Quantification and Minimization of the Simulation-Reality-Gap on a BRIO® Labyrinth Game

Constantin Bergatt[1], Jan Hendrik Metzen[1], Elsa Kirchner[1,2], and Frank Kirchner[1,2]

[1] DFKI Bremen, Robotics Innovation Center
Robert-Hooke-Straße 5, 28359 Bremen, Germany
[2] University Bremen, Robotics Group
Robert-Hooke-Straße 5, 28359 Bremen, Germany

Abstract. In this paper we present a new method, the so-called ϵ-tunnel, that can be used to quantify the simulation-reality-gap by comparing the behavior (the state-trajectory) of a baseline system (for instance a real robotic system) and a model of this system (e.g. a physic-based simulation). With this ϵ-tunnel, the impact of a change of the model's parameter can be analyzed. Furthermore, we present an approach to automate the optimization of these parameters and present some results obtained on a robot system that is based on a BRIO®labyrinth game.

1 Introduction

Robots are applied to increasingly complex scenarios and have to consider many external influencing factors in their decision making. Adaptive methods like reinforcement learning (RL) [1] promise to enable robots to react autonomously to changes of their environment and to adapt their behavior automatically. Applying RL involves training of the robot in order to adapt the parameters or the structure of its controller to the given task and environment. This training can happen either directly in the actual environment or in a testbed that mimics certain characteristics of the environment the robot will face later on. In the second case, the reproduction of the correct environmental conditions (e.g. extreme temperatures, high pressure, zero gravity) poses a major challenge.

One approach to reduce the effort of the training process is to train the robot within a simulation. In contrast to conducting the training in reality, in a simulation even extreme environmental conditions can be achieved and there is no wear out or damaging of the robot during the training. Furthermore, in a simulation the training process can be fully automated (i.e., it does not require human supervision and intervention), since the state of the simulated robot can be reseted directly (e.g. by putting the robot back to its starting position), and the training time can potentially be reduced since a simulation can often run faster than in real time.

Often a robot trained within a simulation acts different in the real world than in the simulation. In [2–4] the discrepancy between the behavior of the robot within the simulation and in reality is called the *simulation-reality-gap* [5]. In contrast, in this work simulation-reality-gap denotes the difference of the state trajectories between both systems when the same actions are executed. Ideally the simulated and the real system should behave identical. That is, if both systems are in the same state and the same action is executed on both systems, the subsequent states of the real and the simulated system should be identical. Because of the noise and error of real components and the inability of simulations to model reality perfectly, there will always be a difference between the states of the simulated and the real system and thus a simulation-reality-gap.

In this work, a new method called the ϵ-tunnel for quantifying the simulation-reality-gap is presented that is based on a direct comparison of state trajectories instead of an indirect comparison of the quality of the learned behaviors. We have tested our approach on the so-called "Labyrinth Testbed", a system based on the BRIO®labyrinth game (see Figure 2), and the corresponding simulation [6]. The goal of the game is to steer a ball indirectly through a maze from a start to a goal position by changing the orientation of the board. As an example, quantification and minimization of the simulation-reality-gap is shown for optimizing the parameters of the simulated system's motor controller.

2 The ϵ-Tunnel

A common approach to quantify the simulation-reality-gap is to train an agent within a simulation and compare the fitness obtained in the simulated system with the fitness achieved on the real system after transferring the agent [2, 4, 5, 7–9]. In contrast, the proposed quantification of the simulation-reality-gap via the ϵ-tunnel is based on a comparison of state trajectories instead of a comparison of fitness values.

For the proposed ϵ-tunnel approach, a fixed sequence of actions is executed repeatedly on both systems (the real and simulated one). In parallel, the m-dimensional state of both systems is recorded with the same sampling frequency. The state $s[t_n]$ of a system at a given sampling point t_n consists of all sensor data and internal state information that are available on the real system.

Next the p recorded state trajectories $s_i[t_n]$ ($i \in \{0, \ldots, p-1\}$) of each system are averaged:

$$\bar{s}[t_n] = \frac{1}{p} \cdot \sum_{i=0}^{p-1} s_i[t_n] \tag{1}$$

Afterwards the minimal $\epsilon^-[t_n]$ and maximal relative deviation $\epsilon^+[t_n]$ of all state trajectories $s_i[t_n]$ from the average $\bar{s}[t_n]$ are determined for any given sampling point t_n:

$$\epsilon^-[t_n] = \min_{i \in [0,p-1]} (s_i[t_n] - \bar{s}[t_n]) \qquad \epsilon^+[t_n] = \max_{i \in [0,p-1]} (s_i[t_n] - \bar{s}[t_n]) \tag{2}$$

Therefore the state space or the so called ϵ-*tunnel* $S[t_n]$ (see Figure 1) of a system can be described as follows:

$$S[t_n] = \{s \mid \bar{s}[t_n] + \epsilon^-[t_n] \leq s \leq s[t_n] + \epsilon^+[t_n]\} \tag{3}$$

Depending on the uncertainties of the system, the diameter of the actual ϵ-tunnel will vary. The ϵ-tunnel of a real robot with noisy sensors and imprecise actuators will be comparably broad, while the ϵ-tunnel of a robot within a deterministic simulation with no noise will be reduced to one single state, i.e., the diameter of the state space will be zero.

Thereupon, the resulting ϵ-tunnel of the real and the simulated system are compared with regard to their *alikeness*. Both systems are *alike* at a given sampling instance t_n, if the ϵ-tunnel of the real $S_R[t_n]$ and the simulated system $S_S[t_n]$ have a non-empty intersection:

$$S_R[t_n] \cap S_S[t_n] \neq \emptyset \tag{4}$$

By inserting (3) in (4) for both systems the condition can be converted to:

$$\epsilon_R^-[t_n] - \epsilon_S^+[t_n] \leq \bar{s}_S[t_n] - \bar{s}_R[t_n] \leq \epsilon_R^+[t_n] - \epsilon_S^-[t_n] \tag{5}$$

Furthermore two additional scaling factors k_R and k_S can be introduced to be able to increase or decrease the diameter of the ϵ-tunnel of both systems according to the desired similarity of both systems:

$$k_R \cdot \epsilon_R^-[t_n] - k_S \cdot \epsilon_S^+[t_n] \leq \bar{s}_S[t_n] - \bar{s}_R[t_n] \leq k_R \cdot \epsilon_R^+[t_n] - k_S \cdot \epsilon_S^-[t_n] \tag{6}$$

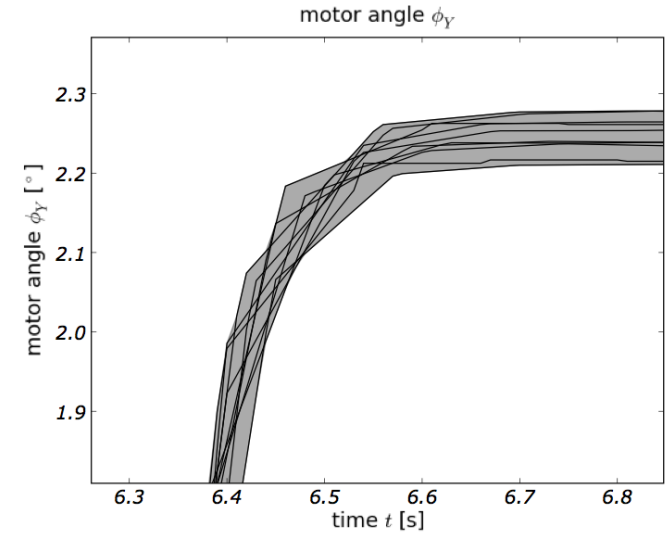
Finally by calculating the percentage of alike states of the real and the simulated system in comparison to the number N of all sampled states (percentaged alikeness), the simulation-reality-gap SRG can be quantified as follows:

$$SRG = \frac{\#\{t_n \mid fulfills\ (6)\}}{N} \tag{7}$$

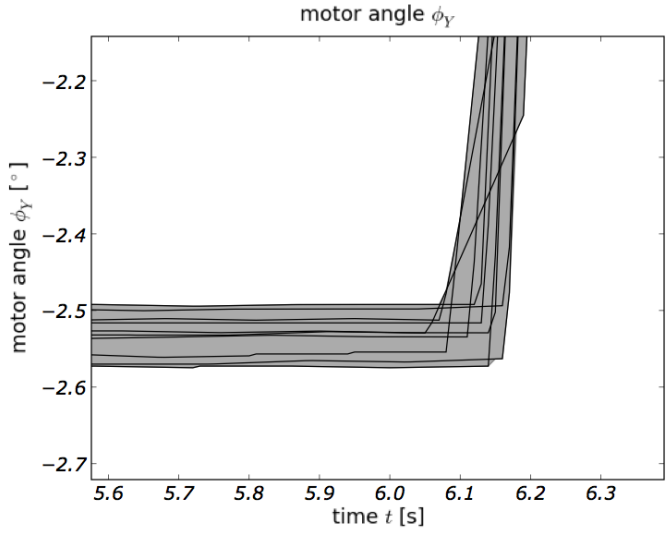
3 The Labyrinth Testbed

An off-the-shelf BRIO®labyrinth game (see Figure 2) has been equipped with two servo motors and two potentiometers, which allows the remote control and the measurement of the orientation of the game's board. Due to a high positioning error of the used servo motors, an extended motor control is used taking into account the local positioning error of each servo motor and the slackness of the used rubber belt.

Furthermore a camera is placed above the game which allows to estimate the current position of the ball on the board. The developed vision algorithm determines the position of the ball in the camera image based on the brightness



(a)



(b)

Fig. 1. The figure shows two exemplary scale ups of the ϵ-tunnel for the motor angle ϕ_Y. The solid lines represent the different trajectories $s_i[t_n]$ of each execution of the action sequence on the real system for the motor angle ϕ_Y. The shaded area marks the state space $S_{\phi_Y}[t_n]$ of the motor angle ϕ_Y.

of each pixel and maps the detected pixel coordinates of the ball position onto the corresponding labyrinth coordinates using landmarks.

Additionally a piezo sensor has been integrated to detect that a ball has fallen through a hole of the maze. If the piezo sensor detects such a ball loss, the custom-made ball depot can be used to supply a new ball and thus, it is possible to play more than one session without putting the ball back manually.

For the simulation of the labyrinth a built-in physics engine is used based on the Open Dynamics Engine (ODE) [10], which provides a high-performance library for simulating rigid body dynamics in real-time. ODE uses a Lagrange multiplier method with a first order integrator and an approximation to the Coulomb friction model. The board of the labyrinth is simulated by a precise 3D mesh model.

The simulation uses actuators whose velocities are controlled by PID-controllers with the default values $P = 1.0$, $I = 0.0$, and $D = 0.0$. The rotational speed of the simulated motors is controlled based on the continuous difference $\triangle\alpha$ between the target and the current motor angle. The current rotational speed v_n is calculated as the sum of a proportional, an integral, and a derivative term (8).

$$v_n = P \cdot \triangle\alpha_n + I \cdot (\sum_{k \leq n} \triangle\alpha_k \cdot \triangle t_k) + D \cdot (\triangle\alpha_n - \triangle\alpha_{n-1}) \tag{8}$$

Also it is possible to interface the simulated system, e.g. to receive the actual sensor readings, to control the board angle of the simulated board or to change parameters of the simulation.

For more information, we refer to *http://robotik.dfki-bremen.de/en/research/robotsystems/testbed-brio-labyrinth.html*

4 Case Study: Optimizing the Parameters of the Simulated Servo Motors

In the following section we will present a case study of the ϵ-tunnel. As an example, the motor parameters of the given simulation will be optimized in order to achieve a similar behavior of the real and simulated actuators. Thereupon, the impact of the optimized parameters on the simulation-reality-gap between the actuators of both system is evaluated by using the ϵ-tunnel.

Whereas the rotational speed of the simulated actuators is controlled based on the difference between the target and the current angle, the real servo motors (Robotis DX-117) are controlled based on the current, discrete position error (target minus current position) of the servo motors [11].

Since the controllers of the real servo motors are fixed, the three parameters P, I, and D of the PID-Controller of the simulated motors need to be chosen appropriately, in order to achieve a similar behavior of the simulated servos and the real servos. Additionally, a delay of the response of the real servo motors to a command was detected. To account for that, an additional parameter *delay* was introduced that needs to be optimized in order to simulate this motor delay appropriately.

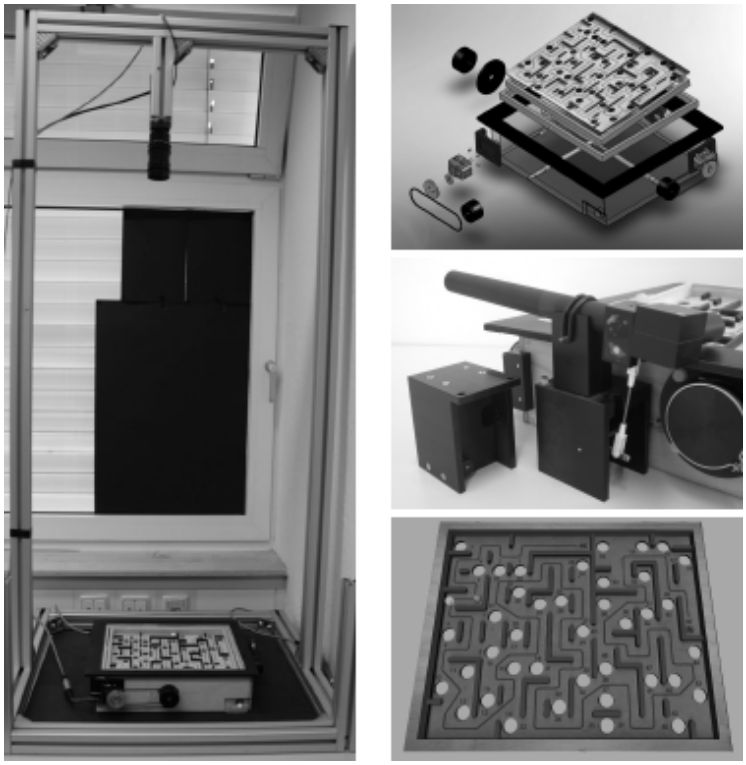


Fig. 2. The left image shows the setup of the BRIO®labyrinth testbed with a camera mounted approximately 1 meter above the board and servo motors installed on the game to control the board's orientation. The upper right image shows an exploded view of the BRIO®labyrinth system, the middle right image shows the ball depot, and the lower right image shows the latest version of the simulation.

4.1 Data Acquisition

For optimizing the four parameters (P, I, D and $delay$) of the simulated servos, the behavior of the real system has been measured as follows:

1. Several different target angles have been issued to the real servo motors (each one 100 times starting always in angle 0°) and the trajectories of the two motor's angles ϕ_X and ϕ_Y have been recorded. The target angles for motor angle ϕ_X have been $-5°$, $-4.5°$, ..., $4°$, $4.5°$, $5°$ and those for the motor angle ϕ_Y have been $-4°$, $-3.5°$, ..., $3°$, $3.5°$, $4°$ (larger angle were not possible due to the physical limitations of the BRIO®labyrinth game).
2. Afterwards the 100 trajectories of each test series are resampled to a sampling frequency of $100Hz$ using linear interpolation.
3. Finally, the 100 resampled trajectories are averaged.

4.2 Manual Optimization

Taking into account all the averaged trajectories of the single rotations of both real servo motors, the parameters for the simulated servo motors were optimized manually by trial and error. In each iteration one of the four parameter values was increased or decreased and the resulting trajectories of the simulated motor angles were compared to the corresponding averaged trajectories of the real system visually. The best results of this manual optimization could be achieved with $P = 20.0$, $I = 0.0$, $D = 0.0$ and $delay = 30ms$ (see parameter set *manual* in Table 1).

4.3 Optimization Using Evolution Strategy

In a second test, an evolution strategy [12] was used to optimize the parameters of the simulated servo motors automatically. For this a $(5+100)$-evolution strategy was implemented in the Python programming language. In this implementation a single individual encodes the four motor parameters P, I, D and $delay$ plus an individual mutation step size σ. Each motor parameter consist of a value between 0.0 and 1.0. Therefore the range of allowed values for each motor parameter (P: $[-25.0, +25.0]$; I: $[-20.0, +20.0]$; D: $[-20.0, +20.0]$ and $delay$: $[0ms, 50ms]$) was normalized to corresponding values between 0.0 and 1.0. The mutation step size σ_0 of the first parent individual was initialized with a value of 0.1. The mutation step size σ_{n+1} of each individual is determined during the creation of the individual based on the mutation step size σ_n of the corresponding parent individual using:

$$\sigma_{n+1} = \sigma_n \cdot N(0,1)^2 \tag{9}$$

The fitness of a single individual is calculated using a technique developed within the "Labyrinth 2" project[3]. In this approach the respective rotation commands C are executed on the real and on the simulated system, where the simulation uses the parameters encoded in the single individual.

The trajectories ϕ_k^R of the real system have been generated using the procedure outlined in Section 4.1. The respective rotation command has to be executed only once for each parameter set on the simulated system since the simulation is deterministic. Afterwards the trajectories ϕ_k^S of the simulated motor angles are resampled to a sampling interval $\triangle t$ of $10ms$ (corresponding to the common sampling frequency of $100Hz$) using linear interpolation and the differences between the trajectories of both systems are integrated over time to estimate the deviation E_C of the two systems for the actual rotation command C (see Figure 3).

The sum E_C is then divided by the duration of the rotation on the real system t_{total} to normalize the result:

[3] http://robotik.dfki-bremen.de/en/research/projects/cognitive-robotics/labyrinth-2.html

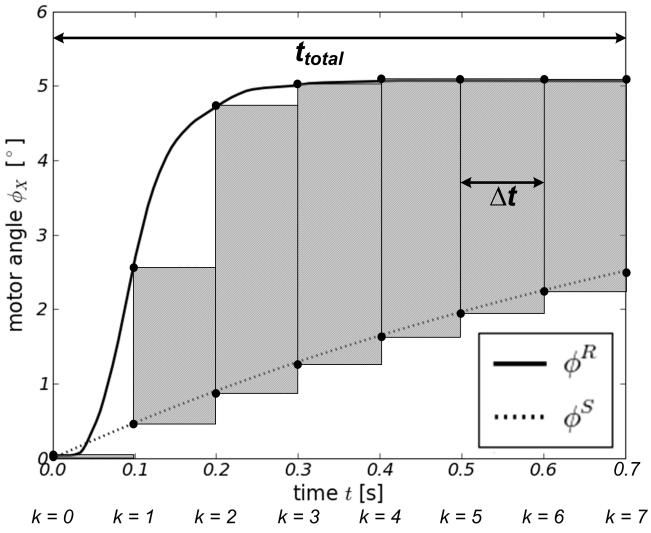


Fig. 3. The figure shows the sum E_C of the deviation for a trajectory ϕ_k^R of the real and ϕ_k^S of the simulated motor angles for time steps $\triangle t$ of $100ms$ for the rotation of the motor angle ϕ_X from 0° to 5°.

$$E_C = \frac{1}{t_{total}} \cdot \sum_k |\phi_k^R - \phi_k^S| \cdot \triangle t \tag{10}$$

The fitness value F of a single individual is set to the sum over all recorded command's errors E_C:

$$F = \sum_C E_C \tag{11}$$

For each servo motor, the evolution was run separately for 150 generations. The resulting set of parameters for the servo motors (ES (ϕ_X) and ES (ϕ_Y)) is shown in Table 1. According to the obtained parameter values of the simulated motor controller ($I \approx 0$ for all configurations) the real servo motor controller seems to behave like a PD controller. Due to the absence of any information about the used controller type in the manual [11] of the servo motor this assumption can't be confirmed.

Figure 4 shows an exemplary comparison of the trajectories of the motor angle ϕ_X of the real system (real) and the simulated system using three different parameter sets (default parameters used by the simulation, manually optimized parameter set and optimized parameter set using evolution strategy). As can be seen in Figure 4 the discrepancy between the behavior of the simulated and the real motors was reduced with both sets of optimized parameters.

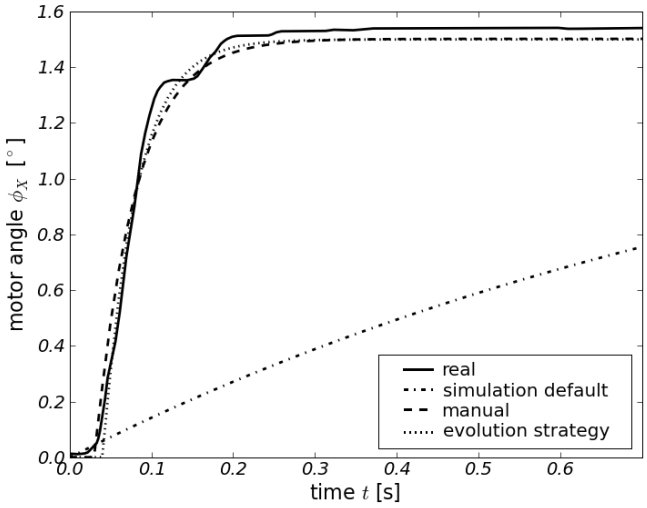


Fig. 4. The figure shows a comparison between the recorded trajectory of the motor angle ϕ_X of the real system and the resulting trajectories of the simulation using the default parameters as well as the manually and automatically optimized motor parameters for the simulated actuators.

Table 1. Used sets of motor parameters for the simulated actuators

Set of motor parameters used by the simulation	P	I	D	*delay* (s)
standard	1.0	0.0	0.0	0.0
manual	20.0	0.0	0.0	0.03
ES (ϕ_X)	24.52	≈ 0.0	0.95	0.049
ES (ϕ_Y)	20.24	≈ 0.0	−7.33	0.049

4.4 Applying the ϵ-Tunnel

For quantifying the impact of the optimization of the simulated system's motor parameters onto the simulation-reality-gap, the ϵ-tunnel is used. In our example the trajectories of the motor angles ϕ_X and ϕ_Y of the real system are compared to the trajectories of the simulated system using different sets of motor parameters (see Table 1).

Since the ϵ-tunnel is computed based on the comparison of the state trajectories for given action sequences, first a sequence of actions to be executed on both systems was selected. At the start state, both motor angles ϕ_X and ϕ_Y have been set to 2.5°. At $t = 0s$ the target angle for ϕ_X was set to −2.5°. Two seconds later ($t = 2s$) the target angle for ϕ_X was reset to 2.5°. Analogously,

the target angle for ϕ_Y was set at $t = 4s$ to $-2.5°$ and reset to $2.5°$ after two seconds ($t = 6s$). After additional two seconds ($t = 8s$) the execution of the action sequence was completed.

The action sequence was executed 10 times on the real system and the resulting states were recorded with a sampling frequency of $100Hz$. The recorded trajectories $s_i[t_n]$ on the real system were averaged according to Equation 1.

Afterwards the maximal positive $\epsilon^+[t_n]$ and negative deviation $\epsilon^-[t_n]$ among all recorded trajectories $s_i[t_n]$ from the average state $\bar{s}_{real}[t_n]$ were determined for any given sampling time-point t_n (according to Equation 2) and were filtered to get a smoother estimate of the upper $\delta^+[t_n]$ and lower limit $\delta^-[t_n]$ of the state space of the real system at each time-point t_n (12).

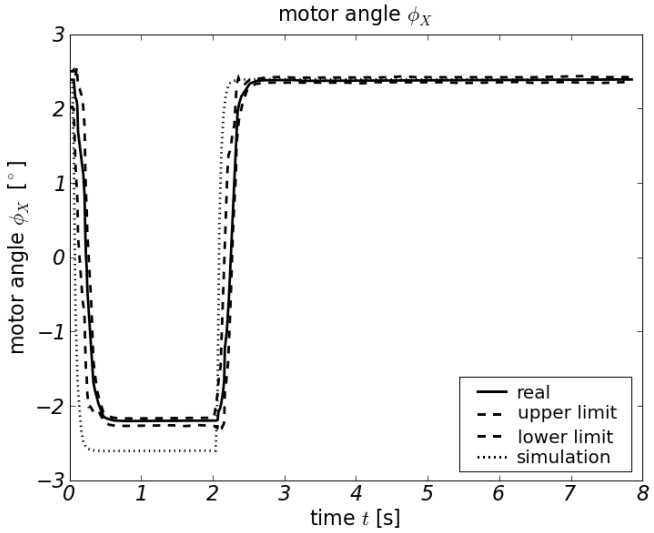
$$\delta^+[t_n] = \frac{1}{11} \cdot \sum_{k=-5}^{5} |\epsilon^+[t_{n+k}]| \qquad \delta^-[t_n] = \frac{1}{11} \cdot \sum_{k=-5}^{5} |\epsilon^-[t_{n+k}]| \tag{12}$$

Due to the deterministic behavior of the used simulation, the given action sequence was only executed once on the simulated system. The state of the simulated system $s_{sim}[t_n]$ was recorded in parallel. The percentaged alikeness of the state of the simulated system with the ϵ-tunnel of the real system was determined for different values of k_R by using (13). In this case the scaling factor k_S could be ignored because of the deterministic behavior of the simulation.

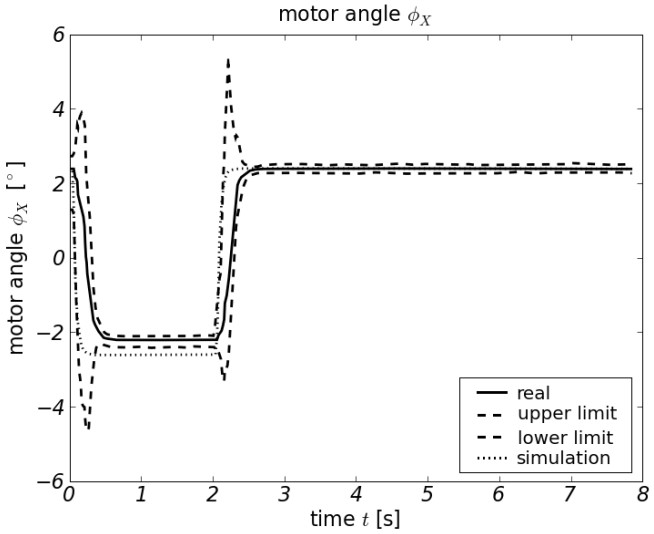
$$k_R \cdot \delta^-[t_n] \leq s_{sim}[t_n] - \bar{s}_{real}[t_n] \leq k_R \cdot \delta^+[t_n] \tag{13}$$

In this example, the resulting percentaged alikeness of the motor angles ϕ_X and ϕ_Y of the simulated system with the state space of the real system's motor angles for the different sets of motor parameters of the simulation was used as indicator for the simulation-reality-gap between the real and the simulated actuators. A high percentaged alikeness for small values of k_R indicates an almost identical behavior of the real and simulated actuator and therefore a small simulation-reality-gap between the actuators of both systems (see Figure 5).

The alikeness between the motor angles of the simulated system and the state space of the real system can be seen in Table 2 and 3. In general, the alikeness between the motor angles of the simulated system and the state space of the real system could be increased using the optimized set of parameters (*manual* and *ES*). The achieved alikeness between the real and the simulated system for the motor angle ϕ_Y exceeds the alikeness for the motor angle ϕ_X. This is due to a higher positioning precision of the corresponding real servo motor. For small values of k_R, the optimized parameter sets *manual* and *ES* achieve a smaller alikeness than the *standard* parameter set for ϕ_X. Closer inspection shows that the simulated trajectory for the *standard* parameters approaches the goal position more slowly and thus stays longer in the range of the real servos trajectory (that doesn't reach the goal position at all). Nevertheless, it can be seen that in general the ϵ-tunnel correctly quantifies the increased similarity of the behaviour of real and simulated system.



(a) $k_R = 1$



(b) $k_R = 3$

Fig. 5. The figure shows an example of the ϵ-tunnel for different values of k_R. The solid line represents the trajectory of the averaged motor angle ϕ_X of the real system. The dashed lines mark the upper and lower limit of the state space motor angle ϕ_X for different values of k_R and the dotted line displays the trajectory of the motor angle ϕ_X of the simulated system.

Table 2. Comparison of the percentaged alikeness of the motor angle ϕ_X of the simulation with the state space of the real system for different values of k_R.

Set of motor parameters used by the simulation	Alikeness of the motor angle ϕ_X of the simulation with the state space of the real system in % for $k_R = 1$	$k_R = 2$	$k_R = 3$	$k_R = 4$	$k_R = 5$
standard	13.74	31.30	42.75	45.67	47.84
manual	0.64	5.98	66.03	77.10	98.35
ES (ϕ_X)	0.76	5.34	73.79	77.35	98.47

Table 3. Comparison of the percentaged alikeness of the motor angle ϕ_Y of the simulation with the state space of the real system for different values of k_R.

Set of motor parameters used by the simulation	Alikeness of the motor angle ϕ_Y of the simulation with the state space of the real system in % for $k_R = 1$	$k_R = 2$	$k_R = 3$	$k_R = 4$	$k_R = 5$
standard	32.70	39.31	44.15	48.35	51.15
manual	87.40	95.17	98.35	99.11	99.75
ES (ϕ_Y)	88.67	95.67	98.60	99.75	100.00

5 Conclusion

In this work we have presented the ϵ-tunnel, a new method for quantifying the simulation-reality-gap between a robot and its corresponding simulated counterpart, that is based on the comparison of the state trajectories of the systems when the same action sequence is executed. This new method offers a high adaptability and flexibility. Depending on the chosen actions and the analyzed state trajectories, the behaviour of the whole system or just parts of it like the actuators can be compared. Furthermore it is possible to compare two systems even despite a large discrepancy between their behavior by introducing scaling factors k_R and k_S.

6 Future Work

In the future, the ϵ-tunnel should be utilized within an automatic calibration of both systems similar to the procedure described in [6]. For this, a set of selected action sequences will be executed on the real system and the state of the system will be recorded in parallel. Afterwards the parameters of the simulation will be optimized with regard to the percentaged alikeness between the simulation and the real system.

As a next step more advanced optimization methods like the exploration-estimation algorithm [13] developed by Josh Bongard and Hod Lipson could be used to optimize the parameters of the simulation. Furthermore, it could be

investigated if a decrease of the simulation-reality-gap quantified by the ϵ-tunnel correctly predicts an improved behavior of a controller that was optimized in the simulation on the real robot.

References

1. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning). The MIT Press (March 1998)
2. Miglino, O., Lund, H.H., Nolfi, S.: Evolving mobile robots in simulated and real environments. Artificial Life **2** (1995) 417–434
3. Römmermann, M., Edgington, M., Metzen, J.H., de Gea, J., Kassahun, Y., Kirchner, F.: Learning walking patterns for kinematically complex robots using evolution strategies. In: 10th International Conference on Parallel Problem Solving from Nature. Volume 5199 of Lecture Notes in Computer Science., Springer (2008) 1091–1100
4. Jakobi, N., Husbands, P., Harvey, I.: Noise and the reality gap: The use of simulation in evolutionary robotics. In: Advances in Artificial Life: Proc. 3rd European Conference on Artificial Life, Springer-Verlag (1995) 704–720
5. Jakobi, N.: Evolutionary robotics and the radical envelope-of-noise hypothesis. Adapt. Behav. **6**(2) (1997) 325–368
6. Metzen, J.H., Kirchner, E., Abdenebaoui, L., Kirchner, F.: Learning to play the BRIO labyrinth game. Special Issue "Reinforcement Learning" of the magazine "Künstliche Intelligenz" (2009) Accepted.
7. Lund, H.H., Miglino, O.: From simulated to real robots. In: Proceedings of IEEE 3rd International Conference on Evolutionary Computation, IEEE Press (1996)
8. Miglino, O., Nafasi, K., Taylor, C.E.: Selection for wandering behavior in a small robot. Artificial Life **2** (1994) 101–116
9. Nolfi, S., Floreano, D., Miglino, O., Mondada, F.: How to evolve autonomous robots: different approaches in evolutionary robotics. In: Artificial Life IV, MIT Press (1994) 190–197
10. Smith, R.: Open Dynamics Engine. (2006) http://www.ode.org/.
11. ROBOTIS: Dynamixel DX-113, DX-116, DX-117 User's Manual. 2nd edn. (11 2005) http://www.robotis.com/.
12. Beyer, H.G., Schwefel, H.P.: Evolution strategies – a comprehensive introduction. Natural Computing: an international journal **1**(1) (2002) 3–52
13. Lipson, H., Bongard, J.: An exploration-estimation algorithm for synthesis and analysis of engineering systems using minimal physical testing. In: ASME Design Automation Conference (DAC04). (2004) 1087–1093