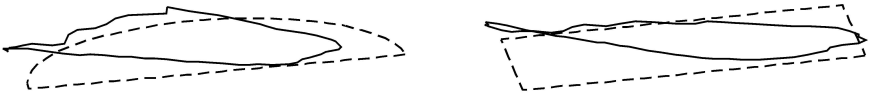
1 Introduction

When working with robots, using a simulation is often of significant importance. On the one hand, it enables the evaluation of different alternatives during the design phase of robot systems and may therefore lead to better decisions and cost savings. On the other hand, it supports the process of software development by providing a replacement for robots that are currently not on-hand (e. g. broken or used by another person) or not able to endure long running experiments. Furthermore, the execution of robot programs inside a simulator offers the possibility of directly debugging and testing them.

One characteristic trait of all simulations is that they can only approximate the real world, this inherent deficit is called *Reality Gap*. This affects all aspects of simulations: the level of detail as well as the characteristics of sensors and actuators. For most of the beforehand mentioned applications, the gap is of minor relevance as long as the simulated robot system performs in a reasonable way.

Within the RoboCup domain, actuator performance is a crucial aspect. Through applying optimization algorithms, impressive results regarding robot velocities have been achieved during the last years, e.g. by Röfer [1] and Hebbel et al. [2] using the AIBO robot, or by Hemker et al. [3] using a humanoid robot.

L. Iocchi et al. (Eds.): RoboCup 2008, LNAI 5399, pp. 121–132, 2009.



(a) Semi-elliptical locus **(b)** Rectangular locus

Fig. 1. Comparison of controlled (dashed) and real (solid) walk trajectories of an AIBO robot's joint during walking [2]

One common attribute of these algorithms is the strong exploitation of the environment's features, i.e. certain characteristics of the motors or the properties of the ground in this case. This leads to control trajectories that strongly differ from the resulting trajectories of the real robot joints, as shown in Fig. 1. For robot simulations, especially when working with legged robots which have a high number of degrees of freedom, this requires a proper parametrization, i.e. to simulate actuators that behave close to real ones. Otherwise, the simulated robot might not only behave unrealistic but could fail completely.

Currently, there exists a broad range of dynamic robot simulators which are used – not only – by RoboCup teams, e.g. Webots [4], the Übersim [5], SimRobot [6], or Microsoft Robotics Studio [7]. Also the RoboCup Simulation League introduced a fully dynamic robot simulation , additionally aiming towards a closer cooperation with real robots [8]. All these simulations allow a detailed specification of the environment, but demand the user to do this manually what might become an exhausting task given the high number of environmental parameters. Additionally, a once working parameter set is not guaranteed to be compatible with a different walking gait learned at a later point of time.

The contribution of this paper is to present a general multi-staged process which minimizes the reality gap between real and simulated robots regarding the behavior of actuators and their interaction with the environment. This optimization is carried out by using an evolutionary algorithm. The approach is kept general and transferable to different kinds of legged robots, its application is shown using a model of an AIBO robot as example.

A similar concept named *Back to Reality* by Zagal et al. [9], implemented within the UChilsim application [10], also automatically determines environmental parameters of a robot simulation. Nevertheless, their focus is the co-evolution of software parameters and the simulation, rather than the detailed optimization of a general purpose model.

This paper is organized as follows: Sect. 2 gives an overview of relevant parameters within a dynamic robot simulation, Sect. 3 describes the algorithm used for optimization. The multi-staged optimization setup is described in Sect. 4, its results are presented in Sect. 5.

2 Relevant Parameters

Within every robot simulation, there exists a vast number of parameters which influence realism and performance. For our work, we used the SimRobot

simulator [6] based on the free *Open Dynamics Engine (ODE)* by Russell Smith [11]. This engine provides a simulation of rigid body dynamics at industrial quality and has already been used in many projects, e. g. in most of the previously mentioned simulators.

In this section, we describe all parameters relevant for a reasonable simulation of actuators and their interaction with the environment. Most of these parameters are generic for all kinds of simulations, some of them are specific to ODE but have similar equivalents in other simulations. We differentiate between parameters that need to be optimized and those that may be kept static.

2.1 Parameters for Optimization

The parametrization of the motors has probably the highest impact on the correctness of the simulation. In our application (as well as in most others), only motor-driven hinge joints have been used to simulate the servo motors of a real robot. For every motor, there are five parameters: the *maximum velocity*, the *maximum torque*, and the three control factors of an – possibly given – *PID* controller. Vendor specifications (if available) of the motors regarding velocity and torque provide a good starting point for the optimization, but are definitely not directly transferable without any further optimization. The control parameters of servo motors are generally not accessible at all, not to mention their implementation.

When considering not only the motion of single joints but the motion of a whole robot, the interaction with the ground needs to be modeled, i.e. the friction between the robot surface and the floor. ODE provides an efficient approximation of the Coulomb fiction model. It is realized by temporarily adding so-called contact joints between colliding objects. For every pair of different surfaces, a set of six parameters is needed to configure these contact joints.

Two additional, ODE-specific parameters are the *Error Reduction Parameter (ERP)* and the *Constraint Force Mixing (CFM)*. They do not have any counterparts in the real world, their purpose is to keep the simulation mathematically stable by adding forces to preserve a correct joint alignment (ERP) and by allowing a certain amount (CFM) of constraint violation. Both can be used as global parameters as well as per joint. To keep the state space for the later optimization small, we assume that these parameters are the same for all joints and do not need to be adapted locally.

2.2 Fixed Parameters

Additionally to all afore-mentioned parameters, there is a huge set of parameters that does not need to be optimized and might be set to fixed values. Among these are the sizes, shapes, and positions of all body parts. Their manual measurement already provides a sufficient estimate. Even when using a dynamically simplified robot model as we do (see Fig. 3c), additionally optimizing these parameters would lead to an extreme growth of the state space.

The masses of all body parts can also be measured manually (after a disassembly of the robot). In case of complex or irreversibly connected parts, it

might be useful to optimize the exact masses and their centers. But this was not necessary for the robot used in this work.

One global force is the gravity. We kept it fixed to its standard value.

3 Optimization Algorithm

The goal of the optimization is to find parameters for the physical simulation engine which result in a behavior of the simulated robot as close as possible to the behavior of the real robot. Notice that rather than finding the real parameters like e. g. the real friction between the robot's legs and the carpet, the aim in this work is to find parameters that overall result in a similar behavior of the simulated robot. Thus, the problem itself cannot be specified mathematically which means that the only way to get the quality of the simulation (from now on called *fitness*) for a certain parameter setting is to try them out in the simulation and to measure how close they match the real robot's movements. This section shortly describes the optimization algorithm which has been used to learn the parameter setting. A standard Evolution Strategy with self-adaptation has been used which is well-established for problems with an unknown structure.

The Evolution Strategy has been developed by Schwefel and Rechenberg for the optimization of technical problems [12]. It is inspired by the biological evolution and is working with a population of individuals. Each individual represents a point $\boldsymbol{x}$ in the n–dimensional real-valued search space; the point $\boldsymbol{x}$ is encoded in the object parameters $x_1, \ldots, x_n$ of the individual and represents its genes. In terms of Evolution Strategies each individual can be assigned a function value $f(\boldsymbol{x})$ representing the quality of the solution, the so called fitness which has to be optimized. The process of the evolution is shown in Fig. 2. An amount of μ individuals constitute the parent population and create a new offspring population of the next generation with λ individuals. An offspring individual is created by recombining (mixing) the real-valued genes of ρ randomly selected parents followed by a random mutation of each object parameter such that the

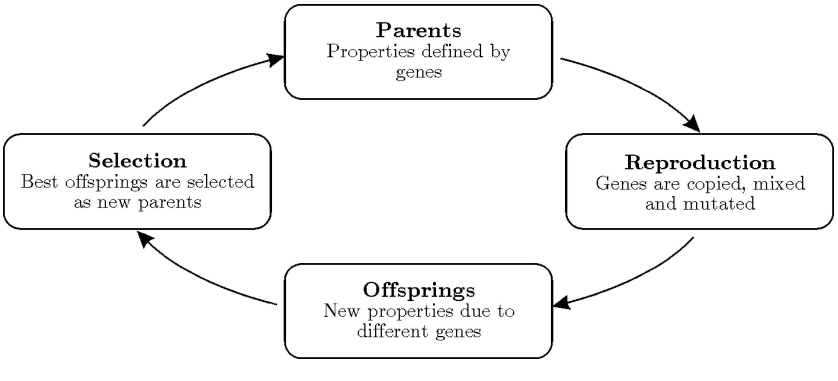


Fig. 2. Optimization cycle of the Evolution Strategy

offsprings differ in their genes from their parents. The fitness of each individual in the offspring generation will be measured and the selection operation chooses the μ best individuals to establish the new parent generation.

The chosen Evolution Strategy uses *self-adaptation* to control the strength of the mutation operation. Each object parameter has its own mutation strength. The values of the mutation strengths have a big effect on the progress speed of the evolution process. With a mutation strength chosen too big, the chance to overshoot the optimum rises, while small mutation strengths increase the probability to get stuck in a local optimum and furthermore the speed to approach the optimum is unnecessary small. Hence it is very important to adapt the mutations strengths to the current situation. Here the mutation strengths are are also included in each individual and are selected and inherited together with the individual's assignments for the object parameters. Thus, they have a higher probability to survive when they encode object parameter variations that produce fitter individuals [13].

After the fitness values of the offspring individuals have been assigned, the selection operator selects the parents for the next generation. Here the next parent generation was only generated from the offsprings, parents of the previous generation always died out, even if their fitness was been better than the best fitness of the best offsprings. Apparently this seems to slow down the convergence speed because previously found better solutions can be forgotten but this trade-off guarantees a good performance of the self-adaptation and also the risk to get stuck in a local optimum is reduced.

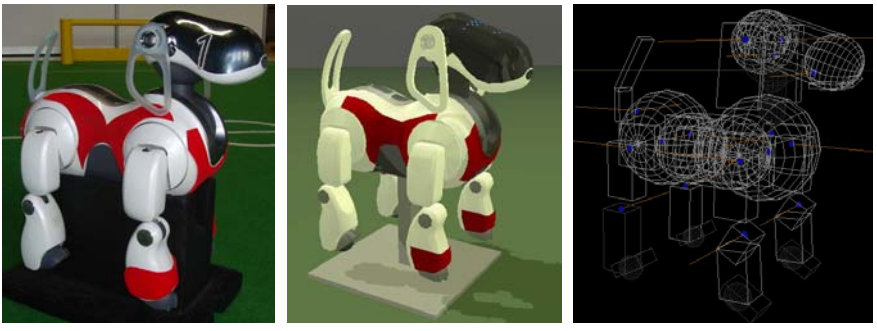
4 Experimental Setup

As pointed out earlier, a simulation model consists of a large number of parameters that need to be optimized. For our concrete application, we chose an AIBO robot. This robot has 12 degrees of freedom for locomotion (the additional degrees of freedom for head, ears, tail, and mouth are not relevant for our experiments). By assuming that the motor configuration is symmetrical, i.e. the left legs use the same motors than the right ones, and that within every leg different motors are used, parameters for six motors need to be considered. Furthermore, we have modeled only one kind of surface for the robot's feet, thus only one kind of friction – consisting of six parameters – will be optimized. Together with the two global parameters CFM and ERP, a total of 38 parameters needs to be optimized.

Learning all these parameters simultaneously would be a complex problem, thus the learning is done in three stages. The problem has been split up and the parameters have been clustered into groups which can be evaluated independently from each other. For each stage, a different setup and fitness function are used.

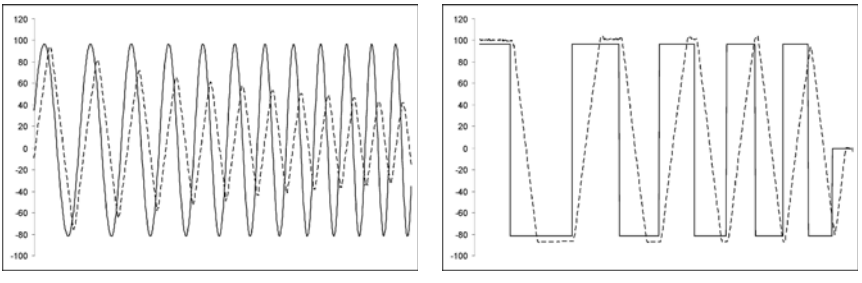
4.1 Motor Velocity and Initial Controller Parameters

In the first step, the maximum velocities and the settings of the PID controllers have to be learned. For this learning step, a special experiment has been set



(a) Real robot **(b)** Simulated robot **(c)** Physical representation

Fig. 3. Setup for learning the maximum velocities



(a) Sinusoidal control **(b)** Rectangular control

Fig. 4. An extract from the controlled (solid) and the measured joint movement (dashed) for evaluating the maximum velocities

up in order to minimize the effect of other physical parameters. As shown in Fig. 3, the robot has been placed on a platform with its legs hanging in the air. Two movement sequences have been performed on the real robot and the controlled and the sensor values of each joint have been recorded. The first sequence consisted of a sine movement with increasing frequency. The amplitude of the sine was set to 80% of the maximum angle of the according joint to be able to get the characteristics of the PID controller trying to position the joint at the requested position. Taking the maximum angle of the joint as the target position would give little feedback about the controller because the movement would be stopped abruptly by the hinge stop. In the second joint sequence, a rectangular oscillation has been used instead of the sinusoidial oscillation. Fig 4 shows this control and the resulting sensor curve exemplarily for the front right shoulder joint. The whole movement sequence for this learning step was approximately 60 seconds long.

In this learning step, the simulated robot's legs are controlled exactly like the real robot's legs. The goal is to match the movement of the real robot with the movement of the simulated robot. The fitness which is a measure for the

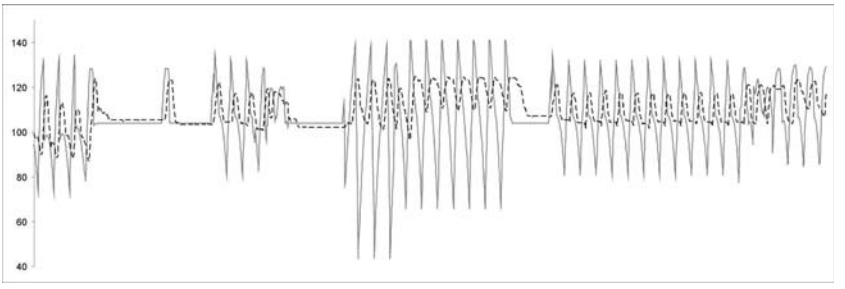


Fig. 5. An extract from the controlled (solid) and the measured joint angle (dashed) of the front right knee joint for evaluating the maximum forces

quality of the matching, is evaluated by summing up the squared differences between the sensor curve of the real robot's movement and the sensor curve of the simulated robot. Consequently, smaller values of the fitness express a better matching of the movements. The time offset between the measured sensor curve and the simulated sensor curve is of no interest and is excluded from the fitness measurement by shifting the simulated sensor curve in a small window in time back and forward; the smallest measured fitness is assigned.

4.2 Motor Torques

The second step serves an intermediate step to get a first estimate of the torques of the joint motors. For this purpose, the robot is walking on the ground as well as doing typical soccer movements, e.g. kicking, getting up, or pushing. Again, the controlled and measured (real) joint values of the physical robot are recorded. The recorded joint controls are used to move the joints of the simulated robot. The resulting joint movements are compared with the real robot's movement. The fitness of the simulation is – as in step 1 – calculated from the summed up squared differences between the real robot's sensor curve and the simulated robot's sensor curve. Figure 5 shows the movement of the front right knee joint of the robot together with the resulting sensor curve. The difference of resulting walk speeds of the simulated and the real robot is ignored yet. Since the maximum forces of the motors certainly depend to some extend on the friction between the robot and the floor (which will be optimized in the following step), the estimated forces in this step are not final.

4.3 Overall Robot Motion

We aim for universally valid parameters for the physical simulation, i.e. a realistic simulation for all walk requests with different walk patterns, instead of just one walk type with different walk requests. For this purpose, a set of 60 different walks of a real robot has been recorded. Among approved walking patterns used for soccer competitions, the set also includes suboptimal walks that contain much stumbling and sliding. In addition to the joint data, the translational

and rotational velocities of the robot have been recorded by an external motion tracking system.

Each walk lasts five seconds (with the first second not being taken into account for the fitness). The highest velocities reached by the robot have been about 480mm/s (translation forward) and about 220°/s (rotation). The fitness of a parameter set is the total sum of the fitness of all walks used. The fitness of one single walk is the sum of the squared velocity differences in both directions and in rotation (multiplied with an additional weighting to transform it into a range of values equal to the translational components).

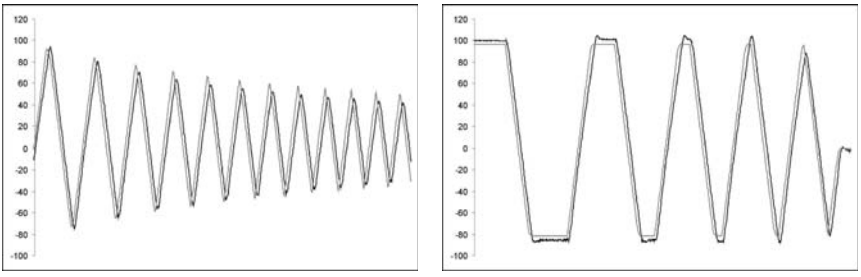
The main purpose of this step is to optimize all global parameters: friction, ERP, and CFM. Additionally, a further modification of the motors' torques and PID control parameters has been allowed, starting from the results of the previous step.

5 Experimental Results

This section presents the results of the three previously explained learning stages. Since each individual of the learning strategy represents a different parameter variation, its fitness can only be evaluated by running a simulation with this parameter setting. Since a lot of simulations are needed to find an optimal parameter set, we distributed the simulations on a compute cluster with 60 nodes. Each node consisted of a PC with Intel Pentium 4 CPU clocked at 2.4 GHz which is able to simulate the robot in real-time.

For each learning stage, the quality of the so far best found solution of each stage is also measured with respect to the resulting robot movement. This was not the primary optimization goal for the first two learning stages, but it is a benchmark to show that the selected learning stages are beneficial regarding the final goal. Table 1 shows the differences between the resulting walk speeds of the real robot and the simulated robot. For each walk direction (x is forward, y is to the robot's left and r is the rotation) the absolute maximum and the standard deviations are calculated, accordingly smaller values represent a better and more realistic simulation. The set of 60 walks was divided into two groups, named A and B. Group A consisted of 40 walk movements and group B of the remaining 20 walks. The entries in the column group A+B are evaluated based on the set union.

In the first stage, the maximum velocities and a preliminary setting for the PID parameters for each joint have been learned. An amount of 5 parents and 25 offsprings have been set as the population size. After 80 generations, the optimization of stage 1 has been aborted because it apparently converged. Fig. 6 shows in comparison an extract of the learning movement of the real robot and the simulated robot with the best learned parameters. The simulated movement gets very close to the real movement. Also the effect of the walk speeds is considerable: all standard deviations and absolute deviations have been decreased significantly. The greatest difference in x direction e. g. was reduced to 386.70 mm/s from 1227.47 mm/s. The maximum velocities in this stage are final



(a) Sinusoidal control **(b)** Rectangular control

Fig. 6. An extract from the real (solid) and the simulated joint movement (dashed) after learning the maximum velocities

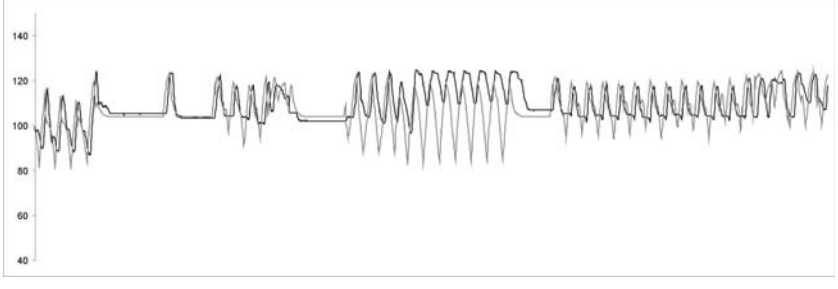


Fig. 7. An extract from the real (solid) and the simulated joint movement (dashed) of the front right knee joint after learning the (preliminary) maximum forces

because the movement of the free swinging legs in this setup is mostly dependent on the maximum available speed. The learned PID parameters instead are just preliminary because until now, there are almost no forces affecting the joints and they certainly also depend on the maximum torque of the motors which will be learned later. Thus the finally learned PID parameters in this stage are taken as starting values for the next learning stage.

In the next stage, the robot learned preliminary values for the maximum torques and the PID controllers. Preliminary for the reason that they are not independent from global physical parameters like e. g. friction between the robot's legs and the ground but will serve as an estimate of the real values for starting torques of the next learning stage. Fig. 7 shows the results of this intermediate stage. The simulated leg movement of the right knee joint gets in some parts close to the real movements, while in other parts the difference is still quite big. However, according to the Table 1 the deviations between the resulting walk speeds are again reduced. In this learning stage, the convergence took longer than in the first stage. After 400 generations the optimization of this stage was aborted, because after approximately 200 generations no significant improvement was observable.

Table 1. Standard and absolute maximum deviations of the velocity differences between the simulated robot and the real robot

		Group A		Group B		Group A+B	
		StdDev	MaxDev	StdDev	MaxDev	StdDev	MaxDev
Initial	x [mm/s]	144.51	574.16	273.33	1227.47	198.78	1227.47
	y [mm/s]	114.67	610.39	91.81	258.69	107.23	610.39
	r [°/s]	20.63	57.87	40.11	134.07	28.65	134.07
Stage 1	x [mm/s]	92.12	386.70	53.81	131.25	80.57	386.70
	y [mm/s]	48.77	180.27	119.93	372.77	82.01	372.77
	r [°/s]	15.47	37.24	24.06	88.24	18.91	88.24
Stage 2	x [mm/s]	89.78	368.90	72.65	264.94	84.49	368.90
	y [mm/s]	50.10	185.76	82.31	258.55	62.44	258.55
	r [°/s]	17.19	42.97	17.76	57.87	17.30	57.87
Stage 3	x [mm/s]	40.00	113.47	55.55	183.40	45.48	183.40
	y [mm/s]	35.49	150.34	75.69	255.98	52.26	255.98
	r [°/s]	10.31	24.06	15.47	41.25	12.61	41.25
Webots	x [mm/s]	123.65	442.58	91.90	199.86	113.01	442.58
	y [mm/s]	147.80	421.33	218.06	502.51	173.35	502.51
	r [°/s]	44.12	178.76	42.97	129.49	43.54	178.76

For the last learning stage, an Evolution Strategy with 9 parents and 50 offsprings has been chosen. In this stage, 32 parameters (8 global parameters and the maximum torques with PID settings for each of the 6 joints) have been optimized at once. The maximum torques and PID parameters for the joint motors were initialized with relatively small mutation strengths because they have already been estimated in the previous learning stage. In this stage, only the 40 different walks in group A have been considered for the parameter optimization. The optimization was not executed on the 20 walks in group B, which serves in this stage as a test case to show that the found parameters for group A are generally valid and not specific for this group. The optimization in this case was stopped after 200 generations. The maximum and standard deviations clearly decreased for the walks of group A, but also the simulated walks in group B got significantly closer to the real walks.

An analysis of the remaining most problematic walks disclosed a shortcoming of our simulation model: only one type of friction is realized between all robot parts and the ground. This is a problem especially for the AIBO robot because it mostly walks on the plastic covers of its elbows which allow some degree of slipping on the ground. Nevertheless, some (especially backward) walks are problematic for our simulation because they make use of the rubber caps at the end of each leg exploiting the good grip between the rubber and the ground.

Finally, to get a rating of the quality of our simulation using the optimized parameters, we compared the walks in group A and B with their performance in the commercially available robot simulator Webots which also includes a dynamic model of the AIBO robot. The results are presented in the last row in Table 1. The simulation with the Webots simulator is mostly better than with

our initial setting, but after the first learning stage, the physical simulation with the optimized parameters can significantly outperform the Webots model with the walks in both groups A and B.

6 Conclusion and Future Works

In this paper, we presented an approach for optimizing parameters of a dynamic robot simulation. By dividing the optimization process into different stages, it was possible to deal with the high-dimensionality of this problem. The application of this approach to a model of an AIBO robot whose joint and friction parameters were improved continuously performed well. This is especially revealed by comparing the results with the accuracy of another, well-established robot simulator. The inaccuracies resulting from modeling only one type of friction verify the assumption that a high level of detail is needed for accurate models in dynamic robot simulations.

Our results lead to several starting points for further investigations. Although the final parameters close the reality gap to a large extend, improvements seem still to be possible. These could eventually be reached by taking more parameters – for instance in a fourth learning stage – into account, e.g. the center of mass of certain major body parts or the anchor points of the joints. Since all experiments could be made offline, computing time was not in the focus of our research. The applied Evolution Strategy demonstrated to be able to cope with the optimization problem but it would be interesting to evaluate, if other algorithms are able to provide comparable results in less time.

To further evaluate the presented approach, optimizing parameters for other robot models is a necessary proceeding. This will be done for a humanoid robot in the near future.

Acknowledgments

This work has been partially funded by the Deutsche Forschungsgemeinschaft in the context of the Schwerpunktprogramm 1125 (*Kooperierende Teams mobiler Roboter in dynamischen Umgebungen*).

References

1. Röfer, T.: Evolutionary Gait-Optimization Using a Fitness Function Based on Proprioception. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, pp. 310–322. Springer, Heidelberg (2005)
2. Hebbel, M., Nistico, W., Fisseler, D.: Learning in a high dimensional space: Fast omnidirectional quadrupedal locomotion. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X. LNCS (LNAI), vol. 4434, pp. 314–321. Springer, Heidelberg (2007)
3. Hemker, T., Sakamoto, H., Stelzer, M., von Stryk, O.: Hardware-in-the-loop optimization of the walking speed of a humanoid robot. In: CLAWAR 2006: 9th International Conference on Climbing and Walking Robots, Brussels, Belgium, September 11-14, pp. 614–623 (2006)

4. Michel, O.: Cyberbotics Ltd. - WebotsTM: Professional Mobile Robot Simulation. International Journal of Advanced Robotic Systems 1(1), 39–42 (2004)
5. Go, J., Browning, B., Veloso, M.: Accurate and flexible simulation for dynamic, vision-centric robots. In: Proceedings of International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004) (2004)
6. Laue, T., Spiess, K., Röfer, T.: SimRobot - A General Physical Robot Simulator and Its Application in RoboCup. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020, pp. 173–183. Springer, Heidelberg (2006)
7. Jackson, J.: Microsoft robotics studio: A technical introduction. Robotics and Automation Magazine 14(4), 82–87 (2007)
8. Mayer, N.M., Boedecker, J., da Silva Guerra, R., Obst, O., Asada, M.: 3D2Real: Simulation League Finals in Real Robots. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006: Robot Soccer World Cup X. LNCS (LNAI), vol. 4434, pp. 25–34. Springer, Heidelberg (2007)
9. Zagal, J.C., Ruiz-del-Solar, J., Vallejos, P.: Back to Reality: Crossing the Reality Gap in Evolutionary Robotics. In: Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles - IAV 2004 (2004)
10. Zagal, J.C., del Solar, J.R.: UCHILSIM: A Dynamically and Visually Realistic Simulator for the RoboCup Four Legged League. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, pp. 34–45. Springer, Heidelberg (2005)
11. Smith, R.: Open Dynamics Engine - ODE (2007), `www.ode.org`
12. Beyer, H.G., Schwefel, H.P.: Evolution strategies – A comprehensive introduction. Natural Computing 1(1), 3–52 (2002)
13. Beyer, H.G.: Toward a theory of evolution strategies: Self-adaptation. Evolutionary Computation 3(3), 311–347 (1996)