Okinawa Institute of Science and Technology
Neural Computation Unit

Internship Report

Artificial Evolution of a Walking Controller for a Robot Dog

Chris Reinke

October 2010 - April 2011

June 2011

Contents

Acknowledgment

I would like to thank my supervisors in Osnabrück PD Dr. Helmar Gust and Prof. Dr. Stephan Evert and my dean of studies Prof. Dr. Achim Stephan. They gave me the opportunity to use the time at the OIST as part of my study project.

Furthermore I would like to thank my supervisors Dr. Kenji Doya and Dr. Eiji Uchibe for the possibility to work in their group at the OIST. The Neural Computation group has a cooperative and loyal working atmosphere which is accompanied by a high scientific standard. There members made my stay very pleasant and academically fruitful.

I also acknowledge the administration of the OIST which made my stay very convenient. Especially the students affairs staff helped me with any problem which I encountered.

1 Introduction

This document describes the conducted work and the activities during my internship at the OIST (Okinawa Institute of Science and Technology). My internship lasted six months from October 25, 2010 till April 16, 2011. My internship was mainly concerned with the new introduced spDog (spring dog) robots.[1] A controller for the walking behavior was implemented with help of a CPG (central pattern generator). Furthermore the walking was optimized for speed on the physical spDog and on the simulation of the spDog.

The first part of this report describes the spDog robots and their original simulator. Afterwards the implemented walking behaviors and used CPG get introduced. This is followed by the description of the implemented optimization processes and their results. The last part describes the introduction of a new simulator for the robots to accomplish a better adaptation of the simulation model to the physical robots.

The described work covers only the main points of the research during the internship. In addition I helped to test the spDog and to develop software components for it.

2 The spDog Robot System

The spDog (spring dog) robot system was introduced into the Neural Computation Unit to replace the CyberRodent Robots [2]. The spDog is manufactured by an external company and still under development. The robots are used to perform embodied evolution and reinforcement learning experiments. A wheel and a legged version of the spDog exists. The conducted work during the internship was performed with the legged version. Figure 1 shows the robot and depicts some of its parts. The robot consists of two main body elements which are connected by a servomotor in the transverse plane. The legs are connected to the front and rear body element by servomotors which operate in the sagittal plane. Each leg consists of a hard spring which is bend by 90 degree at the location of the knee joint. Furthermore a head component exists which can be controlled with two degrees of freedom. Thus seven degrees of freedom exist for the whole robot. The spDog has several sensors: the joint angular positions for each servomotor and the three dimensional accelerometer and gyrometer which are attached to the front body element. The legs have a touch sensor for the foot and a touch sensor for side forces onto the body elements. A USB camera is attached to the head. The nose part of the head is able to attach to battery packs which reload the battery of the robot. Moreover a serial connection between two spDog's can be established when their nose parts have physical contact to each other. Each robot has an embedded CPU board with a Linux operating system on which controllers can be executed. Communication between a remote computers and the robot is possible via wireless LAN or a serial cable connection.

A simulator was implemented by Eiji Uchibe which is based on the ODE[2] (Open Dynamics Engine) and the wxWidgets library[3]. The simulator is in an early development stage and provides only a simple programming interface to control the simulated robot. The properties

[1] www.nc.irp.oist.jp
[2] www.ode.org
[3] www.wxwidgets.org

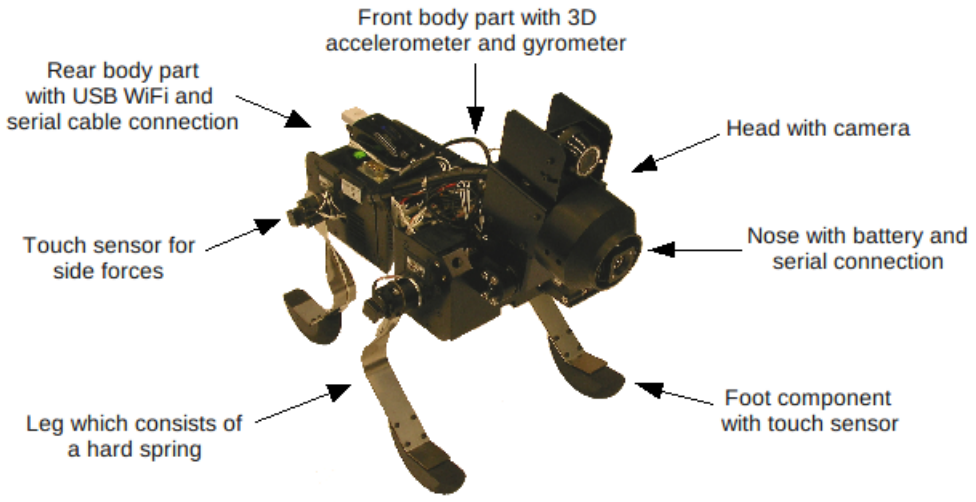


Figure 1: Legged spDog with main components. The robot has a height of 80 cm, a width of 20 cm and a length of 75 cm.

of physical spDog were only coarse modeled which results in a different behavior of the simulated robot and the physical robot for the same motor commands over time. This problem is demonstrated and further discussed in Chapter 4.3.

3 Walking Behaviors

The first task for the work with the legged spDog was to implement a controller for a forward walking behavior. The controller generates angular position commands for the leg servomotors over time. The legs have to oscillate in a certain pattern to generate a forward movement. The movement is defined by the characteristics of the oscillation of each leg and their coupling, i.e. if they oscillate in phase or out of phase. A CPG was used to generate the motor commands which gets described in the next part. It is followed by a description of the implemented controllers which use the CPG for the walking behaviors.

3.1 Central Pattern Generator

A CPG (central pattern generator) is used to generate the motor commands for the walking behavior. CPG's can be found in invertebrate and vertebrate animals. They are neural circuits which can produce a rhythmic neural activity without a rhythmic input. The term central originates from the fact that they need no sensory feedback to produce their rhythmic activity. The concept of CPG's were introduced into the field of robotics for rhythmic movement generation. They are mostly modeled as dynamical systems. See [4] for an overview about CPG's in animals and robots.

The used CPG is defined by Rhigett and Ijspeert [6] and is composed of four coupled modified Hopf oscillators. See Figure 2 for an illustration. An oscillator exists for each leg which generates

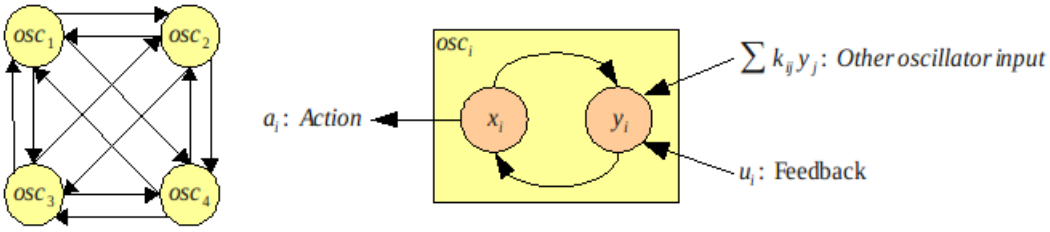


Figure 2: (left) Schematic view of the CPG with its coupled oscillators. The output of the oscillators 1 and 2 is used for the front left and right leg and 3 and 4 for the rear legs. (right) A single oscillator with its components.

the motor command for the angular position of the corresponding leg servomotor.

Each oscillator consists of a x and y component. The dynamics of the oscillator is given by the following dynamical system:

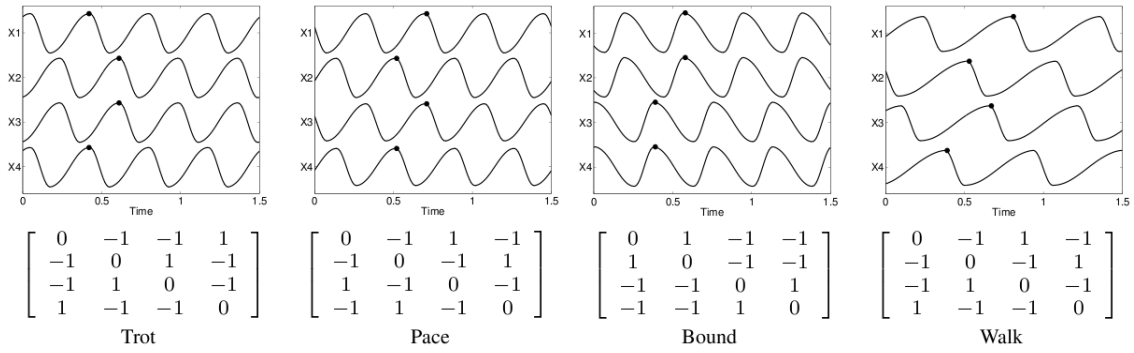
$$\begin{aligned} \dot{x}_i &= \alpha(\mu - r^2)x_i - \omega y_i \\ \dot{y}_i &= \beta(\mu - r^2)y_i - \omega x_i \\ r_i &= \sqrt{x_i^2 + y_i^2} \\ \omega_i &= \frac{\omega_{stance}}{e^{-by_i}+1} + \frac{\omega_{swing}}{e^{by_i}+1} \end{aligned}$$

μ defines the amplitude of the oscillation. The ascending and descending phases and the frequency of the oscillation can be defined by ω_{stance}, ω_{swing} and b. The parameters α and β control how fast a single oscillator reaches its limited cycle.

The single oscillators are coupled by their y component to each other. The coupling matrix K defines the coupling between the i'th and j'th oscillator. This coupling defines if the oscillation is in phase or not. Different gait behaviors can be expressed by different matrices. Figure 3 shows four different configurations and their behavior. For example for the trot gait the left front (X1) and right rear leg (X4) oscillate in phase. This holds also for the right front (X2) and left rear leg (X3), but they are in anti-phase to X1 and X4. To achieve in-phase oscillations between two oscillators the connection between them has to be positive and symmetric ($k_{ij} = +1$, $k_{ji} = +1$). Anti-phase oscillations are defined by negative and symmetric connections ($k_{ij} = -1$, $k_{ji} = -1$). To generate gaits where the oscillation is in a non-phasic way the connections have to be asymmetric (e.g. $k_{ij} = +1$ and $k_{ji} = -1$). That the resulting oscillation shows its desired behavior and that they are stable is mathematical proven in [3]. The walking gait showed the most promising results during the first experiments with the physical spDog . Thus the further conducted work was only concerned with the walking gait.

Feedback from the robot sensors is given to the y - components of the oscillators. This option was not used during the experiments. A problem on the robots made it impossible to send motor commands and retrieve sensor information at the same time. Thus only open loop controllers were implemented. Rhigett and Ijspeert [6] describe the usage of sensor information to improve the robustness of the walking behavior.

During the work with the CPG a disadvantage was discovered. The CPG needs up to 5000 simulation steps before it reaches its limited cycle attractor when it starts from the recommended



$$\text{Trot}: \begin{bmatrix} 0 & -1 & -1 & 1 \\ -1 & 0 & 1 & -1 \\ -1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 0 \end{bmatrix} \quad \text{Pace}: \begin{bmatrix} 0 & -1 & 1 & -1 \\ -1 & 0 & -1 & 1 \\ 1 & -1 & 0 & -1 \\ -1 & 1 & -1 & 0 \end{bmatrix} \quad \text{Bound}: \begin{bmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{bmatrix} \quad \text{Walk}: \begin{bmatrix} 0 & -1 & 1 & -1 \\ -1 & 0 & -1 & 1 \\ -1 & 1 & 0 & -1 \\ 1 & -1 & -1 & 0 \end{bmatrix}$$

Figure 3: Coupling matrices for different gaits. $\omega_{stance} = 2\omega_{swing}$ for the trot and pace gaits. $2\omega_{stance} = \omega_{swing}$for the bound gait and for the walking. Other parameters: $\mu = 1$, $\alpha = 5$, $\beta = 50$. The figure is taken from [6].

standard initial values for the x_i, y_i, $\dot{x}_i$ and $\dot{y}_i$ components. To avoid the problem of unwanted behavior in the initial time for new CPG parameters the CPG is simulated for 5000 steps before the output is used to control the motors. To reduce this initial simulation time for known CPG parameters the CPG gets directly set into their known attractor.

3.2 Behaviors

Based on the CPG two basic behaviors were implemented. The first is a simple walking which can be controlled by a direction and a velocity setting. The second behavior is based on the first and is a search behavior. The robot tries to find an object with a specific color in its environment and then walks towards it.

To accomplish a simple walking, it was necessary to find CPG parameters which provide a stable forward movement. Table 1 lists the parameters which were manually identified by trial and error experiments. Furthermore a method to set the direction and the velocity of the movement had to be implemented because this is not described for the CPG of Rhigett and Ijspeert [6]. A simple solution is used which changes the amplitude of the generated CPG motor signals x_i by a coefficient c_i: $m_i = c_i \cdot x_i$. The amplitude of all oscillators gets adjusted in the same manner to regulate the velocity. To control the direction, the amplitude of the legs on the side to which the robot should walk gets reduced. The resulting behavior was stable and the velocity and the direction of the robot was controllable. Nonetheless the maximum walking speed was not high and the robot turned in a circle with a diameter of approximately 60 cm.

The search behavior was implemented to present the spDog during the OIST “Open Campus Day” to a public audience. The simple walking is used as basis and gets extended by a visual search component which was already implemented by Dr. Eiji Uchibe. The task of the visual component is to find objects with a certain color on the camera images. This information is used to turn the head of the robot into this direction to focus the object in the middle of the camera view. The visual component gets executed in parallel to the walking component on the robot. The direction setting for the walking is defined by the turning angle of the robot head in the coronal plane. When the robot turns its head into a certain direction, the walking gets directed into this direction. The velocity gets controlled by the number of pixels of the target

Parameter	Value
μ	1.75
ω_{stance}	16.0
ω_{swing}	4.0
b	50.0
α	10.0
β	20.0

Table 1: Manually identified parameters for the spDog walking gait.

color which are detected in the camera image. If only a few pixels are seen the velocity is high and gets reduced to zero if enough pixels are visible. Thus the robot walks fast if the object is far away and stops in front of the target object instead of bumping into it. In general the resulting behavior is stable. Sometimes misclassifications of colors occure.

4 Evolutionary Experiments

The experimentally detected CPG parameters for the walking behavior result in a stable but slow movement of the robot. An evolutionary algorithm was used to optimize the parameters to generate a faster movement. The optimization was conducted on the physical robot and the simulated robot. The used Evolutionary Strategies algorithm is summarized in the next part. Afterwards the experiments and their results will be described.

4.1 Evolution Strategies Algorithm

This section describes the Evolution Strategy (ES) algorithm which goes back to the work of Rechenberg (See [5]). The implemented $(\mu/\rho + /, \lambda)$-ES (Mu-Slash-Rho-Plus-Or-Comma-Lambda Evolutionary Strategy) improved version of the algorithm is described in [1].

The goal of the algorithm is to optimize a fitness function $F(o)$. The arguments o of the fitness function are called object parameters. The ES tries to find object parameters o which maximize the fitness function F.

ES operates on populations of parent $\mathfrak{P}_p^{(g)}$ and offspring $\mathfrak{P}_o^{(g)}$ individuals $\mathfrak{a}$. Each individual is defined by a set of object parameters o and their resulting fitness value $F(o)$. Moreover each individual holds a set of endogenous strategy parameters s which are an important part of the implemented algorithm. They control the statistical properties of the mutation operator for the object parameters. Endogenous means that the strategy parameters evolve and change during the evolutionary process. This endogenous property was chosen to avoid the problem of setting the strategy parameters by hand. The evolution should find the optimal parameters by itself. Furthermore exogenous strategy parameters exist which are fixed during the whole process. These parameters control statistical properties of the mutation for the endogenous strategy parameters s and are the same for each individual.

The ES routine (See Algorithm 1) starts with a parent population $\mathfrak{P}_p^{(0)}$ of μ individuals. The

Algorithm 1 $(\mu/\rho + /, \lambda)$-ES

1: $g := 0$
2: initialize$\left(\mathfrak{P}_p^{(0)} := \left\{\left(o_k^{(0)}, s_k^{(0)}, F(o_k^{(0)})\right), \quad k = 1, \ldots, \mu\right\}\right)$
3: **repeat**
4: **for** $i := 1$ **to** λ **do**
5: $\mathfrak{E}_i :=$ marriage $\left(\mathfrak{P}_p^{(g)}, \rho\right)$
6: $s_i :=$ s_recombination $(\mathfrak{E}_i)$
7: $o_i :=$ o_recombination $(\mathfrak{E}_i)$
8: $\tilde{s}_i :=$ s_mutation (s_i)
9: $\tilde{o}_i :=$ o_mutation $(o_i, \tilde{s}_i)$
10: $\tilde{F}_i := F(\tilde{o}_i)$
11: **end for**
12: $\mathfrak{P}_o^{(g)} := \left\{\left(\tilde{o}_i, \tilde{s}_i, \tilde{F}_i\right), \quad i = 1, \ldots, \lambda\right\}$
13: **case** selection_type **of**
14: $(\mu, \lambda):$ $\mathfrak{P}_p^{(g+1)} :=$ selection $\left(\mathfrak{P}_o^{(g)}, \mu\right)$
15: $(\mu + \lambda):$ $\mathfrak{P}_p^{(g+1)} :=$ selection $\left(\mathfrak{P}_o^{(g)}, \mathfrak{P}_p^{(g)}, \mu\right)$
16: **end case**
17: $g := g + 1$
18: **until** termination condition

object parameters for each individual are randomly assigned or given by the user. All strategy parameters are set to a user defined value (1 was used for the conducted experiments). At first the fitness value for each parent individual gets calculated. Afterwards the ES routine works in a loop where each iteration reflects a certain generation g of parent and offspring individuals. The loop consists of two major steps. The first step is the reproduction step (line 4 until 12) which generates the offspring population $\mathfrak{P}_o^{(g)}$ with λ new individuals from the parent population $\mathfrak{P}_p^{(g)}$. The second step (line 13 until 16) selects the individuals from the populations $\mathfrak{P}_p^{(g)}$ and $\mathfrak{P}_o^{(g)}$ which are used as parent population $\mathfrak{P}_p^{(g+1)}$ for the next generation.

The reproduction step consists of a loop with several operators. The loop is iterated λ times and each iteration creates a new offspring. The marriage operator (line 5) selects ρ individuals from the parent population $\mathfrak{P}_p^{(g)}$. The selection is completely random and does not depend on the fitness value $F(o)$ of the parents. The selected individuals are the parental pool $\mathfrak{E}$ for one offspring. The recombination operators (line 6 and 7) recombine the object and strategy

parameters of the parents $\mathfrak{E}$. An intermediate recombination operator was used which uses the mean of all parent parameters for the parameter of the offspring. The mutation operators (line 8 and 9) introduce changes to the recombined strategy and object parameters. The mutation of the object parameters is controlled by the strategy parameters. The mutation step generates a random number by a Gaussian distribution with $\mu = 0$ for each object parameter. The number is then added to the parameter value. For each object parameter exists a strategy parameter which defines the variance σ for the Gaussian distribution. Furthermore the strategy parameters σ_i get mutated. The mutation is different to the object parameter mutation and defined by the equation $\tilde{s}_i = s_i \cdot \exp(\tau \cdot \mathcal{N}_1(0,1))$. τ is a fixed exogenous strategy parameter which was set to 1 for all performed experiments. $\mathcal{N}_1(0,1)$ defines a random number from a unfirom distribution beween 0 and 1. Afterwards, the fitness value $F(\tilde{o})$ of the new individual is being calculated. The complete reproduction step results in an offspring population $\mathfrak{P}_o^{(g)}$ with λ new individuals.

The selection operators (line 14 and 15) select the individuals from the current generation which are used as parent population $\mathfrak{P}_p^{(g+1)}$ for the next generation. The selection gives the evolution a certain direction. It is the antagonist to the variation operator of recombination and mutation which provide random changes to the individuals. Only the fittest individuals get selected for the next generation. Thus, individuals with a good performance have a higher probability to be chosen as parents and to hand over their object and strategy parameters to the offspring. The current work uses the plus version of the selection operator where the μ best individuals from the parent population $\mathfrak{P}_p^{(g)}$ and the offspring population $\mathfrak{P}_o^{(g)}$ are used for the next generation.

The principle of the ES loop is to produce many generations of individuals and that the fitness value of the individuals increases over the generations. The loop stops if a certain number of generations has been reached.

4.2 Evolution on the Physical spDog

The ES Algorithm was used to optimize the CPG parameters which are listed in Table 1 on the physical spDog. To perform an optimization, a new behavior had to be implemented which measures the fitness value for certain object parameters. The description of the new measurement process is followed by the results of the optimization.

The fitness is measured by the time which the robot needs to walk a certain distance. The robot is walking in a square to avoid that he has to be set manually back to its start position. Four colored patches on the floor define the square. The search behavior described in Section 3.2 is used to accomplish that the robot is walking from one patch to the other and to search afterwards the next target patch. See Figure 4 (left) for an illustration. The resulting fitness is defined by $F(o) = \frac{1}{t}$ where t is the mean time calculated from four successfully measurements, i.e. the robot walked succesfully four times from one patch to another. The fitness is set to -1 when the target patch is not reached in a certain time or the robot lost the contact to the target patch on its camera image before reaching it. After such an error, the robot switches back to the standard walking parameters and searches the next target to start a new measurement from there. The behavior was stable and could run without supervision. Nonetheless sometimes parameters appeared which produced a very unstable walking behavior. The robot felt to the side and had to be put back manually.

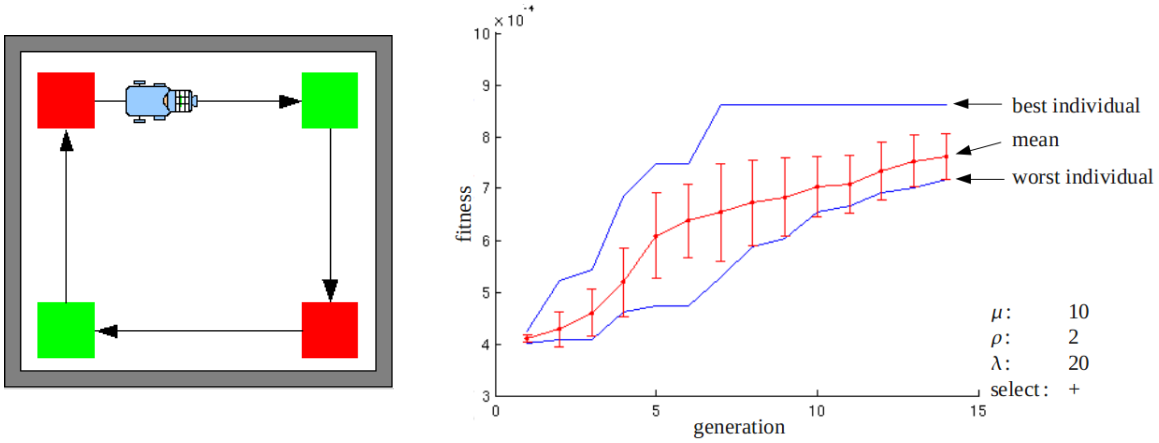


Figure 4: (left) Schematic view from above of the measurement environment for the physical spDog. The robot walks from one colored patch to the other. The needed CPG update steps are measured and used to define the fitness. (right) Results of the ES optimization run on the physiscal spDog. The fitness of the best and worst individual (blue) and the mean fitness for each generation with the standard deviation (red) is given.

One ES optimization was performed on one of the spDogs with $\mu = 10$, $\rho = 2$ and $\lambda = 20$. The object parameters of all initial parents where set to the standard parameters. The optimization was performed for 13 generations which resulted in the measurement of 270 individuals.

The results are illustrated in Figure 4 (right). The best evolved individual showed a performance twice as high than the standard parameters. Nonetheless the optimization was very time consuming because each fitness measurement needed approximately four minutes which resulted in 18 hours for all 270 individuals. Unfortunately a deeper analysis of the behavior of the evolved parameters was not possible because one of the motors break after the 18 hours of usage.

4.3 Evolution with the spDog Simulator

After the physical spDog was damaged the evolutionary experiments were conducted in the simulator. Another reason for the usage of the simulator is the problem that evolutionary experiments with the physical system take a lot of time. This part describes at first the evolutionary experiments and their results. Moreover qualitative results are given for the application of the evolved CPG parameters on the physical robot. The results show a problem which arises with the usage of the simulator. A simulation is only a model of the physical world. Thus the behavior of the simulated and the physical robot differs when they execute the same motor commands. If the difference is too high it is not possible to use evolved controllers from the simulator for the physical robot.

The CPG parameter evolution in the simulator differs slightly from the evolution on the physical robot. The fitness measurement gets easier because the robot can be reset to its start location and the measurement of exact distances is possible. Thus the fitness was changed to the distance the simulated robot can walk in a certain direction for a given time t. Only one

measurement is needed for each CPG parameter set because the simulator has a deterministic behavior. After each measurement the simulated spDog was reset to its start position for the next measurement.

One ES optimization was performed with $\mu = 10$, $\rho = 2$ and $\lambda = 100$. All start individuals were initialized with the standard parameters. The evolution was running for 63 generations which needed 6240 fitness measurements and could be done in 12 hours. The obtained results are illustrated in Figure 5 (right). CPG parameters could be evolved which had more than twice of the performance ($F = 5.36$) of the standard parameters ($F = 1.62$).

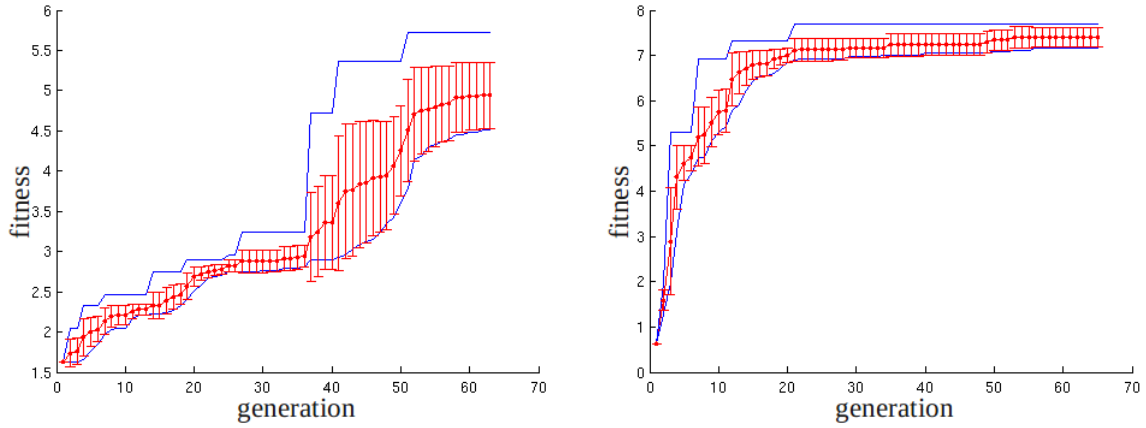


Figure 5: Results for two ES optimizations with the simulator ($\mu = 10$, $\rho = 2$, $\lambda = 100$). The results show the fitness of the best and worst individual (blue) and the mean fitness for each generation with the standard deviation (red). Different contact point frictions are used in the simulator for the ES optimizations: left = 0.0035, rigth = 0.01.

After the physical spDog was repaired, the evolved CPG parameters where tested on it. The best parameter set produced a stable walk without falling. Figure 6 shows the results of a fitness measurement with the behavior described in Chapter 4.2. Five single measurements were conducted for the standard parameters and the evolved parameters. The mean performance for both parameters is similar but the variance of the evolved parameters is higher. Thus the evolved behavior is unstable and has no speed advancement on the physical spDog. Moreover other evolved parameters which showed a higher performance than the standard parameters in the simulator produced an unstable walking behavior. As a consequence the robot felt to the side. These results show that the current version of the simulator is insufficient to reproduce the behavior of the physical spDog for the same motor commands.

One further evolution was conducted with a higher friction coefficient for the contact points between the robot and the floor in the simulation. Figure 5 (right) shows the results of the evolution. It was also possible to evolve CPG parameters which produced a stable and faster movement in the simulator. The two best evolved CPG parameters sets were tested on the physical spDog. Both sets produced an unstable behavior which let the robot fall to the side. To summarize the results of both evolutionary experiments with the simulated spDog the simulator needs to get better adapted to the physical spDog to generate useful results.

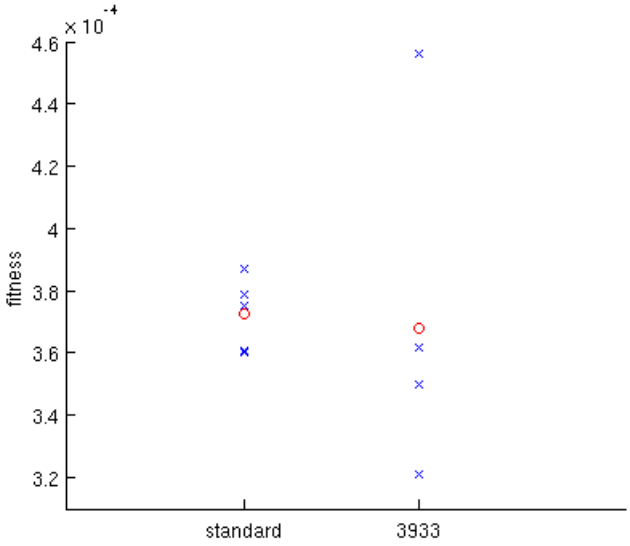


Figure 6: Fitness of the best evolved individual in the simulator (id = 3933) and the standard parameters (See Table 1) on the physical spDog. The crosses depict single measurment and the circle the mean fitness.

5 Nerd spDog Simulator

The unsuccessful application of the evolved CPG parameters in the simulator for the physical spDog showed that the simulator needs to get adapted to the properties of the physical spDog. This has to be done by adapting the simulator parameters like friction or damping settings. Done manually this process is very time consuming. Moreover the spDog simulator is in an early development version and the change of the simulator settings is not possible via an user interface. I proposed to the Adaptive Systems Group to use the simulator from the NERD toolkit[4]. The toolkit was developed by Christian Rempis for the Neurocybernetics group of the University of Osnabrück. NERD is a simulator for robotic systems which can be controlled by neural networks. Furthermore evolutionary tools exists to evolve neural network controllers. The NERD simulator is based on the same physical library as the spDog simulator. Its advantage is that it is in a higher development stage and has an easier user interface and many useful tools. Another advantage is that the NERD Model Optimizer tool exists which adapts the NERD simulator automatically to the physical properties of a robot. The tool was written as part of my bachelor project. To show the abilities of the NERD simulator a coarse model of the spDog was implemented for it. It holds already all joints and motors and the foot touch and accelerometer sensors. The NERD simulator is now used by the Adaptive Systems Group and the NERD Model Optimizer will soon be used to adapt the spDog model.

[4] www.ikw.uni-osnabrueck.de/~neurokybernetik/tools/nerd

References

[1] H.G. Beyer and H.P. Schwefel. Evolution strategies - a comprehensive introduction. *Natural Computing*, 1(1):3–52, March 2002.

[2] K. Doya and E. Uchibe. The cyber rodent project: Exploration of adaptive mechanisms for self-preservation and self-reproduction. *Adaptive Behavior-Animals, Animats, Software Agents, Robots, Adaptive Systems*, 13(2):149–160, 2005.

[3] Martin Golubitsky, Ian Stewart, Pietro-Luciano Buono, and J. J. Collins. A modular network for legged locomotion. *Phys. D*, 115:56–72, April 1998.

[4] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4):642 – 653, 2008. Robotics and Neuroscience.

[5] I. Rechenberg. *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution*. PhD thesis, 1973.

[6] L. Righetti and A.J. Ijspeert. Pattern generators with sensory feedback for the control of quadruped locomotion. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 819–824. IEEE, 2008.

[7] G. Schöner. Dynamical systems approaches to cognition. *Cambridge Handbook of Computational Cognitive Modeling*, pages 101–126, 2007.

[8] L. Wasserman. *All of statistics*. Springer New York, 2004.