Evolving a Neurocontroller for a Fast Quadrupedal Walking Behavior

Diplomarbeit in Computer Science

submitted
by

Irene Markelić

born 13.06.1979 in NRW, Germany

Written at

Institut für Computervisualistik
Arbeitsgruppe Aktives Sehen
Universität Koblenz–Landau

and at
Fraunhofer Institut
für Autonome Intelligente Systeme
Sankt Augustin

Advisor: Prof. Dr.-Ing. D. Paulus and Prof. Dr. F. Pasemann

Started: 01.09.2004

Finished: 30.05.2005

Abstract

In order to develop a fast walking behavior for a specific quadruped robot, a controller consisting of a neural network is generated by using Artificial Evolution. To achieve this an ODE-based physical simulation of the robot is built. The evolved controller is verified by transferring it successfully to hardware. Afterwards it is simplified and examined to find out how the observed behavior is accomplished. This is driven by the desire to discover possible principles underlying walking in general. The outcome of this work is a comparably fast gait produced by a small controller, concerning the amount of neurons and synapses of the network.

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Richtlinien des Instituts für Studien- und Diplomarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts

Koblenz, den 28.05.2005 Unterschrift

Contents

Chapter 1

Introduction

The motivation for this work is twofold. One reason is the desire to find out more about the principles underlying walking. It is believed that this way one could gain more insights about the functioning of brains and use this knowledge to build an intelligent machine. This is the long-term goal setting the general frame of this work. The second reason is more pragmatic. Although many legged robots have been built, there is still no satisfactory device available to control them, concerning speed, adaptivity, and energy-consumption. Honda's robot Asimo, which represents state-of-the-art moves with about 2 km/h. Its batteries last about 20 minutes and the stairs it climbs must be especially prepared [wal]. This lack of a truly autonomous and adaptive walking behavior heavily narrows the possible areas of application for these machines. Legged robots have the great potential advantage to move in environments where wheeled ones cannot be utilized, for example in such that contain obstacles or where ground-damages must be avoided.
The price for the greater flexibility of legged machines is the need for a more complex control. A great number of degrees of freedoms (DOFs) has to be coordinated in such a way that it results in a translation of the robot's body over the ground. Furthermore the robot should be able to perform in the real world, not laboratory conditions, but with humps, wholes, limited energy supply etc.. Since foreseeing all possible pitfalls exceeds human capabilities it is not feasible to write a program of a walking behavior that could cope with every situation and run it on the robot. Instead it is desired that the robot takes

care of these things itself – autonomously. A control that is reactive to the determining factors such as the environment, the robot's body itself, or a specific demand like fast or very stable walking, is therefore needed.

This work does not aim at developing an adaptive walking behavior, but with a special subtask of it, namely with finding a particularly fast gait.
For this task a commercially available quadrupedal robot platform, namely the pet-dog "Aibo" [aib05a] of Sony is chosen. As an off-the-shelf solution much research has been conducted using this machine. Numerous publications dealing with walking behaviors for the Aibo are available, e.g. [BI00], [HSYF05], [ZON02]. The Aibo is also used for the Four-Legged-League in Robot Soccer [Rob] and as basic behavior fast walking is of concern of nearly every team, see for example [CV04], [Rö04],or [KS04]. Since the hardware platform is always the same the behaviors are well comparable which facilitates the evaluation of one's own results.

1.1 The Control of Legged Machines

The aim of this section is to provide the reader with an overview of the most important control paradigms for gait-generation in legged robots and the milestones that had lead to them.
The term "gait-generation" refers to the "formulation and selection of a sequence of coordinated leg and body motions that propel a legged robot along a desired path" [WT92]. Hence, a "gait" denotes the according *sequence*. An alternative definition of a gait is given in section 3.1.

1.1.1 Historical milestones

- Automata:
  The first effort to construct legged machines was made in the mid 18th century. Automata were constructed by J. de Vaucasson, P.-J. Promond, and H. Lois to mimic life and were demonstated at fairs and exhibitions. Bellows and levers were used to control them [Ree99].

- Mechanical Kinematic Linkages:
  In 1850 the Russian mathematician Chebyshev invented mechanical kinematic linkages. They connected joints so that they moved together and made the machine move along a horizontal path. The Mechanical Horse patented by Lewis A. Rygg, although never actually built, is a famous example. It took almost 90 years until this technique was found to be too restrictive [Rai86].

- Human-Control:
  In the mid 1960's Ralph Mosher at General Electric built the "versatile walking truck" [LM68]. This is an impressive four-legged machine harnessing a human as control unit. The driver was seated in a cabin on top of the machine controlling its legs with the help of handles or pedals. This form of control was very demanding on the driver and actually ignored the control-problem.

- Digital Computer:
  The next milestone was the first use of digital computer control in 1977. Robert McGhee used it primarily to solve kinematic equations for his insect-like hexapod.

- Passive Control:
  In 1990 Tad McGeer showed that much of our walking is produced by mere swinging movement of the legs. He let a biped, which was not more than a planar mechanism with two legs, walk down a slight slope. This was possible due to carefully tuned weights, and needed no other control or energy input.

1.1.2 A Classification of Computer-Based Approaches

With McGhee's work, using computers as control for gait generation started. They have been employed in several ways. Although often used in combination they can be distinguished as follows.
The taxonomy presented here is partly based on the classification proposed by Wettergreen [WT92] and does neither claim completeness nor uniqueness since there are many other possibilities to structure the various approaches existing in robot control.

Rule-Based

In this approach a given gait is described by a plan in the sense of classical Artificial Intelligence. The gait/plan is generated by applying a set of rules or heuristics to a world model (a symbolic representation of the world).
This kind of analyzing perception, planning and action independently of each other is called *functional decomposition* [Har93].

The Adaptive Suspension Vehicle by Waldron and Song ([SW89]) is controlled in such a way, as well as the Titan robots by Hirose and Yoneda [HFK86].

Because a symbolic representation cannot describe the real world adequately, the plan which is generated on top of this model can never be sufficiently robust and general. This

is the major drawback of this approach.
Another disadvantage is that the gait described by a plan has to be there first, i.e. some*one* has to decide which kind of gait to use. This might lead to suboptimal results, since it cannot be known if the chosen gait is optimal for a given machine.

Constraint-Based

In this approach the range of possible moves is predefined. This set of movements is constrained by certain factors such as kinematics, terrain and stability. A gait can be designed from the remaining possible moves [WT92]. This can be done manually or by applying automated search- and optimization methods as in many of the walking behaviors developed for the Four-Legged Robot League in Robocup [Rob], e.g. [KS04] or [KU03]. Designing a gait manually can, like the rule-based approach, lead to a suboptimal result. Another disadvantage is the potentially great search space, scaling it down can also reduce the quality of the resulting gait.

Control-Based

Controllers associated with this approach use control-theory to regulate variables, e.g. balance, which can lead to gait-generation as a by-product. This kind of gait generation is used in most bipeds today [WT92]. To produce a fast and dynamically stable gait control-theory was first employed by Marc Raibert in 1986 ([Rai86]). The idea was inspired by the classical pole-balancer, that maintains stability by using the appropriate feedback. With a controller that was divided into three subparts and used certain feedback, Raibert enabled a one-legged machine to keep its balance while traveling at a specified rate. This was possible because the machine contained a springy leg, that allowed it to bounce. The three parts seperately controlled hopping motion, forward travel, and posture of the body. A finite state machine synchronized them. No classical planning was used and the control was comparably simple. Later he expanded this approach successfully to other machines, including a quadruped.

Behavior-Based

This approach can be seen as contradistinction to the rule-based approach, since it rejects the view of functional decomposition. Instead it adopts that biological systems produce gaits without conscious reasoning [WT92]. It brings together biological science and robotics by incorporating principles and insights found in the field of non-linear dynamics, neurobiology, zoology, evolutionary biology and other "life-based" sciences. Control networks as regulating device are used. These can be artificial neural networks (ANNs) designed after a nervous system (see section 2.1.2) which are considered to control behavior in organisms.
The first who broke with functional decomposition was Brooks in 1986 with the subsumption architecture [Bro89]. This was based on the idea that many parallel and loosely coupled processes that connect sensors to actuators can control an agent. Others, that adopt biological aspects in their work of legged vehicle control are e.g. Beer [BQRC97], Kimura [KFK01], Dillmann ([lAD00]), and Billard and Ijspeert [BI00]. The advantages of this paradigm are that no world model is needed and no complicated formulas have to be evaluated as can occur in the constraint based approach. Furthermore it constitutes a good platform to test biological hypotheses and to even generate them. A drawback is that no process exists that guarantees the correct synthesis of such a controller. The reader is referred to [Leg] and [Kim] for further information about milestones in legged robot development or an overview about existing robots.

1.2 Applying the Behavior-Based Approach

As could be seen in the previous section one way of generating controllers for a walking behavior, is to provide an algorithm (rule-based and constraint-based approach). It is argued here that this is restrictive, because it contains much implicit knowledge. Since all humans have a certain understanding of how a particular behavior should be carried out they are likely to implement what they believe is best. Thus, man made solutions are always biased. That does not mean that they necessarily perform worse than other solutions, but that others which could perform better will not be found. In this sense the rule-based and, under certain conditions, also the constraint-based approach are a restricted form of control. Nevertheless it has been commonly used for Aibo-walking. In [Dü04] the creation of a walking behavior which is predetermined is described. A similar approach was conducted in 2000 by [HDI00]. It was the winning team of the 2000 RoboCup Tournament, and their suggested walking behavior was widely adopted and improved [Dü04].

This work aims at generating a fast walking behavior for a quadruped, the Aibo-robot, that is not limited by the inevitable bias of designed solutions. A technique that admits this is Artificial Evolution which was employed to generate a neural network as control unit – a so-called *neurocontroller* (see section 2.1.2).

To save time and prevent the hardware from damages Artificial Evolution was not directly carried out on the physical robot but instead on a computer simulation of it, which was especially built for this purpose as described in section 2.4.

1.3 Outline

This work is structured as follows:

- In chapter 2 the used concepts, tools and techniques such as behavior, neurocontroller, recurrent neural networks, Artificial Evolution, and the simulation of the robot, are motivated and elucidated.
- In chapter 3 is shown how these tools were employed to achieve the desired controller. Also approaches are pointed out to that did not lead to a satisfactory result but nevertheless contributed to the outcome of this work.
- In chapter 4 the best performing controller generated by Artificial Evolution is investigated. It is shown how this network can be simplified and improved.
- The results presented in the previous chapter as well as the used methods are discussed. General conclusions are drawn that can contribute to future work. Finalizing a summary of the work is given.

Chapter 2

Methods

To accomplish the stated goal of this work several techniques and tools were necessary. What these are, why and how they were used is content of this chapter.

2.1 Basic Concepts

The use of neurocontrollers is motivated by the Behavior-Based approach in combination with the Dynamical Systems approach. In the following these concepts and their relations to each other are described.

2.1.1 Behavior

A *behavior* is the interaction of a body with its environment. For interaction, a body must dispose of means to perceive and alter the environment. These are sensors and actuators. Informations about the environment are perceived by sensors and passed to the sensory system where a preprocessing takes place. This includes filtering and transferring these

information into appropriate input for the next entity where they are send to afterwards. This unit, where the main processing takes place, is referred to as *cognitive system*. It processes the received information and sends according output to the motoric system. All three systems are made of neural structures and the information flow between them is bidirectional, i.e. the cognitive system can also send information to the sensory system and receive them from the motoric system.
This concept is called *sensory-motor loop* [Pas96] and is visualized in figure 2.1.

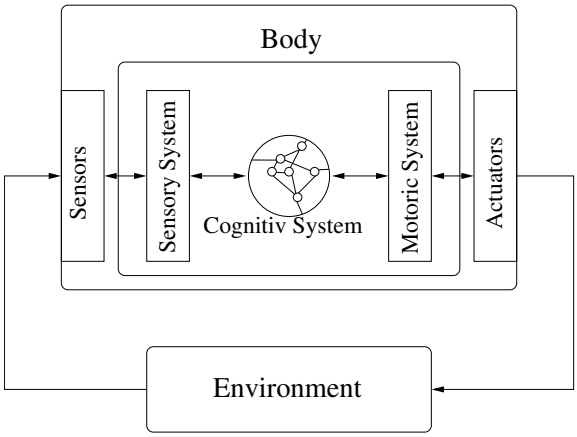

Figure 2.1: Neuro-Sensory Loop. Adapted from [Mah03].

As a consequence, a behavior is initiated by the cognitive system, but cannot be considered isolated from the body, in which it is embedded, and the environment. In other words: "[...] behavior emerges from interactions of nervous system, body and environment" [CB97].

2.1.2 Behavior and Dynamical Systems Approach

Examining behavior-generation in biological systems is difficult. It is assumed to be more effective to investigate an abstraction of the (biological) cognitive system instead. This abstraction must, of course, contain the relevant features of the biological example. The Dynamical Systems approach assumes these features to be the dynamic character of such a system. The employment of recurrent artificial networks as modeling tool is motivated by this approach.

"The brain is a self-organized, pattern-forming, dynamical system" (p.285 [Kel95]) which is non-linear [EFL+04]. As such, it can be described in the language of dynamical systems theory. The Dynamical Systems approach indicates that when an artificial neural system is to be built "the mechanisms can be considered as composed of a finite number of interacting components [...]" [HPW+05]. This is the case when employing artificial neural networks to model a neural system, as e.g. suggested in chapter 5 in [PS99]. For instance, a recurrent artificial neural network (RNN) based on sigmoidal transfer functions also states a non-linear dynamical system, and dynamics observed in brains, like chaos, bifurcation and hysteresis, can also occur in these systems. This has been shown for minimal systems [Pas93]). Due to this relation RNNs are considered to be an adequate tool to model neural systems.
Furthermore it is assumed that understanding the dynamics, within such an artificial network that lead to a behavior, may contribute to a broader understanding of behavior generation in organisms.

2.1.3 The Neurocontroller

In the following a *neurocontroller* is an artificial network acting as cognitive system within a body (compare to section 2.1.1). A body is considered to have a spatial extension greater than zero and to obey the laws of physics. This includes robots and physical computer simulations of such. Furthermore the body must dispose of sensors and actuators.
The in- and output neurons of the neurocontroller can be considered to be part of the sensory resp. motoric system. Sensors deliver raw input data, which is transformed into suitable network input during a preprocessing. Accordingly, network output is transformed into actuator signals during a postprocessing.
Section 2.5 describes how the neurocontroller was embedded in the Aibo-robot.

2.2 Neuron Model and Networks

Artificial neurons and synapses make up artificial neural networks. They are inspired by biological neurons, which serve as information processing units in neural systems. A great range of neuron-models exists, that differ in how detailed they are modeled after their biological paragon. A discussion of several existing models can be found in [DA01].
Here, it was assumed that if it should be possible to understand how a behavior is accomplished by the mere ordered co-action of processing units. No unnecessary complexity should be added. Therefore the neuron model was desired to be simple, but complex enough to be able to generate the desired complex dynamics.
One that is suitable and used here is the time discrete standard additive neuron model with sigmoidal transfer function σ. In this case the hyperbolic tangent was used, $\sigma = tanh$. The activation a of a neuron is the weighted sum of all incoming neurons plus a bias. The formal description is given in 2.1 and 2.2, where:

- a_i denotes the activation of the ith neuron
- t stands for a discrete time step
- w_{ij} refer to the weight of the connection from neuron j to i
- N is the total number of neurons in a given network

$$a_i(t+1) = \theta_i + \sum_{j=1}^{N} w_{ij}\sigma(a_i(t)), \qquad i = 1, ..., N \tag{2.1}$$

$$o_i = \sigma(a_i(t)) \tag{2.2}$$

For the networks referred to throughout the rest of this work the following holds: Input neurons are treated as buffers for incoming sensor signals, $\sigma(x) = x$. All neurons are connected via directed and weighted synapses. These are either inhibitory (the synapse weight was negative) or excitatory (the synapse weight was positive). Concerning the topology of the used networks it is important to note that no restrictions were imposed on the connections between inner and output neurons. This means that an arbitrary number of

in- and output synapses of a neuron was possible and in particular self-connections were allowed. Neurons that are neither in- nor output neurons are called hidden neurons. When a network contains one or more self-connections or loops it is called a recurrent neural network. The graphical representations of networks used in this work are according the conventions depicted in figure 2.2.

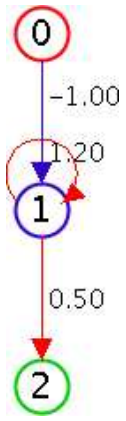

Figure 2.2: A small recurrent network consisting of three neurons. Input neurons (0) are denoted by a red circle, hidden neurons (1) denoted by a blue-, and output neurons (2) by a green circle. Excitatory synapses are represented by red arrows and inhibitory synapses by blue ones. The weight of each synapse is indicated by the number next to it.

2.3 Artificial Evolution

Biological cognitive systems are the most impressive control systems available. They were not built but instead produced by evolution. In this section it is described how the concept of evolution was employed to produce neurocontrollers.

In the 1960s the discipline of Artificial Evolution arose with work by John Holland and L.-J. Fogel in the United States and by Ingo Rechenberg in Germany [PS99]. It is closely related to the field of Artificial Intelligence and Artificial Life. Artificial Evolution (AE)

is a heuristic optimization method based on principles of biological evolution, such as recombination, selection and reproduction. For an impressive example of its usage is referred to the creatures of Karl Sims [Sim94]. Hornby et al. used AE for automatic gait generation for the Aibo [HSYF05], [ea99]. Chernova and Veloso used it for the special purpose of fast walking [CV04], also for the Aibo. In each case AE lead to satisfactory results according to the goal each work aimed at. Also in each case the training took place on the physical robot itself. The learning algorithm was executed on an off board computer in [CV04]. To prevent the hardware from damages and to save time in [CV04] the search space was constrained, which is problematic. Depending on the level of restriction, the possible benefit achieved by AE can be lost. In this work AE is applied to a simulation of the robot, which has several advantages as well as some weakness as will be discussed in section 2.4 and chapter 5.
Another motivation to use AE is the desired employment of RNNs. Till this day there is no general analytical method available for constructing RNNs that can be used to generate a non-trivial behavior. Therefore it is often not feasible to design such controllers "by hand". RNNs exhibit a high dynamical complexity and so far only networks with maximal three arbitrarily connected neurons can be mathematically understood [Pas02]. Or as Inman Harvey [Har93] puts it: "Although behavior-based approaches to robot control appear to be far more promising than traditional model-based functional decomposition methods, [...] the design of such control systems is still prohibitively difficult."

2.3.1 The Algorithm ENS^3

Evolution of neural systems by stochastic synthesis (ENS^3) is the evolutionary algorithm used in this work. It has already been successfully used in several cases e.g. [PD97] and [PSHL01]. It is implemented as part of the ISEE-package (see 2.3.2). It can be used with the proposed neuron model and allows to develop network structure as well as parameter optimization. Parameters are weights of synapses and bias values, the term structure refers to the topology of the network.
The algorithm in combination with its implementation provides a very flexible and thus powerful method for generating RNNs. Only the number of in- and output neurons is predetermined, since it depends on the amount of the sensors and actuators of a given

morphology. Furthermore no self-connections of input neurons are admitted. Apart from this it almost sets no constraints on the development of networks. All kinds of connections are possible, in particular self-connections. Connection types can either be excitatory or inhibitory.
Since the algorithm is biologically inspired the terms used in relation to ENS^3 are also adapted from biology, i.e. the algorithm is said to operate on a population of individuals, $p(t)$, at time t, where an individual $n_i(t)$; $i = 1, \ldots, I$, is a RNN and $I = |p(t)|$.
In addition there are so-called parents, or parent-networks as well as offspring respectively offspring-networks. A population consists of parents and offspring.
The evolutionary process is determined by values that are to be set by the user at start and which can be altered during runtime. These parameters comprise:

- The maximal amount of inner neurons and connections.
- The average amount of offspring.
- A probability indicating how often neurons, resp. synapses are mutated.
- A probability indicating to what extent neurons and/or synapses are mutated.
- It is possible to apply the concept of costs to neurons and synapses. This means that adding neurons to networks is allowed but reduces the fitness. Thus, the generation of small networks is fostered. The values describing these costs have to be set by the user.

Furthermore a fitness function must be provided, which can also be altered during runtime as well as an environment for the simulated robot.
When an evolution is commenced two different scenarios are possible: First, the start from an empty network, i.e. the only neurons present at start are the predetermined input and output neurons and second the start from an initial network. This can be achieved by a previous evolution or by hand design. The base of ENS^3 is a variation-evaluation-selection-loop which can formally be represented as follows:

$$p(t+1) = \boldsymbol{S}(\boldsymbol{E}(\boldsymbol{V}(p(t)))) \tag{2.3}$$

$\boldsymbol{S}$, $\boldsymbol{E}$, and $\boldsymbol{V}$ each represent a set of operators that determine the selection-, evaluation- and variation-process. Their functionality shall be explained by running through the loop. When a new evolution is started and the required parameters are set, the loop is triggered for the first time:

1. A certain amount of stochastically mutated copies of the parent networks is created by V. This is achieved by insertion and deletion of neurons as well as alteration of bias and weights. If and to what extent any of these actions is carried out, is regulated by parameters alterable through the user. For example if the probability for insertion of neurons is set to zero no further neurons will be added to the networks.

2. The operators of E assign a fitness-value to each of the networks of $p(t)$. Therefore a fitness-function must be provided which is then used to evaluate the performance of each individual.

3. According to the fitness-values the selection operators of S determine which networks qualify as parents for the next generation. This is realized by means of a Poisson distribution along with parameters set by the user. In the extremes only the best performing network is allowed to reproduce, or all. The chosen networks are then passed as parents on to the next generation $(p(t+1))$ and the loop is repeated.

There is no stop criterion to this algorithm therefore it has to be halted by the user.
Finally ISEE (see 2.3.2) allows the concurrent generation of several populations, akin biological co-evolution. In this case fitness is given to several networks acting together. Each individual network is part of a different population. That way, not necessarily one eventually big network for a particular problem has to be generated, but a division into smaller sub-modules is possible. This concept, *co-evolution*, was used in this work.

2.3.2 Artificial Evolution with ISEE

The program package *ISEE* (*Integrated Structure Evolution Environment*), developed at the Fraunhofer Institute in Sankt Augustin, provides the required platform for carrying out the previously presented concepts of AE and permits the design of such a simulation as

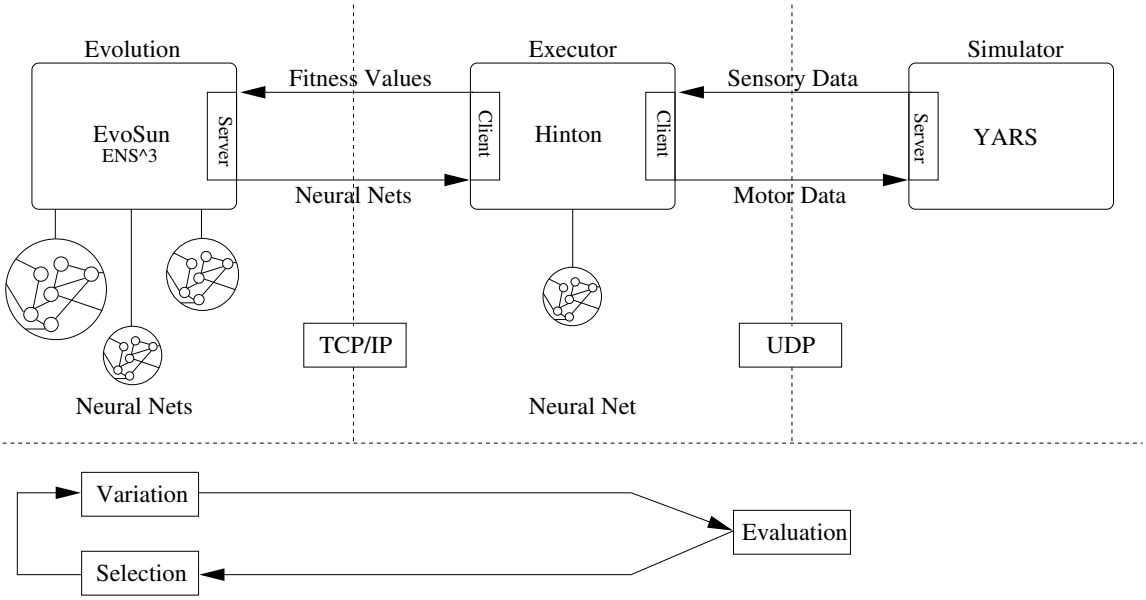

Figure 2.3: Schematic description of AE with ISEE. Adapted from [Mah03]

described in 2.4. Its functionality is schematically depicted in figure 2.3. ISEE consists of three main parts, the programmes *EvoSun*, *Hinton* and *YARS* (*Yet Another Robot Simulator*). YARS is based on ODE (Open Dynamics Engine) [ode05] and permits the realization of physical simulations.

EvoSun contains an implementation of ENS^3. It administrates the population(s) and applies variation and selection to each individual. Every generated network is sequentially sent to Hinton where it is processed and evaluated with the help of YARS. Therefore Hinton sends the processed network output to YARS, which realizes the according action in the simulated robot in the simulated environment. Afterwards YARS calculates new sensory input and sends it back to Hinton which uses it as input for the network. According to the given fitness function Hinton evaluates the current network and sends this value back to EvoSun which employs it for the selection mechanism.

Usually a given robot equipped with sensors receives sensory data in a fixed frequency

which is determined by an internal clock. The same applies for motor data. The Aibo for example receives sensory data and can send motor data every 8ms. This frequency is an important constraint which must be considered in the evolutionary process.

2.4 The Simulator

Carrying out Artificial Evolution in a simulator can have several advantages given that certain criteria are fulfilled. The following section describes what these criteria are. Thereafter the assembly of the used simulator is explained.

2.4.1 Requirements

There are two main reasons that justify the extra effort of constructing a simulation of a given robot when evolving a neurocontroller. First, during AE behaviors might come about that can cause damage to the often expensive hardware. This can be avoided when AE takes place in a simulator instead. Second, since a great number of offspring guarantees genetic diversity in a population, in AE it is sometimes necessary to evaluate many individuals per generation. This can be highly time-consuming when conducted in real-time. A simulator can offer faster calculations and therefore save days or even weeks of evolution time. As discussed in [Jac98] the simulation of a given real robot must fulfill some requirements so that the advantages concerning time are not lost. On the one hand it must not be computationally expensive to ensure that calculations can indeed be carried out faster than real-time. Furthermore the design of a simulated robot must not be too complex and the developed neurocontroller robust. If developing the simulation takes too much time obviously there is not much benefit left in using it. A *robust* neurocontroller is immune to minor alterations of the body. That is behavior-generation should only be dependent on essential morphology-features. For instance, changes in motor-strength, maybe due to low batteries in the robot, should not impede the correct functioning of the neurocontroller. This guarantees that a controller can be transferred to hardware with minimal effort. If this is not the case, the possible benefit of using a simulator is lost. Robustness can be achieved in AE by “breeding” generalists instead of specialists, e.g. by adding

noise when appropriate, or using different environments.

Concluding, the simulation shall be computationally fast, simple to design and support the generation of robust controllers. A program package designed for this purpose (ISEE) has been described in section 2.3.2.

2.4.2 Physical and Simulated Robot

For the reasons stated above, this work was not aimed at constructing an exact simulation of the Aibo-robot. Instead it was supposed to be *physically sufficiently similar*. That is, it was considered to be good enough when an observable behavior of it was also carried out by the real robot. To accomplish this, the basic features of the robot relevant for walking had to be modeled. The robot is equipped with a large number of motors/joints and sensors, including numerous touch sensors, a camera, distance-sensors, foot-contact sensors at each paw, and one sensor per leg-motor. Furthermore 20 motors are built in for its legs, head, jaw, ears and tail. As basic features for modeling the simulated robot were considered: body parts (head, trunk and limbs) defined by its expansion and weight, and the following sensors and motors:

- Foot-contact sensors. They were situated on the sole of each "foot", notifying about ground contact.
- Motor-sensors which continuously monitored the deflection angle of the according motor.
- Three motors per leg, which were each defined by a deflection angle, strength and velocity. The individual deflection angles are depicted in figure 2.7.

The real Aibo (A) and the simulator (B) are shown in figure 2.4.

The mass of each body part as well as its dimensions were acquired from its specification and [Aib05b]. The total mass of the Aibo is declared to be "approx. 1.65kg" [aib05a] including battery and memory stick, the simulated robot's total weight is 1.7kg. Each de-

Figure 2.4: Real robot (A) and simulator (B).

gree of freedom in the simulator was modeled by an ODE-motor. This kind of motor is defined by a maximal velocity in which it can move from one extreme deflection to the other and a maximal force or torque that the motor will use to achieve the desired velocity. Experimental measuring of these parameters on the Aibo produced slightly different values for each motor. Because it was assumed that all motors are of the same strength an average value was used. Precisely the value used for maximal force was 0.43N, and 286°/s for maximal angular velocity.

To facilitate referencing the 12 motors used in simulation an order of legs and motors was chosen. Leg 1 to 4 correspond to right foreleg, right hind leg, left hind leg and left foreleg. Left and right are determined according to the robot's (or simulated robot's) view, see figure 2.5. Joint or motor order for each leg and the according coordinate system are demonstrated in figure 2.6. Motors are enumerated from 1 to 3, where motor 1 is located at the "shoulder" carrying out movement in the xy-plane (according to the given coordinate

body part	height (m)	width (m)	depth (m)	mass (kg)
head	0.04	0.065	0.12	0.08
trunk	0.09	0.09	0.23	1
thigh	0.08	0.03	0.04	0.08
lower Leg	0.04	radius:	0.02	0.075

Table 2.1: Dimension and mass of simulator body parts. Since the lower leg is in shape of a capped cylinder the radius is given instead of width and depth.

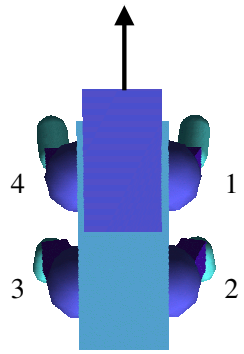

Figure 2.5: Top view on simulator. The arrow denotes the view of simulator and number 1 to 4 specify right foreleg, right hind leg, left hind leg and right foreleg.

system in 2.6). Motor 2 enables movement in the yz-plane and finally motor 3 allows the lower limb to move in the xy-plane.

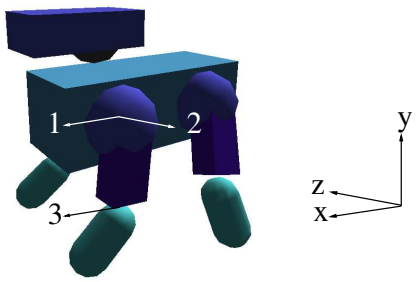

Figure 2.6: Joint order from 1 to 3 in simulator leg. Each joint is denoted by an arrow which indicates its location and the plane in which the specified joint enables movement, according to the global coordinate system shown on the right.

Each of the robot's joints can be deflected within a particular range. All second joints have the same deflection range of -0.1745rad 1.535rad as shown figure 2.7 D. The first and third joint ranges differ, depending if they belong to a fore- or a hind leg. The exact values in radian are visualized in figure 2.7 A, B and C.

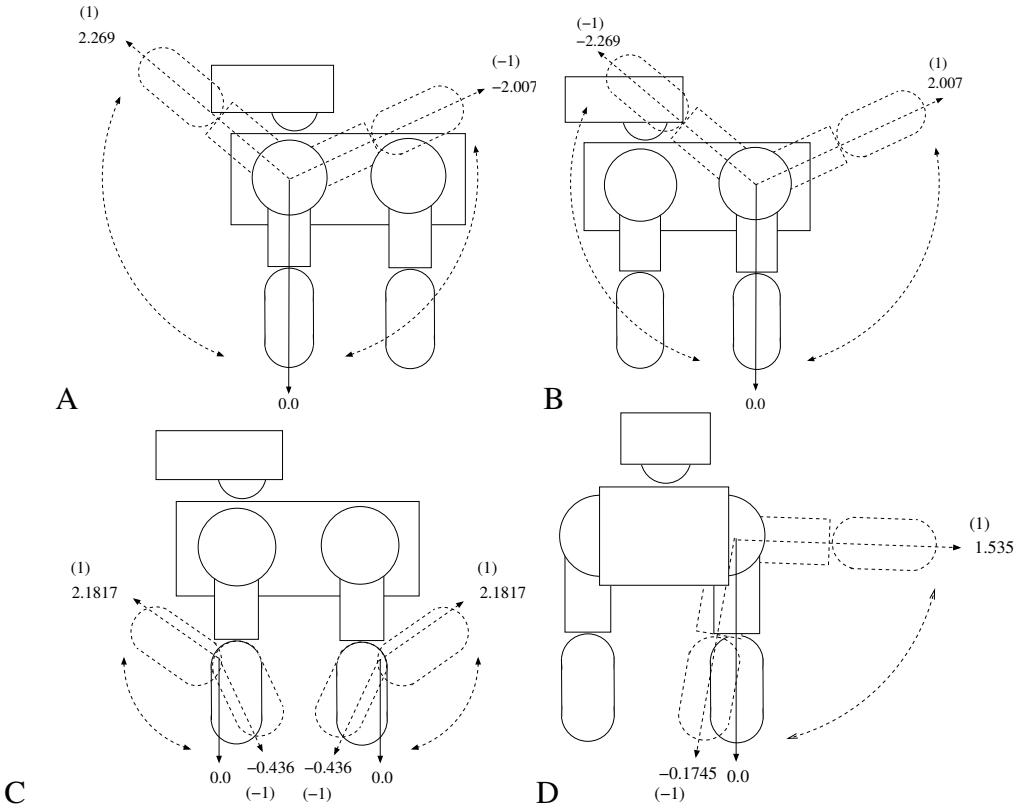

Figure 2.7: Deflection ranges of joints in radian. For all second joints the values depicted in D apply. For first and third joints, ranges for fore- and hind legs differ as shown in A, B and C.

Each motor comes with a sensor which detects the current deflection of it. These sensors are enumerated as the according joints. Thus, sensor two of the first leg specifies the sensor that observes the second motor of the right foreleg. In the simulator these sensors where entailed with a Gaussian noise of 1%. Although the sensors of the physical robot deliver very accurate values this noise was added to enhance the robustness of the controller, as discussed in [Jac98].

2.5 Embedding the Neurocontroller

The neurocontroller as described in section 2.1.3 per definition acts in the sensory-motor loop. In this case the body is the Aibo-robot or the computer simulation of it.
The general setup can be explained as follows:
Sensors (foot-contact sensors and joint sensors) send data to the input neurons of the neurocontroller. Network output can be sent to the motors in the same frequency. The information flow is depicted in figure 2.8. An interface is needed which ensures that sensor

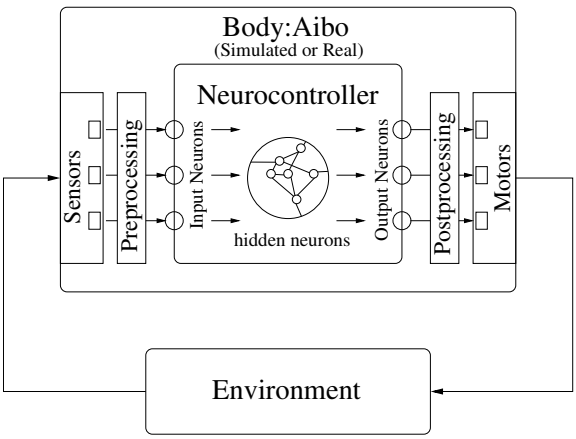

Figure 2.8: Embedding of neurocontroller in Aibo-body. Adapted from [Mah03].

data is sent to the input neurons of the controller and that output signals are transferred to the actuators. The *German Team* Software, described in the following chapter 2.5.1, constitutes an adequate platform.

2.5.1 Neurocontroller and German Team Software

The *German Team* (GT) is the German national robotic soccer team participating in the Sony Four Legged League, and is world champion of 2004. Its software is freely available at [Ger05]. This software is based on a modularized architecture [GT204]. There are modules, among others, for vision, self-localization, body sensor processing, and motion. This presents a convenient way of embedding the neurocontroller as another module in a stable and well-functioning environment and thus to test it on the physical robot. Implemented as motion module the neurocontroller has access to sensor values and its output was automatically passed on to the hardware motor controllers, which finally caused the motors to encounter the postulated position. The sending of motor data and the receiving of sensor data is implemented in the GT software to take place every 8ms or 125 Hz.
To execute a neurocontroller on the hardware it had to be converted to a German Team code module and then added to the GT software. The new module was written on a memory stick which was to be inserted into the Aibo itself. Modules could be chosen by selecting the according entry in the GT software user interface as shown in figure 2.9. Communication between the robot and the computer on which the software runs, is realized via wireless LAN. When selected, the according module is executed on the robot. The user interface of the program "RobotControl" is shown in figure 2.9. Visible is the drop down menu from which several walking modules can be chosen.

Furthermore the GT software provides the possibility to store received sensor- and sent motor-values for a determined time span, which was used for the analysis of the motion process.

2.5.2 Pre- and Postprocessing

Because the proposed neuron-model for the neurocontroller employs the tangent hyperbolic as transfer function (compare section 2.2), output values of the network were between $[-1, 1]$. These output values were used to determine motor positions of the robot's legs. In the GT software the positions are measured in milli radians. Therefore, incoming sensor values for each motor were linearly mapped from milli radians to [-1,1]. Accord-

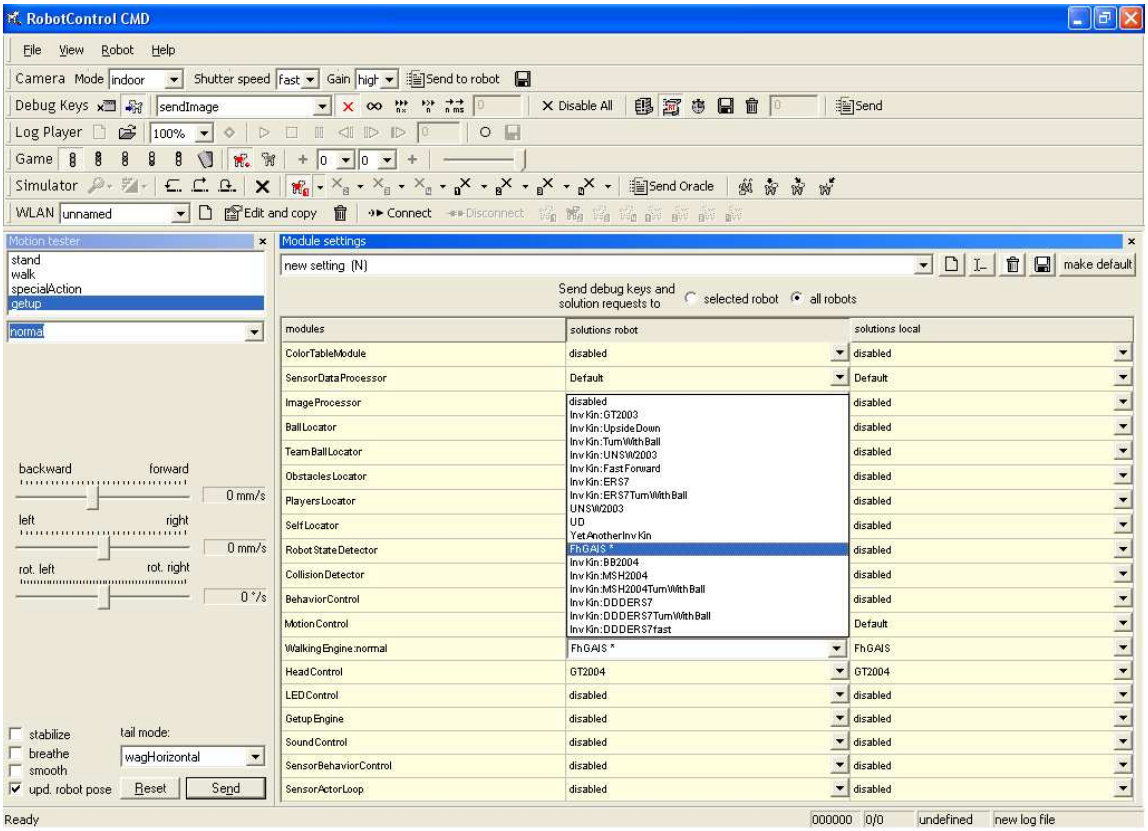

Figure 2.9: User interface of the program “RobotControl” of the German Team. Visible is the drop down menu from which several walking modules can be selected.

ingly the network output had to be mapped back from [-1,1] to the appropriate range in milli radians. Hereby the negative range values in milli radian were mapped to -1 and the positive values to 1 as shown in figure 2.7.

Chapter 3

Evolving the Neurocontroller

In the previous chapter the used tools for generating a neurocontroller for a fast walking behavior for a quadruped have been explained. In the following it is described how their usage lead to the desired controller.

In the previous chapter it was stated that no general analytical method exists to hand-design a neurocontroller for a non-trivial behavior. Instead AE was suggested as a heuristic method to generate such a network. Unfortunately there is no recipe explaining how to employ AE in order to achieve a controller for a certain behavior. Therefore it has to be determined experimentally. Several experiments were conducted and some of them incrementally lead to a controller which generated a satisfactory walking behavior for the robot. A representative selection of these experiments is presented in the following. As starting point for all experiments basic features of walking were considered.

3.1 Features Of Walking

Consulting the relevant literature on walking lead to the following list of properties.

- A *gait* according to [Hil89] is "a regularly repeating manner of moving the feet".
- "A *step* is the advance of one leg, *step cycle* the cyclic motion of one leg and *step length* the distance between two consecutive footholds of one leg in a ground frame."[1][Rid99].
- A step cycle can be divided into a *power stroke* and a *return stroke* , e.g. [Ful93]. During the power stroke the body is supported and propelled forward, whereas during the return stroke the leg is moved from one foothold to the next.
- "A *stride* consists of as many steps as there are legs, i.e. typically each leg completes a cycle of motion and the *stride length* of a gait is the distance the trunk translates during one stride. *Stride duration* is the duration of one stride and the locomotion velocity of periodic locomotion is simply stride length divided by stride duration" [Rid99].
- Important issues, when describing gaits is the sequence and the timing in which the legs advance.
- "Control of insect walking can be considered hierarchical and modular" [Del99].

Based on the above items it was concluded that a gait could functionally be decomposed into

1. individual leg movement (the step cycle) and
2. leg coordination, i.e. the order and the timing in which the legs advance.

This lead to the experimental setup described in the following section.

[1] Ground frame refers to the "inertial frame fixed with respect to the ground"

3.2 General Experimental Setup

The previously described decomposition of a gait was transferred to the control problem. One network for each leg to control the step cycle and one for the leg coordination was considered a plausible partitioning. These networks, or modules, were connected to form the neurocontroller. Furthermore one assumption was made, namely that contra-lateral legs carried out the same step cycle. Thus, all four legs could be controlled by two different networks, which reduced search space and hence sped up the evolutionary process.

The following setup was used for all conducted experiments: Co-evolution was applied to three populations. Thereof was one population for the fore legs, one for the hind legs and one for a coupling or coordination module. These three networks connected together formed the neurocontroller. The connection was realized by *inter-neurons* of which each leg network contained one. The inter-neurons were connected to the coordination module via two synapses which enabled bidirectional data transfer. The modular concept is visualized in figure 3.1. In the following, neurocontrollers are illustrated according to this modular approach. An exemplary presentation is given by figure 3.6. The order in which sensors and motors are addressed is shown in figure 4.3 (A) for the fore legs and (B) for hind legs.

When a walking behavior was observed in simulation the according neurocontroller was transferred to hardware and the speed of the gait was measured. Therefore a test course was set up as shown in figure 3.3. Two labels on a wall indicated the length of one meter. The robot had to cover this distance from a flying start and with fully charged batteries. This was filmed by a digital camera which afterwards enabled to locate the robot's position during the run within time frames of 1/25s and thus to determine its achieved speed. Each neurocontroller was tested 10 times, and its speed was set to be the average value of these runs.

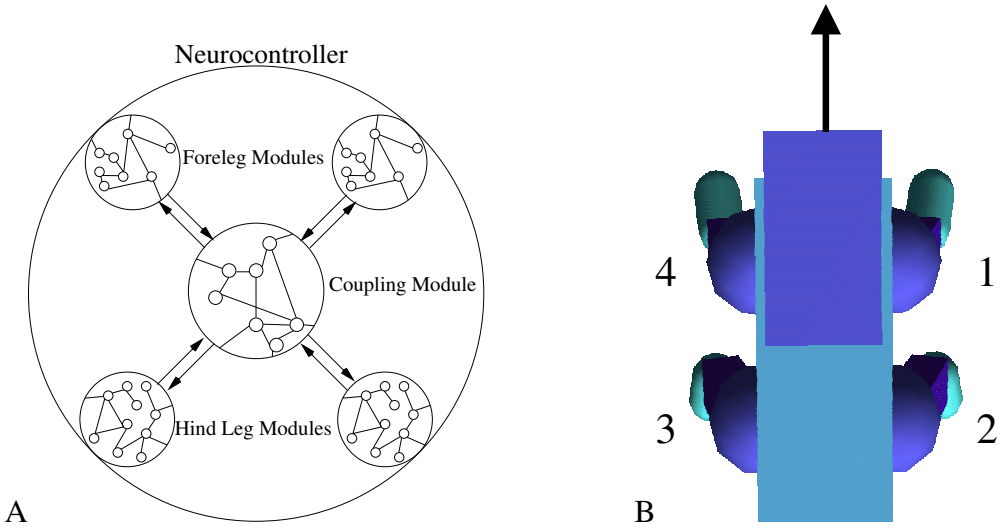

Figure 3.1: Schematic description of the modularized neurocontroller (A) shown next to the robot (B) to which it was applied.

3.3 Experiments

The conducted experiments are distinguishable concerning the following items.

- Whether evolution was started with empty networks, or imposed structures (compare section 2.3.1).
- How the modules were connected to form the neurocontroller.
- The used fitness function.
- The parameter settings.
- The environment.

3.3.1 Empty Leg Networks and CPG

The first evolution was started with empty leg networks, i.e. only in- output- and interneurons. The coordination module contained two coupled oscillators, as shown in figure

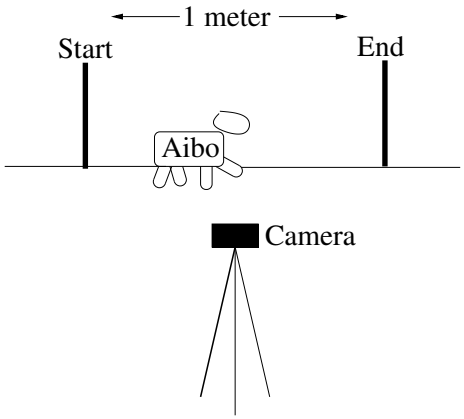

Figure 3.2: Test course setup for velocity measurement of the physical robot.

Figure 3.3: View through camera.

3.4. This network was adopted from another project [KZP04] and is what is generally called a *central pattern generator* (CPG). CPGs are "neural networks that can endogenously (i.e. without rhythmic sensory or central input) produce rhythmic patterned outputs" [Hoo00]. It is generally accepted that rhythmic movement, like walking, swimming or breathing is induced by CPGs [Hoo00]. The signal output of the used CPG were four sinusoidal curves as shown in figures 3.4 and 3.5. The connection between the three modules was realized by the described inter-neurons. One interneuron per leg module received one of the CPG signals. Furthermore signals from the leg networks could be sent back into the CPG via the interneuron. This was put into practice by an inner neuron within each leg network that was connected via two synapses to the coordination module. One lead from the CPG to the leg and the other synapse lead from the leg to the CPG, as depicted in figure 3.6.

The used fitness function (F) is given by:

$$F = \sum_{t=0}^{T} x(t)^2 \tag{3.1}$$

where t denotes discrete time steps and x indicates the position along the x-axis of the simulated robot in world coordinates. The coordinate system was given in figure 2.6. The

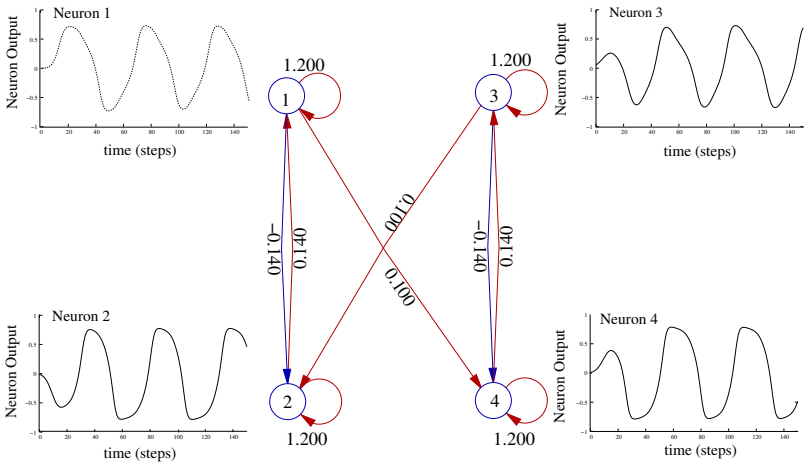

Figure 3.4: Structure and signal output of the used CPG.

sum of the squared covered distances was used instead of the plain covered distance to increase the selection pressure.

During the first generations no changes within the coupling module were admitted, i.e. no neurons or synapses were allowed to be added or deleted. Because no usable results were achieved with this setup these restrictions were loosened. However, this did not lead to improvements within a time span of several days and it was concluded that the chosen approach caused a large search space for AE. To reduce it it was decided to provide initial leg networks. This experiment is described in the following.

3.3.2 Imposed Leg Networks and CPG

Hand-designed networks were used as start networks for the legs. These caused inverted-pendulum-like movements of the legs which were induced by sensory input. The structures of the networks are depicted in figure 3.6. This accelerated the process significantly and after several generations the simulated robot moved forward.

However, it always lapsed into a vibration-like “gait”, i.e. its legs exhibited high frequent oscillations. By this fast back and forth movement of the legs the robot was propelled

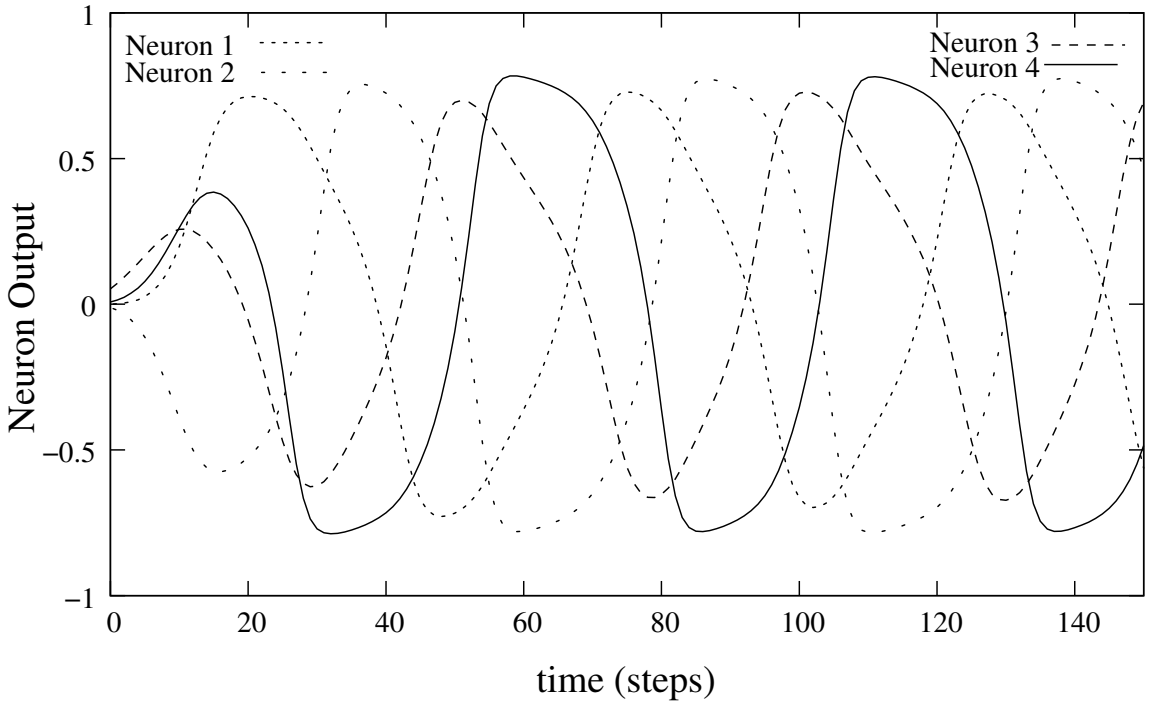

Figure 3.5: The four output signals of the used CPG. Each curve is labeled according to the neuron labels in figure 3.4.

forward, due to the simplified friction model of ODE. Because the individuals that had developed this kind of locomotion gained high fitness scores and reproduced, after a few generations they had ousted all others. Two countermeasures were introduced. First the environment was changed by building in low barriers, as shown in figure 3.7. Individuals with the vibration-like gait could not overcome theses hindrances and thus could not receive a high fitness ranking. Second the fitness function was altered to punish high numbers of change of direction in leg movement as shown in the following formula:

$$F = \sum_{t=0}^{T} x(t)^2 - c(t) * \alpha \tag{3.2}$$

where α is a factor quantifiable during runtime and c denotes the change of direction in leg movement.

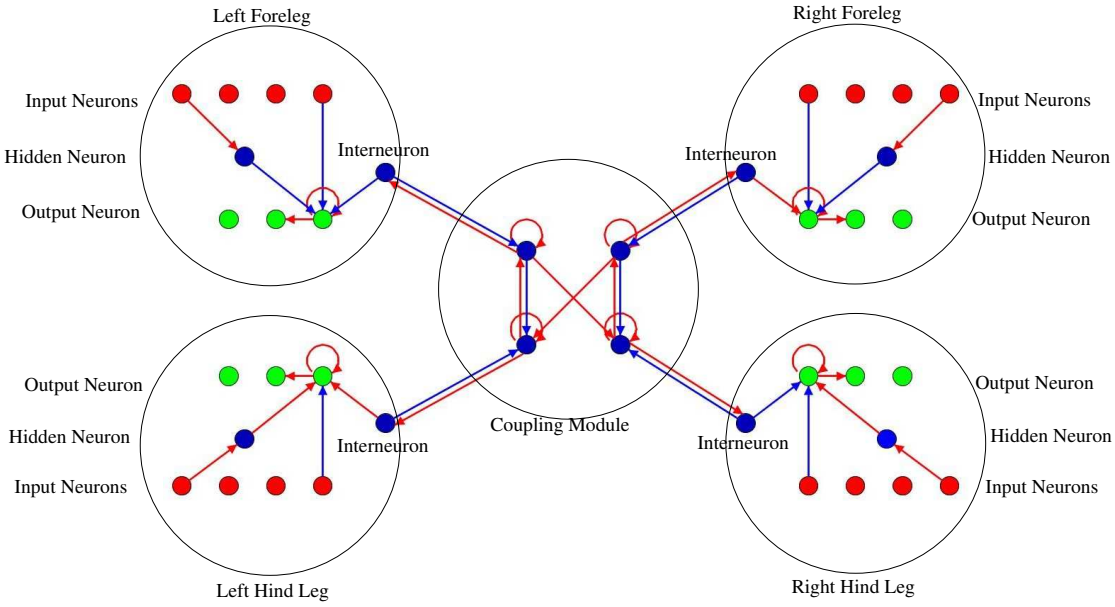

Figure 3.6: Neurocontroller with hand-designed leg networks. Each circle indicates the according module the neurocontroller is composed of. The leg modules contain neurons for sensory input and output neurons. One hidden neuron per leg module serves as connection unit (interneuron) enabling bidirectional data transfer between modules.The order of sensor and motors is according figure 4.3. The fourth input neuron receives foot contact information.

Step by step this lead to neurocontrollers which generated a walking behavior. It was not satisfactory, but could safely be tested on the hardware. Unfortunately the observable behavior of the physical robot differed strongly from that of the simulated one. It was found that the problem occurred due to the insufficient implementation of the simulator's foot contact sensors. In the real machine the sensor reacted to force. A certain threshold had to be overcome to result in a positive sensor signal indicating foot contact. In simulation the contact sensors were modeled by means of very short infrared sensors which was a constraint imposed by YARS. Whenever the foot was close enough to the ground a positive sensor signal was sent to the neurocontroller. In reality the same foot position did not always result in a positive sensor signal, which changed the behavior significantly. Furthermore the leg architecture of the physical robot also had affect on the activity of the

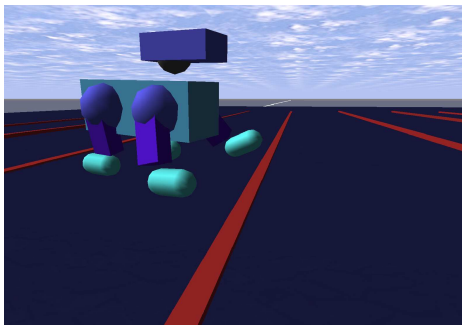

Figure 3.7: Simulated robot within a barrier containing environment.

contact sensors. Its design was more complex than that of the simulated robot as can be seen in figure 3.8. A remodeling of the legs did not lead to better results and it was decided to do without the contact sensors.

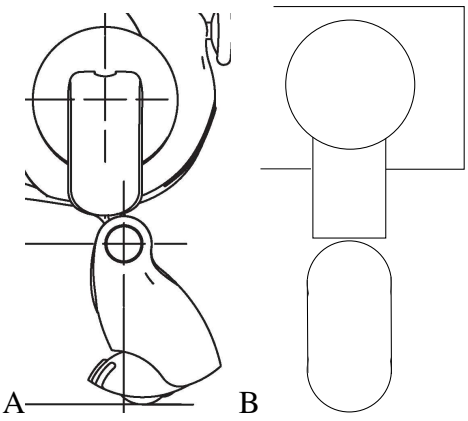

Figure 3.8: Schematic description of a hind leg of the physical robot (A) (adapted from [Mod]) and the simulated robot (B).

3.3.3 Imposed Leg Networks and no Contact Sensors

After the contact sensors had been removed the leg networks were adjusted. The initial network-setup is shown in figure 3.9. Parameters for the CPG-module were set so that no changes within this structure could occur. After 2092 generations a neurocontroller that

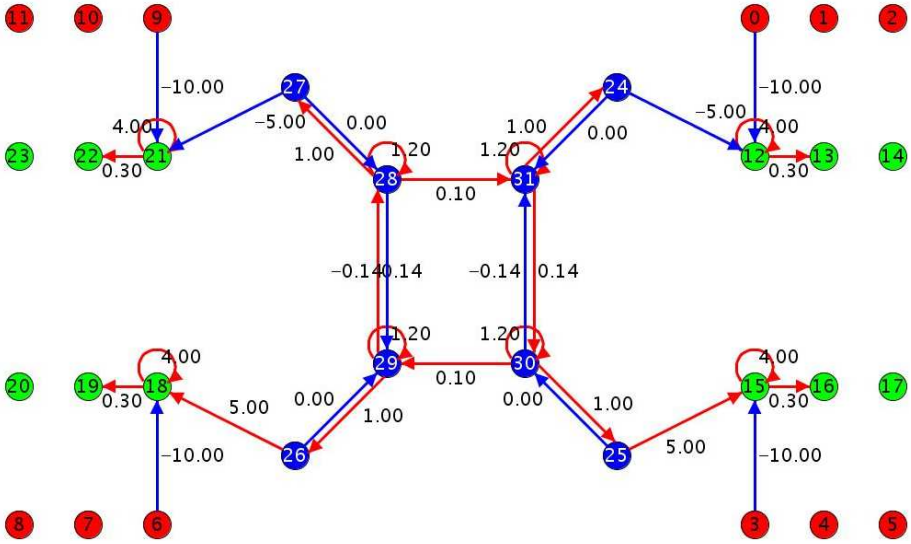

Figure 3.9: Initial neurocontroller with hand-designed leg networks without contact sensors.

lead to an acceptable walking behavior was produced. It is depicted in figure 3.10.

Because small networks were preferred, evolution was continued and costs for neurons and synapses were introduced, as explained in section 2.3.1. That way, after 4320 generations, a considerably smaller neurocontroller was produced, as can be seen in figure 3.11, which led to a faster walking behavior. The measured speed achieved by the physical robot was 39.56 cm/s. This was the best performing neurocontroller generated by AE of all conducted experiments. It is analyzed in chapter 4. In the following it is referred to as "neurocontroller 4320". Successively evolved controllers, although gaining a higher fitness in evolution, did not result in a faster behavior on the real machine. This is discussed in chapter 5.

3.3.4 Connection Module and Fixed Legs

In the previously conducted experiments a given CPG was used as coupling module. To find out if, and if so, which other coupling structures could lead to an equivalent behavior

no initial coordination module was provided this time. Leg modules were taken from another well-performing controller which is shown in figure 3.10. An evolution was started in which leg modules were not allowed to change, but the connection network. A neurocontroller that generated a walking behavior evolved. It is shown in figure 3.12. The output of this module was plotted and the resulting curves were similar in shape, amplitude and phase to the CPG output shown in figure 3.5. When synapses leading to this module were removed it still produced similar output. Thus, what had evolved was also a CPG, but a more complex one. When the leg networks were admitted to be changed, the walking behavior slowly degenerated.

Other conducted experiments focused on imposing a structure on the coupling module, that was determined by sensory input instead of a CPG. In [vT04] it is shown that a walking behavior for a simulated hexapod can be realized this way. This approach did not lead to any functioning neurocontrollers.

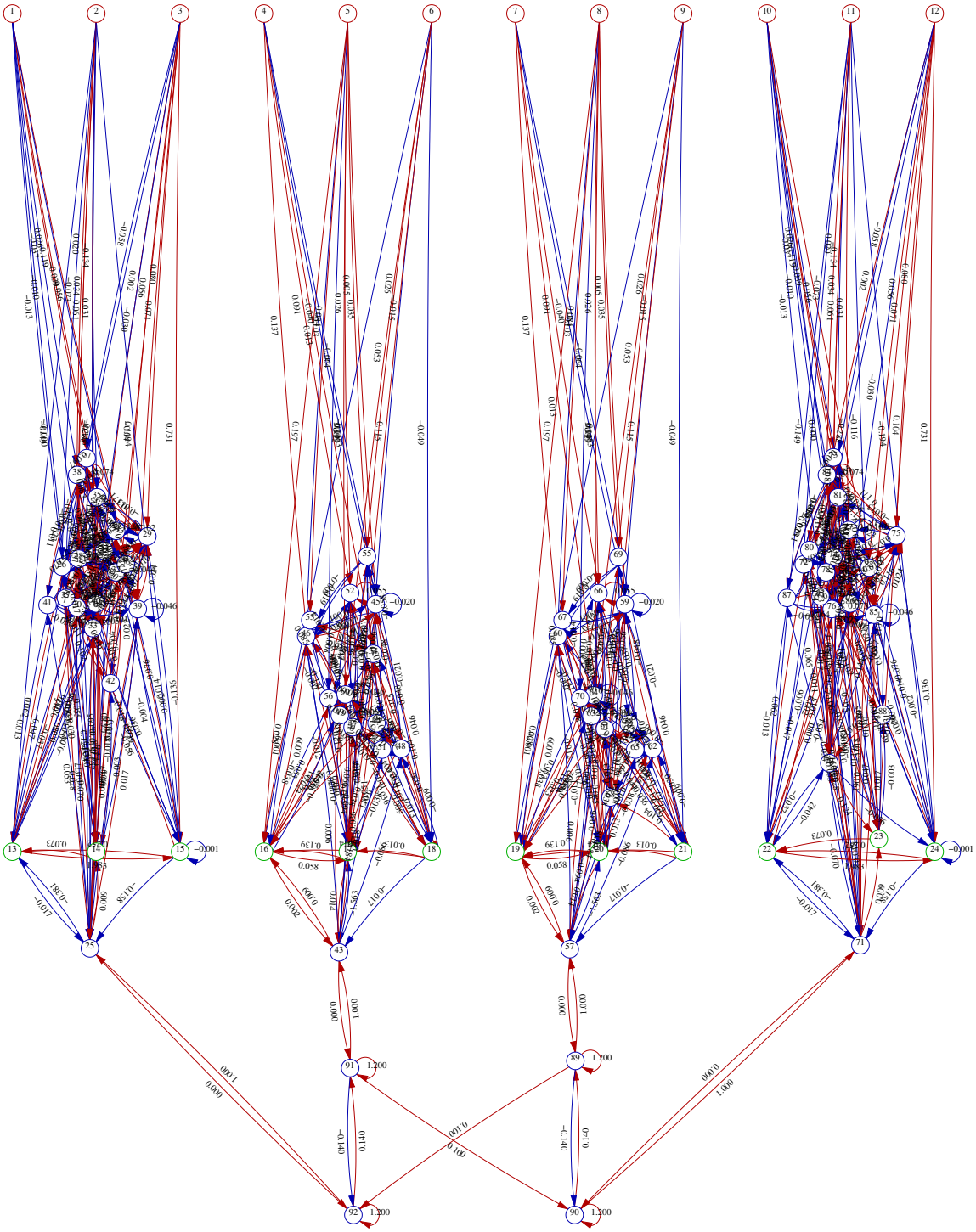

Figure 3.10: First evolved neurocontroller that generated an acceptable walking behavior. Due to the large amount of neurons and synapses the illustration differs from the other presented neurocontrollers. Each partition denotes a leg network in the following order: right foreleg, right hind leg, left hind leg and left foreleg. The four neurons underneath constitute the CPG. Observable is the complex structure of the network.

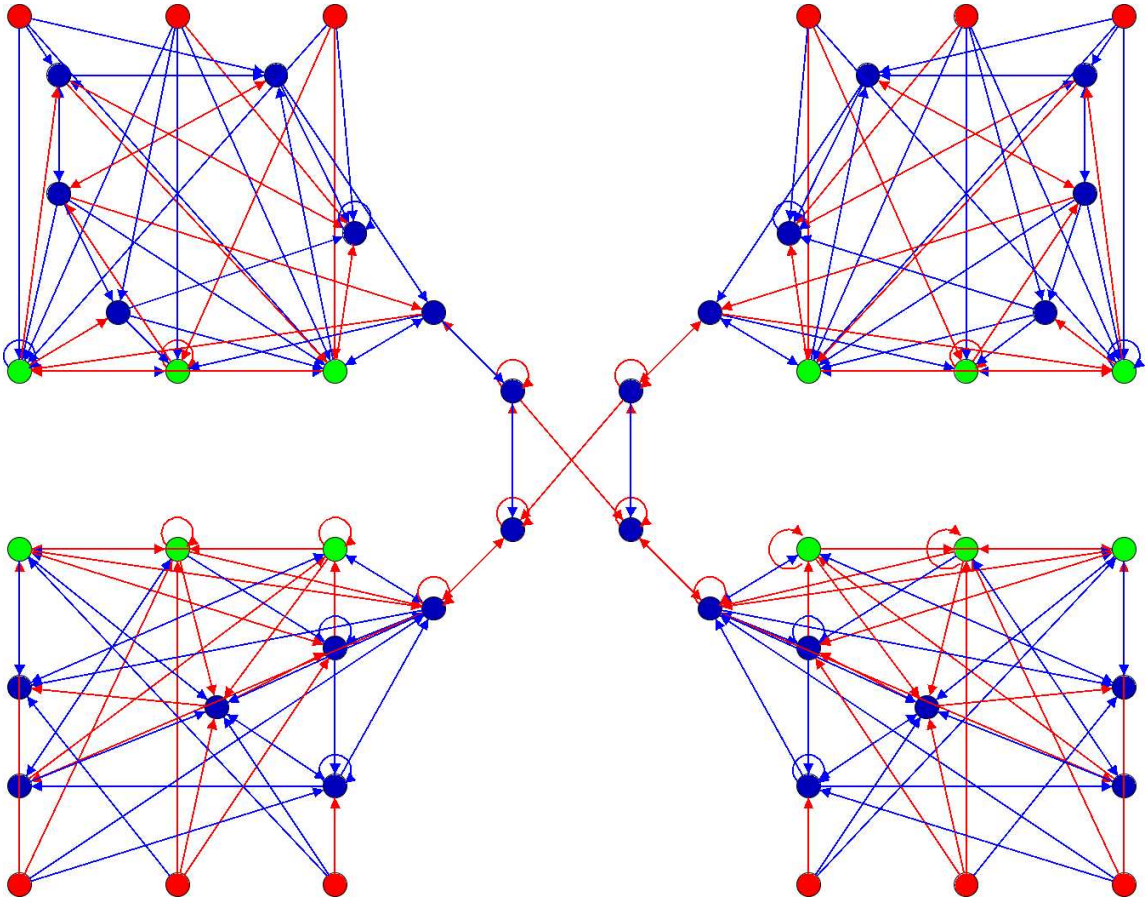

Figure 3.11: Structure of the best performing neurocontroller generated by AE after 4320 generations.

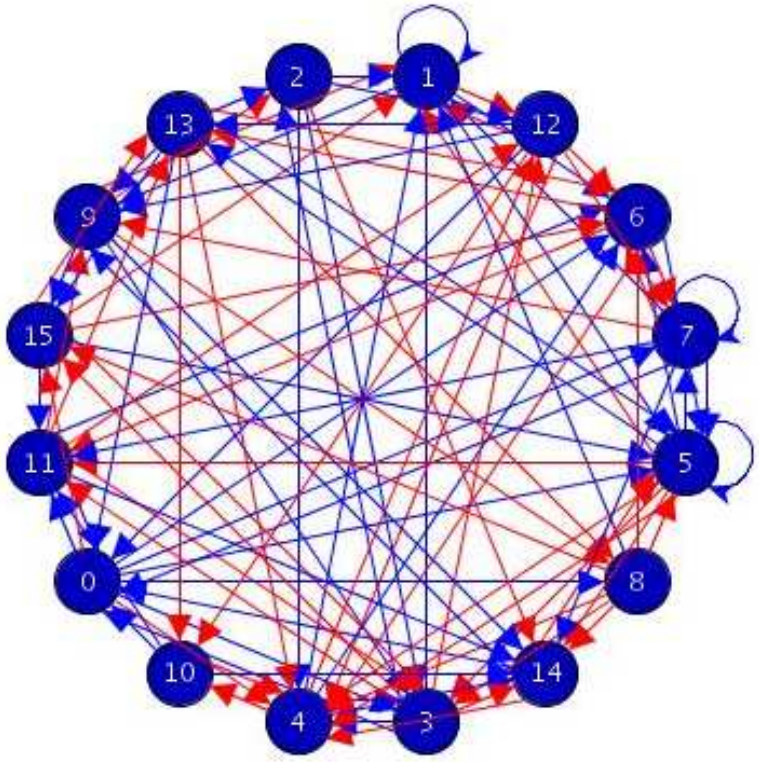

Figure 3.12: The only evolved coupling module that lead to a walking behavior.

Chapter 4

Analysis

A neurocontroller that lead to a fast walking behavior in simulation and on the physical robot was developed by Artificial Evolution. To find out how this behavior is generated the controller is simplified at first and then analyzed. The results are used to improve the walking behavior. The procedure of reducing, analyzing and subsequent improvement of the controller is content of this chapter.

In the introduction it was stated that the motivation of this work is twofold. On the one hand a neurocontroller for fast quadrupedal gait generation was to achieve, and on the other to understand *how* this behavior is accomplished.
So far an adequate neurocontroller has been developed using Artificial Evolution (neurocontroller 4320). In this chapter its functionality is examined, questions addressed are for example if and how sensory input is processed. It is also described how the neurocontroller was simplified to facilitate its analysis.
Finally it is shown how results of the analysis could be used to improve the walking behavior concerning speed.

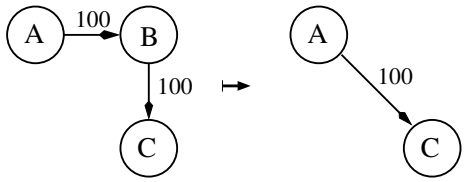

Figure 4.1: The two depicted paths are set equal. The new path does not lead to the exact same network-output, since neuron C requires one time step less to calculate its output. But because one time step is of very short duration (0.008s) compared to the duration of one step cycle (0.45s, see section 4.3.3) this was considered to be negligible.

4.1 The Neurocontroller

The controller performing best, in simulation and on hardware is shown in figure 3.11. As one can see the network is too complex for an intuitive understanding. For this reason an effort was made to reduce it to its essential components.

4.1.1 Simplifying the Neurocontroller

The reduction of the controller was conducted the following way:

- A neuron without outgoing synapses can be deleted (and all the synapses leading to this neuron), since it has no impact on other neurons. If thereby other neurons arise without outgoing synapses they are deleted recursively until no more "dead-end-neurons" are left.

- The observed neuron output of each hidden neuron is examined. If it is zero or very close to zero ($|o| < 0.01$) it can be assumed that it does not have much effect on the connected neurons and is deleted.

- Synapse-paths that can be substituted by shorter ones are identified and replaced as shown in figure 4.1.

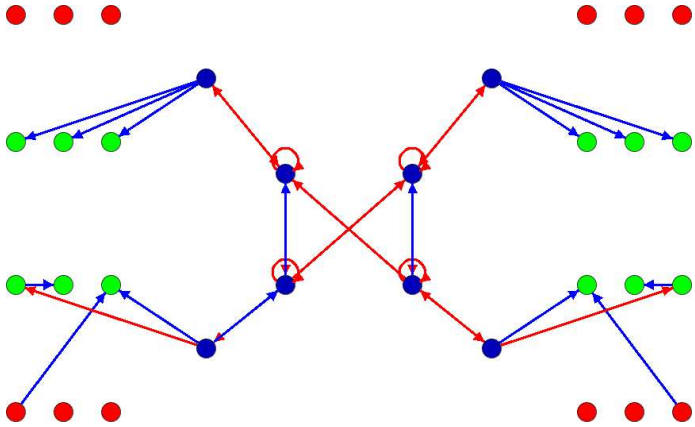

Figure 4.2: Structure of the simplified controller.

By applying the above items to neurocontroller 4320, it could be reduced to a considerably smaller network as shown in figure 4.2. The individual network modules are separately shown in figures 4.3 and 4.4 (B) as well as tables for the according bias values 4.1, 4.2 and 4.5.

4.1.2 Verification of the Simplified Neurocontroller

To ensure that the simplified controller still produced the same, or nearly the same output as the original, output signals of both controllers were plotted and compared to each other, see figure 4.6. The differences were found to be insignificant.

Since there is very little difference between the outputs of the original and the reduced controller it was concluded that they were of comparable quality.

Furthermore speed of the generated gaits by both controllers was measured in simulation. Velocity was measured five times for each controller and the averaged values were compared. It was found that the velocity achieved with the simplified controller was 1.66% slower than with the original. Both controllers were also tested on the hardware and the speed attained with the simplified controller was 2.17% slower. This loss in speed was considered as negligible.

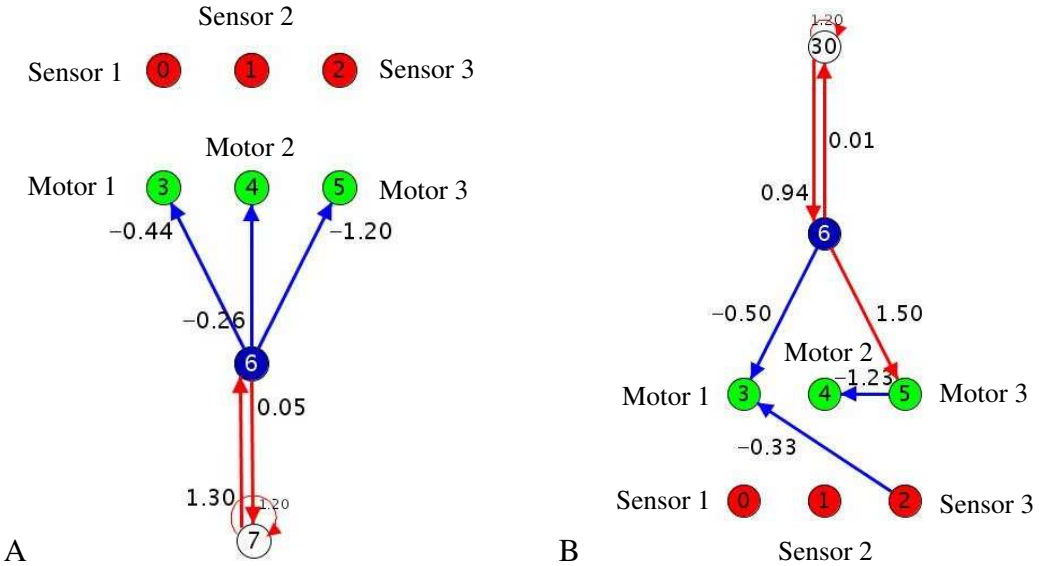

Figure 4.3: The foreleg network (A) and the hind leg network (B) of the simplified controller. Neurons without color filling are not part of the network, but indicate the connection to the coupling module.

neuron	bias
0	0
1	0
2	0
3	0.1111780742956206
4	-0.39
5	0.5
6	0

Table 4.1: The bias values of the neurons of the foreleg of the simplified neurocontroller, shown in figures 4.2 and 4.3 (A).

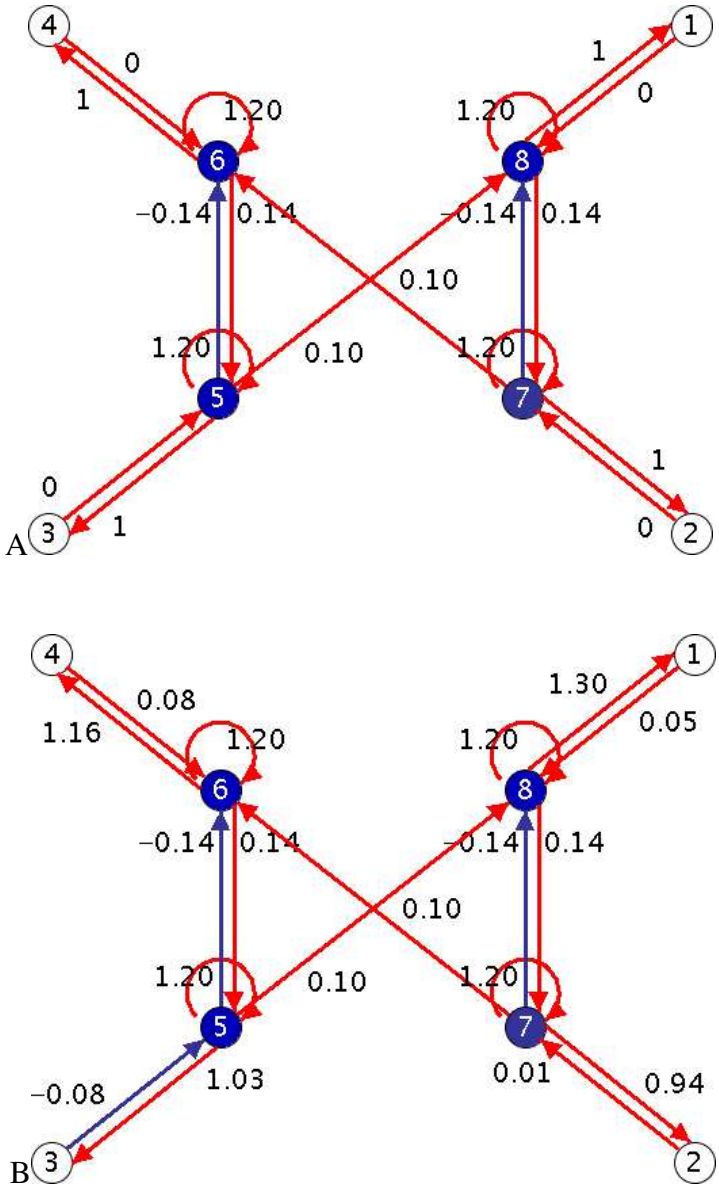

Figure 4.4: The CPG with its connection to the inter-neurons of each leg (not color filled neurons). The originally used CPG (A) and the CPG with evolutionary altered parameters (B). The according parameters are listed in table 4.5.

neuron	bias
0	0
1	0
2	0
3	0.07922423324963722
4	-0.8655142036648698
5	0.9817831631826901
6	0

Table 4.2: The bias values of the neurons of the hind leg of the simplified neurocontroller, shown in figures 4.2 and 4.3 (B).

Now that it was ensured that the reduced neurocontroller indeed performed as well as the larger one it could be examined to find out how the fast walking behavior was accomplished.

4.2 Analysis of the Simplified Neurocontroller

Rather than analyzing the neurocontroller mathematically it was examined experimentally. The goal was to describe how the network output that lead to the observable behavior was produced by the controller. The analysis was conducted according to the modular decomposition in fore legs and hind legs. Furthermore the connection between the modules was examined. The coordination module was not studied because it was not altered and a study on it is available (see [PHZ03]). Of particular interest was if the reconnection from the leg module to the CPG had any impact – and if so which. This was closely coupled to another question, namely what impact did sensory signals have on the behavior and in particular on the CPG. For medium-speed walking of animals a neural system consisting of CPGs and reflexes is proposed [FKC03] but it is not known how exactly both, CPGs and reflexes, interact, and it is a topic of current research, e.g. in [BM98]. Here, a reflex is considered to be a sensory signal that had direct impact on network output.

neuron number	bias original cpg	bias altered cpg
1	0	-0.0029
2	0	0.0016
3	0	-0.002
4	0	0.0065
5	0	0.0175
6	0.01	0.0053
7	0	0.0037
8	0	-0.0044

Figure 4.5: A comparison of bias values of the original and evolutionary altered CPG neurons. The enumeration of neurons is according figure 4.4.

4.2.1 Foreleg

The network for the fore legs is simple. As can be seen in figure 4.2 it does not contain any hidden neurons with exception of the required interneuron. No sensory input is incorporated. Furthermore there are no re-currencies with exception of the synapses between the interneuron and the connection module. Each output neuron receives the same input signal differing only in amplitude. The output of all three output neurons for the right foreleg is plotted in figure 4.6 (B) and for the left foreleg in (D).

4.2.2 Hind Legs and Sensory Input

The hind leg network, visible in 4.2 does not contain any hidden neurons, with exception of the interneuron. All three output neurons receive the same input signal differing in amplitude. However, in contrast to the foreleg network, the signal received by the second and third output neuron are inverted, as can be seen in 4.6 (F) and (H). The hind leg network makes use of one sensory input. The output neuron for the first motor receives input from the third sensor. What impact this has on the walking behavior is tackled in the following. In figure 4.3 it can be seen that of both leg modules only the one for the hind legs makes use of sensory input. However this signal is not propagated further, i.e. there are

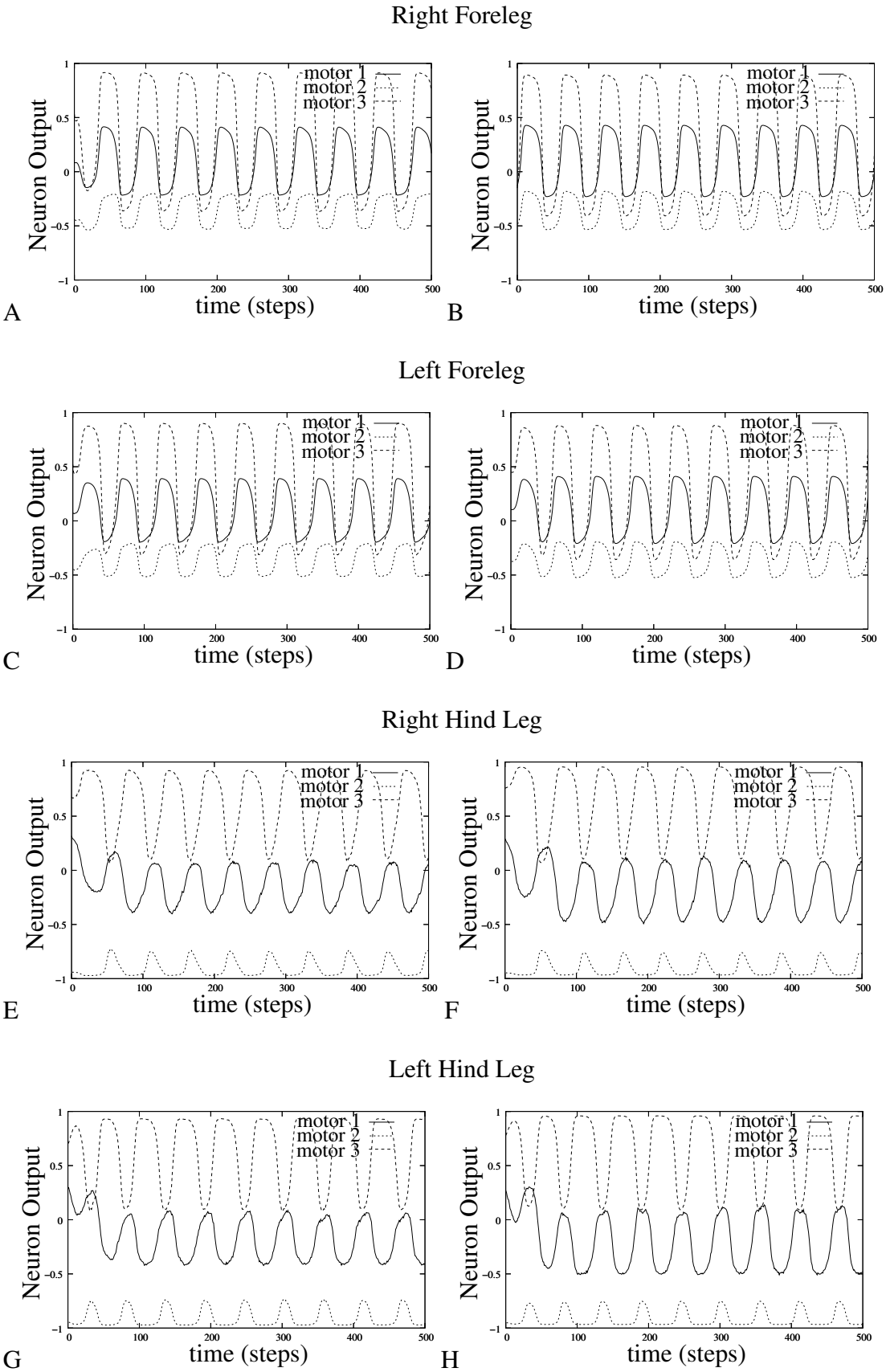

Figure 4.6: Comparison of output of neurocontroller 4320 (left column) and its simplified version (right column) for each output neuron for all legs. As can be seen, the differences between the output of both controllers are small.

no outgoing synapses from output neuron three. To find out what impact this input signal has, the output of neuron three was plotted twice. One time in its original form (o), and a second time where the incoming synapse was deleted ($\hat{o}$). Both plots are shown in figure 4.7. As can be seen o and $\hat{o}$ differ in offset to x-axis and amplitude. Also slightly in wave-

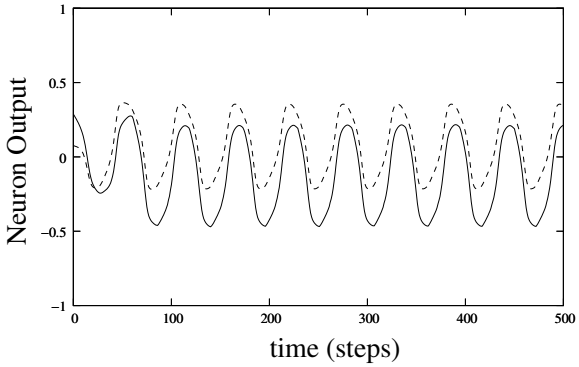

Figure 4.7: Plot of output of output neuron three (according to labeling in figure 4.3 with, o, (solid line) and without, $\hat{o}$, (dotted line) sensor input. Both plots differ significantly in offset and amplitude.

form and phase. By manipulating bias of neuron three and $w_{3,2}$, amplitude and offset of $\hat{o}$, were adjusted to o. The resulting curve is plotted and compared to o in figure 4.8. The resulting neurocontroller was tested on the hardware. The robot moved, but slower than with the original controller. This was probably due to the differences in waveform. When tested in the simulation no walking behavior was accomplished. The simulated Aibo fell over immediately. It was found that hind legs were deflected so close to the Aibo's torso, that balance could not be maintained. Responsible for this hind leg movement was the difference between the transients, which can be seen in figure 4.8. The initial values of o are above zero while values of the manipulated $\hat{o}$ are below zero. An output signal below zero causes the hind legs to move towards the robot's torso, as depicted in figure 2.7. In simulation the robot starts moving from a standing position. Thus the transients of the output signal determine the moves that lead from standing to moving. The incorporated sensory input is therefore important to the simulated robot's initial behavior which leads to walking. The physical robot however was not started from a standing position. Rather

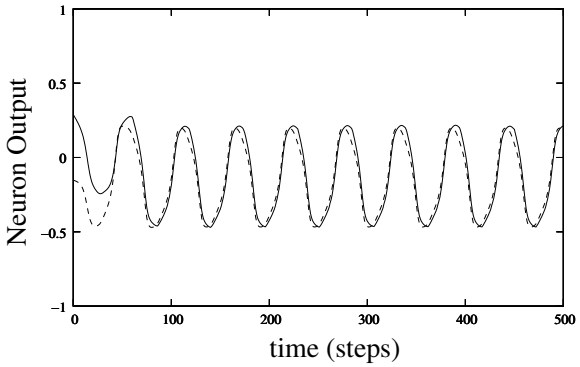

Figure 4.8: Plot of o (solid line) and manipulated $\hat{o}$ (dotted line). Bias, and weight of incoming synapse of the latter were altered to match the original signal. Differences between both plots are small.

it was put on the ground when it already carried out walking movements. Because this was also possible without sensory input it can be concluded that it was not crucial for the walking behavior.

Furthermore it was possible to substitute the synapse leading from neuron five to neuron four of the hind leg (comp. figure 4.3 (B)) neuron three by another, leading from the interneuron to neuron four, as depicted in figure 4.9. After that the walking behavior was still carried out by the robot (but slower). This shows that fore- and hind leg structurally only differ in the sign of the weight of the synapse which leads to the third output neuron and thus determines lower limb movement. Although the sensory input did not alter CPG dynamics, it modulated the output of the CPG and thus contributed to the accomplishment of the walking behavior. Considering the above it can be concluded that the sensory input was not essential for accomplishing a walking behavior but for getting into the position for it.

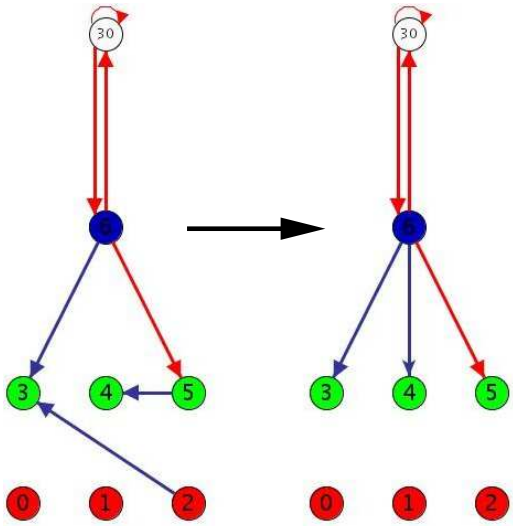

Figure 4.9: The network structure for the hind leg, as shown in figure 4.3 (B), can be transferred to the depicted structure without leading to a loss of walking ability. Thus, fore- and hind legs can be controlled by networks of the same structure.

4.2.3 Impact on CPG

Obviously the "heart" of the walking behavior is the used CPG. During AE it had not been subject to change, due to the fact that in other experiments this had lead to a degeneration of the walking behavior. For the presented network only the synapse weights between CPG and inter-neurons were modified, compare figure 4.4 (A) and (B) and table 4.5 were the according bias values are listed.

As can be seen the modifications of weight values were small (< 0.3). To find out if these had any impact on the patterns generated by the CPG the output of each interneuron of the initial network was plotted and compared to the plotted output of the inter-neurons of the altered network. These plots are shown in figure 4.10.

It can be observed that the small variations in synapse weight and bias caused similar changes in the output signal of all four inter-neurons. These changes are:

- an increased amplitude,

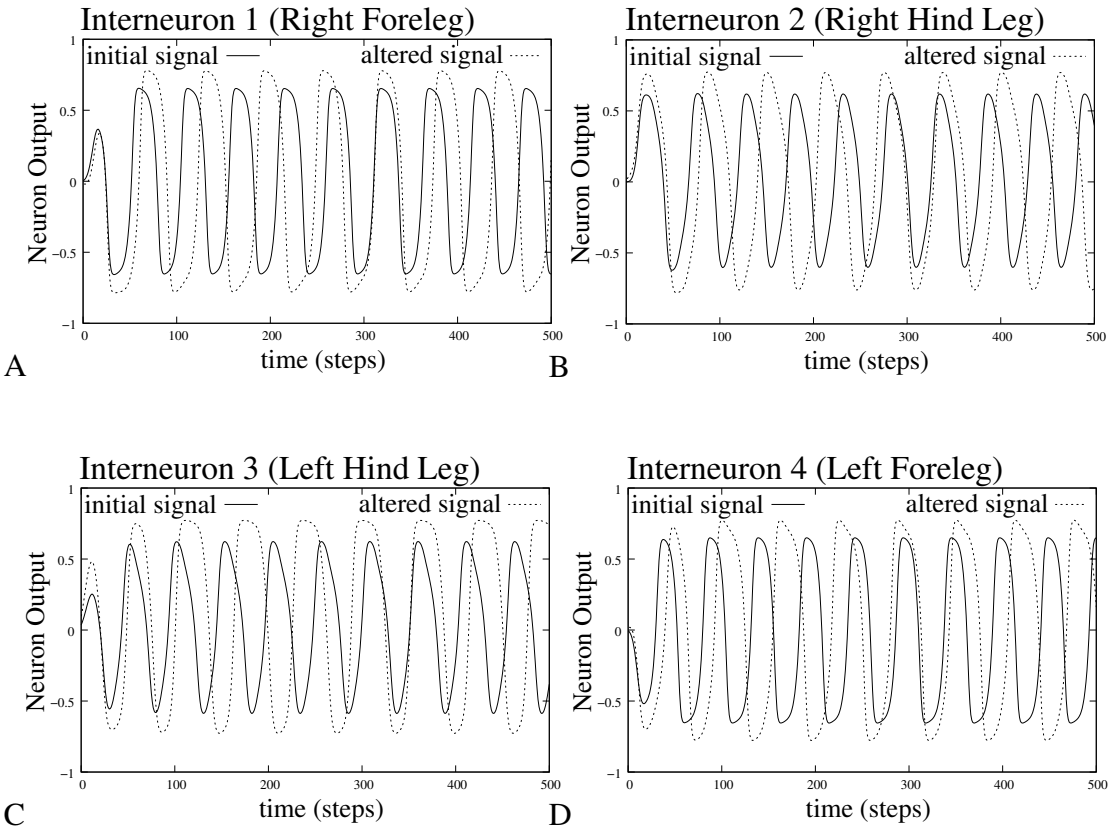

Figure 4.10: Comparison of interneuron-output of initial network (solid line) and output of the equivalent interneuron of the altered network (dotted line). The altered signals differ in frequency, waveform and amplitude from the original signals.

- a lowered frequency and
- a rounder waveform.

The frequency of the altered signal is lower than that of the original one, this leads to the assumption that a lower frequency leads to a faster movement. This assumption will be tested in the next section.

4.3 Improving the Neurocontroller

After the functionality of the simplified neurocontroller had been examined this knowledge was used to adjust some of its parameters by hand. Doing this is justified, since the simulated Aibo and the real one differ from each other in many aspects (comp.section 2.4.2). Thus the exact same behavior cannot not be expected. Since no more learning takes place, when the network is applied to the real robot, the only way to improve performance is changing parameters manually.

4.3.1 Path Trajectory

One observed discrepancy between the walking behavior of the simulated and the real robot was the robot's path trajectory. In simulation this was straight forward, the real robot however tended to turn to the right and – if given enough time – completed a full circle. Because this had to be due to different input signals to the leg networks, output of each interneuron was examined. In figure 4.12 (A) the output of all four inter-neurons of the initial neurocontroller (comp. figure 3.9) is plotted. Next to it, in figure 4.12 (B) the output of all inter-neurons of the simplified neurocontroller. It can be seen that for the latter an increase in amplitude of the signal for the right foreleg occurred. Furthermore the phase shifts between all signals in (B) differ from those in (A). The greater amplitude of the signal of interneuron 1 was considered to lead to greater strokes of the right foreleg in comparison to the left one. The question arose why this did not lead to a turn to the left in the robot. An answer was found when examining the used hardware. Each leg of

the robot could be moved with little pressure to the outer limits of the deflection angle the according motor could be set to. For instance, if the first joint of a leg was set to move between -30°and 30°, then – if pressure was applied – the origin, around which the motor moved, was shifted into the same direction. See figure 4.11 for an illustration. The higher

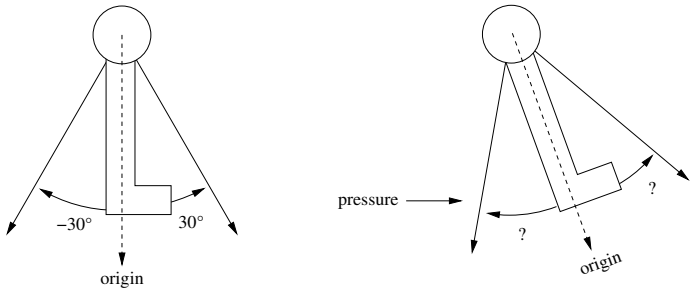

Figure 4.11: A leg and the according motor, which is set to move between -30°and 30°. If a force is applied on one side, the origin around which the motor moves is shifted.

amplitude of the signal caused the first motor of the right foreleg to move within wider angles than the other legs, and bodyweight lead to a shift of the said origin. This resulted in in the observed right turn of the path trajectory of the robot. To solve this problem synapse weights were manually changed until the amplitudes of all incoming interneuron signals were equal as is shown in figure 4.12 (C). The robot's path trajectory became straighter after that, but still tended to turn to the right.

It was observed on the physical robot that its hind legs were not equally deflected, i.e. the right hind leg was constantly kept closer to the robot's torso than the left one. Although both received relatively equal input. It was assumed that this was due to the said phase shift between signals. In 4.12 (A) it can be seen that the signals for both hind legs have a difference in phase of approximately 180°. The signals for the hind legs of the altered CPG this "inversion" was displaced. It was assumed that due to this displacement the right hind leg was not given enough time to move to its correct position. By changing weights of the according synapses leading from the inter-neurons to the CPG, it was possible to adjust the phases of the two hind leg signals. A plot of the result is shown in figure 4.12 (D). After that the path trajectory of the robot was much straighter.

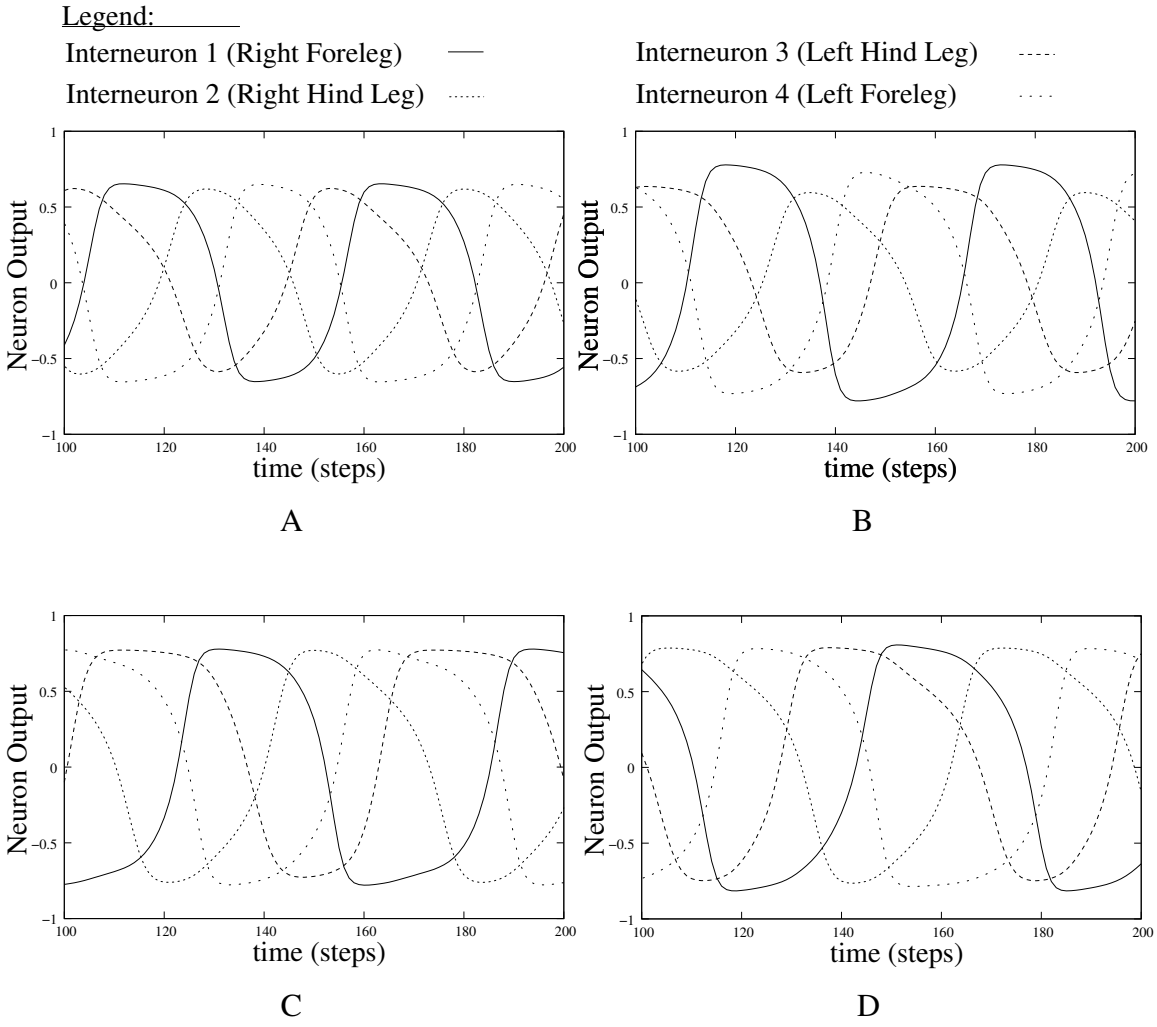

Figure 4.12: Output of inter-neurons of originally used CPG (A) and CPG altered by evolution (B). Note that amplitude of signal for the right foreleg in (B) increased. Also different phase shifts between signals than before, i.e. in (A), are visible. Output of inter-neurons after amplitudes were equalized (C), and after amplitudes and phase shift between signals were adjusted (D).

4.3.2 Stride Length

Because stride length and stride frequency affect the locomotion velocity in animals [Ful93] it was tested in this work if these two items could contribute to the robot‘s walking speed. At first the affect of stride length was examined. It is determined by the signal's amplitude of the output neurons. Because both leg networks were feed forward, i.e. not containing re-currencies, the amplitude of these signals could easily be adjusted. Only the weight of the synapse leading from the interneuron to the leg network had to be changed. The neutral position of each leg is determined by the bias value of the output neuron. A more upright position of a leg "elongates" it and can lead to a larger stride length, therefore several leg positions were also tested. In table 4.3 is listed how the networks were changed, what the according speed (an average value of 5 trials) was and what was concluded by that. All changes were applied to the simplified neurocontroller. The order of all given values (e.g. bias) in the table is according to the introduced motor order in figure 2.6). Figure 4.13 visualizes the reached speed of each network.

Network	Description	Speed (cm/s)	Conclusion
1	evolved neurocontroller	39.56	
2	simplified neurocontroller	38.7	
3	weights of synapses from inter-neurons to legs increased by adding 0.1 to the original values	38.28	
4	weights of synapses from inter-neurons to legs increased by adding 0.2 to the original values	38.58	
5	weights of synapses from inter-neurons to legs increased by adding 0.6 to the original values	38.16	
6	weights of synapses from inter-neurons to legs increased by adding 0.2 to the original value and bias of output neurons of fore legs changed. From the original values of 0.11, -0.39, 0.5 to 0, -0.7, -0.2	40.04	Change of bias of fore legs to a more upright position is contributes to walking speed.

continues...

...continued

Network	Description	Speed (cm/s)	Conclusion
7	bias of output neurons for fore legs changed from 0, -0.7, -0.2 to 0, -0.39, 0	41.15	The more the fore legs point into the opposite direction of the walking direction, the faster is the gait. Maybe because they can exert more force to the ground.
8	bias of output neurons for fore legs changed from 0, -0.39, 0 to -0.08, -0.39, -0.1	42.09	
9	weights of synapses from inter-neurons to hind legs set to 1.4	39.81	An increasing stride length for hind legs contributes to overall speed.
10	phase shift between interneuron signals for hind legs adjusted	42.38	More time between leg movement might be required.
11	phase shift between interneuron signals for hind legs adjusted and fixation of 2nd and 3rd motor of fore legs	40.13	
12	as 11 but with higher amplitude of signal to first output neuron of fore legs (from -0.44 to -0.5)	39.53	2nd and/or 3rd motor of fore legs should not be stiff.

continues...

...continued

Network	Description	Speed (cm/s)	Conclusion
13	same as 11 with amplitude of signal to first output neuron of fore legs set to -0.8	43.41	Greater stride length of fore legs increases velocity.
14	same as 11 with amplitude of signal to first output neuron of fore legs set to -1.2	Not able to walk.	Stride length is limited by motor strength.
15	All insights won by the above conclusions applied to one network	43.40	

Table 4.3: Adjusting parameters to change stride length. The order of all given values (e.g. bias) is according to the introduced motor order in figure 2.4.2.

By increasing stride length as described above, an increase in velocity of 3.84 cm/s was achieved.

4.3.3 Frequency

Stride frequency is determined by the back- and forward motion of a leg. In case of the Aibo-robot this motion was basically carried out by the first motors, which all received their signals from the first output neurons. The frequency of these signals therefore indicate stride frequency. Fast Fourier Transformation (FFT) was used to calculate the frequencies of these signals. FFT was applied to a convolution of a 5-second-signal with a Hanning-window. The primary signal of the first output neuron of the right foreleg oscillates 11 times within 5 seconds, thus the frequency is 2.2 Hz.The according Fourier spectrum is shown in figure 4.14. Amplitude of the signal was determined by calculating

Neuron Output for first Motor of:	Frequency	Amplitude
Right Foreleg	2.2	0.3127278
Right Hind Leg	2.2	0.2172463
Left Hind Leg	2.2	0.2419408
Left Foreleg	2.2	0.2931104

Table 4.4: Amplitude and frequency of the incoming signals to all first motors.

Weight Value (Absolute)	Velocity (cm/s)
0.16	40.1
0.15	40.88
0.14	42.62
0.13	45.95
0.12	not able to walk

Table 4.5: Measured velocities in relation to the given (absolute) values determining stride frequency. Higher values indicate a higher frequency and vice versa.

the difference between its maximum (0.3880150) and minimum (0.1982058) value and then dividing it by two. Hence, the amplitude of the examined signal is 0.2931104.

Table 4.4 lists frequency and amplitude for all according signals. To find out if the resulting gait could be accelerated by a higher stride frequency, the frequency of the signal to the first output neurons had to be increased. In [PHZ03] is described how this can be achieved by changing $w_{6,5}$, $w_{5,5}$, $w_{7,8}$ and $w_{8,7}$ (enumerations of neurons is according figure 4.4). In the initially used CPG the absolute value of these weights was 0.14, increasing them within certain ranges leads to higher frequencies and vice versa. Smaller and larger values were tested until the quality of the behavior worsened, i.e. the measured speed became slower. The results are listed in table 4.5. The fastest gait was achieved when the absolute value of the said weights was 0.13. The according frequency of the input signal to all fore legs (calculated as previously described) was 1.8 Hz.

neuron	bias
0	0
1	0
2	0
3	-0.08
4	-0.39
5	-0.
6	1.3020103564406793

Table 4.6: The bias values of the neurons of the foreleg of the manually improved neurocontroller, shown in figure 4.15. The enumeration of neurons is according figure 4.3 (A).

Concluding it can be said that a faster walking speed was achieved by a lower stride frequency.

4.4 The Final Neurocontroller and Speed

The simplified neurocontroller was altered according to the results of section 4.2. That is, parameters were adjusted to achieve:

- a straighter walking trajectory,
- a larger stride length, and
- an appropriate stride frequency.

The resulting neurocontroller is depicted in figure 4.15 and all according bias values are listed in tables 4.6,4.7 and 4.16. The enumerations of neurons in these tables is according the illustrations of fore- and hind legs in figure 4.3 and that of the coupling module according to figure 4.4. It leads to a walking speed which is 16.15% faster than the originally evolved neurocontroller (neurocontroller 4320).

neuron	bias
0	0
1	0
2	0
3	0.1
4	-0.93
5	0.05
6	0

Table 4.7: The bias values of the neurons of the hind leg of the manually improved neurocontroller, shown in figure 4.15. The enumeration of neurons is according figure 4.3 (B).

Team	Speed
JollyPochie	405mm/sec
NUbot	410mm/s
UPenn04	415mm/s (estimated 430mm/s)
German Team	400+-3mm/s

Table 4.8: Published speeds of walking behaviors of Robocup teams.

After the simplified neurocontroller had been improved it was tested on the hardware. The measured top speed of the generated gait by the controller on the physical robot was 47.34 cm/s. In table 4.8 the published velocities of walking behaviors of some Robocup Teams are listed. Due to the fact, that the used Aibo model, ERS-7, is new, not many teams have published their results yet.

The speed attained on the robot by means of the neurocontroller is faster than all published results listed in table 4.8.

As shown in section 4.2.2 the sensory input incorporated in the hind leg network is not necessary for a walking behavior. Thus a neurocontroller that does not require any sen-

sory input for generating a walking behavior was produced. It requires only 20 neurons and 42 synapses to control 12 degrees of freedom of a quadruped robot. In the generated walking behavior two different step cycles for fore- and hind legs are carried out, in which fore legs are observed to pull and hind legs push.

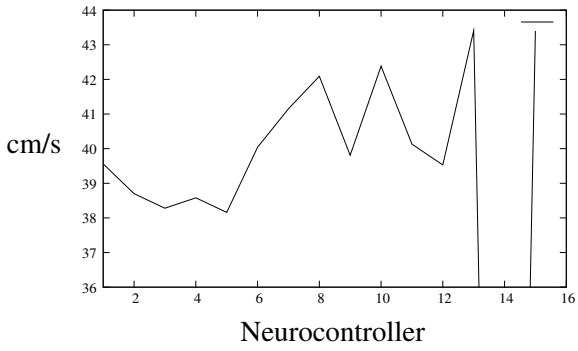

Figure 4.13: Measured speed achieved by each of the neurocontroller shown in table 4.3.

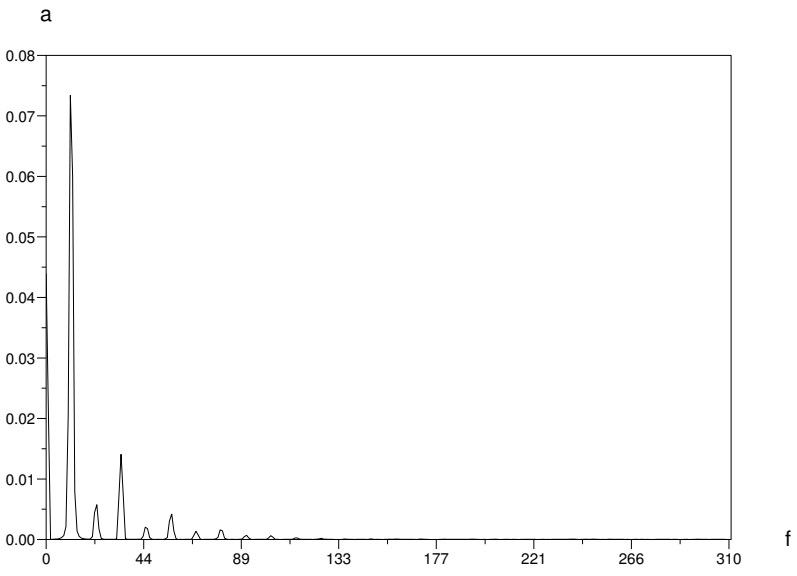

Figure 4.14: Fourier-spectrum (positive frequencies only) of a 5-second signal of the first output neuron for the right foreleg. Axis f denotes frequency and a amplitude.

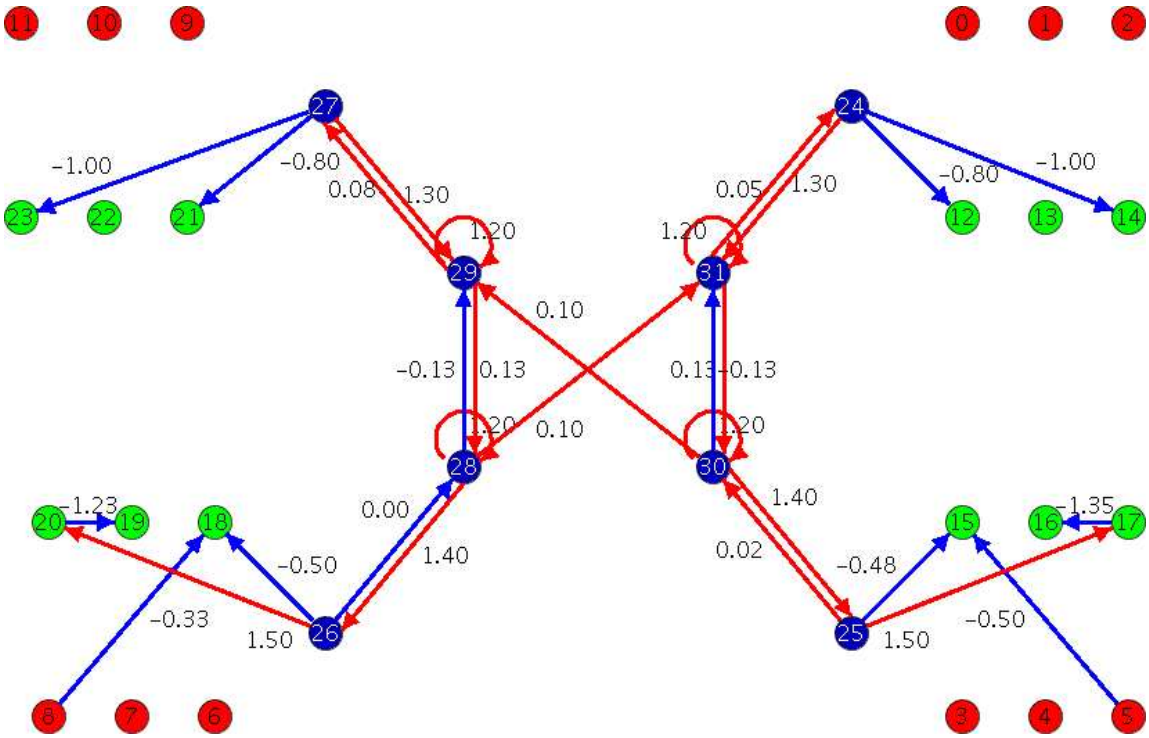

Figure 4.15: The final neurocontroller.

neuron number	bias of improved cpg
1	0
2	0
3	0
4	0
5	0.01745249286257482
6	0.005292848975793777
7	0.0037281823785782577
8	-0.004426097518234096

Figure 4.16: Bias values of the manually improved CPG, shown in figure 4.15. The enumeration of neurons is according figure 4.4.

Chapter 5

Discussion

In the previous chapters the used tools and conducted experiments were described that lead to a controller for a quadruped robot. The interpretation of the results presented in chapter 4 is the content of this chapter. Also the assets and drawbacks of the used tools are discussed. General conclusions that can contribute to future work are drawn. Finalizing the main achievements and conclusions for future work are summarized.

Artificial Evolution in combination with a simulator was successfully employed to generate a neurocontroller for a fast walking behavior for a quadruped robot. It can therefore be concluded that these tools state an adequate means for this purpose. However in section 3 it was described that not all conducted experiments lead to usable results. In this chapter possible reasons for success and failure of these experiments are identified. Also general asserts and drawbacks of the used tools are discussed. The results presented in chapter 4 are interpreted. The order in which the said items are addressed is analogous to the order in which they appeared in this work.

5.1 Artificial Evolution

Artificial Evolution was successfully employed as tool for generating a neurocontroller for a fast walking behavior. In section 3.3.2 it was found that the evolutionary process could significantly be accelerated by providing suitable initial leg networks. They caused the legs of the simulated robot to carry out inverted pendulum like movements. It is assumed here, that the increase in evolution speed resulted from the cyclic character of this movement. Walking bases on step cycles, which, as the name implies, must be cyclic. This feature did not have to evolve time consumingly, but was already introduced by the provided network structure. A "real" step cycle could evolve from this movement incrementally. It was also found that pendulum like movements with smaller deflection angles were more suitable. If the angles were chosen too big the robot fell over what sometimes led to an "extinction" of this behavior and thus the possible benefit of introducing the initial leg networks was lost.

It might be argued that this form of imposing an initial structure contradicts to what was said to be one of the great benefits of AE, namely that it can lead to behaviors without employing a priori knowledge or bias. That this is not the case, when the network is allowed to be changed by evolution, can be seen when considering the leg networks that resulted from the initially imposed structure (see figure 3.10). Comparing them to the start networks (figure 3.6) shows that nothing of the initial structure remained.

In section 3.3.2 it was explained that barriers were incorporated into the environment of the simulated robot. Therefore a walking behavior was expected that was reactive to its environment. That this was not the case (the evolved neurocontroller 4320 does not require sensory input as shown in section 4.2.2) was surprising at first. It is assumed that the barriers were so low that, when all legs were lifted high enough, no sensory input was needed to cross the barriers.

5.2 Simulator

Artificial Evolution was carried out on a simulator, instead of the real robot. The advantages, as listed in section 2.4.1, prevent the hardware from damages and to save time, applied and contributed to the results. The transfer of the neurocontrollers to hardware could easily be accomplished, i.e. no further manipulations of the neurocontroller were necessary.

It was stated in 2.4.1 that the simulator was supposed to contain all relevant features of the particular hardware of interest. It was found that contact sensors could not be modeled well with the current version of the used program YARS. This could be improved during future work. Furthermore in section 3.3.2 it was observed that neurocontrollers that gained a higher fitness in evolution did not always lead to a faster walking behavior on the real robot. It was observed that for the speed of walking it was very important in which angle hind legs contacted the ground. If in an appropriate angle, the paws of the real Aibo could contribute to the force exerted to the ground and thus propel the robot forward with a greater velocity. The simulated robot did not have paws (compare figure 3.1) which was later found to be a drawback. They certainly constitute a relevant feature for fast walking and should be incorporated in the simulation of the robot.

The neurocontroller generated by AE in simulation performed well (39.56cm/s) but a considerably gain in speed (16.15%) was attained when adjusting parameters manually, see section 4.3. In section 4.3 it was argued that this was justified because no more adaptation of the neurocontroller takes place on the hardware. Judging by the gain of speed after the optimization procedure, it is concluded here that an optimization problem like that of a fast gait is so hardware-specific that it cannot be modeled adequately by such a simulation as postulated in section 2.4.1. It is therefore suggested to combine Artificial Evolution applied to a simulator with a machine learning approach that can be carried out on the hardware. In such a setup AE can produce a robust behavior that can then be “fine tuned” for a specific robot. That way the benefits of both approaches can be used and their drawbacks avoided.

5.3 The Neurocontroller

Incorporation of a CPG in the coordination module (comp. section 3.3.1) lead to a relatively quick and good solution of the problem (comp. section 3.3.2). Other approaches did not lead to equally good results, as described in chapter 3. This supports the hypotheses that a CPG states an adequate mean for fast walking. It is also shown in section 4.2.2 that no sensory information for the generation of a walking behavior is needed, when a CPG is present.

In section 4.1.1 it was shown that both leg networks can be transformed into structurally equal networks. The different step cycles (pulling for the fore legs and pushing for the hind legs) were the result of only one different sign of the weight of the synapse to the third output neuron of the hind leg (see section 4.2.2). This shows that no structurally different leg modules are required for fore- and hind legs in quadrupedal systems.

5.4 The CPG

In section 4.2.3 it was described how CPG output was altered by the input of the interneurons, which lead to a lower output signal frequency. It indicated that a lower stride frequency (comp. section 4.3.3) contributed to a faster walking behavior. This was verified in section 4.3.3. The question why a lower stride frequency lead to a faster walking speed arises. It can be explained by the fact that the output signal which determines the positions of the motors changes less often with a lower frequency. Thus, the motors have more time to reach the desired positions provided by the signals, and this leads to a larger stride length. It can be concluded that an increase in speed by increasing the stride frequency was impeded by hardware limitations. Faster motors would probably have lead to a faster gait when using the higher output signal frequency. So the gain of speed in walking was due to a larger stride length.

5.5 Summary and Future Work

A neurocontroller for a fast walking behavior for the pet dog Aibo of Sony was generated by means of a simulator and Artificial Evolution. It was simplified, successfully transferred to hardware, analyzed and manually improved to generate a faster walking behavior. The velocity of the resulting gait is comparably fast (top speed: 47.43cm/s, average speed over 10 runs: 45.95cm/s). The final neurocontroller requires 22 neurons and 44 synapses to control 12 degrees of freedom of a quadruped robot to. It was shown that a (slower) walking behavior was carried out when the controller was reduced to 20 neurons and 42 synapses. For the latter no sensory input was required.

It is suggested to combine the used methods, AE and a simulator, with a machine learning approach that can be conducted on the real robot for an optimization of a behavior. For future work it is proposed to add a tropism to the walking behavior so that the robot can move directively towards a given goal.

Chapter 6

Acknowledgment

Without the help and support of many people this work would not have been possible. Therefore I want to thank:

- My family for their patience and moral support.
- Keyan Zahedi for being an encouraging, diligent and always available supervisor.
- Dr. Manfred Hild for the time he devoted me explaining the simplification of networks and for many other valuable suggestions.
- Prof. Pasemann for inspiring the hand-designed of networks and for supervising this work.
- Prof. Paulus for supervising this work and for the fast and reliable replies to my numerous emails.
- The INDY team for an inspiring atmosphere.
- All members of the GermanTeam, especially Uwe DÂ¨uffert and Max Risler for writing the walking engine.

Bibliography

[aib05a] 2005. http://www.sony.net/Products/aibo/.

[Aib05b] 2005. http://www.aibo-freunde.de/portal.php.

[BI00] Aude Billard and Auke J. Ijspeert. Biologically inspired neural controllers for motor control in a quadruped robot. In *Proceedings of the International Joint Conference on Neural Networks, Com*, volume VI, pages 637–641. IEEE Computer Society, 2000.

[BM98] A. Büschges and A. El Manira. Sensory pathways and their modulation in the control of locomotion. *Curr. Opin. Neurobiol.*, 8:733–739, 1998.

[BQRC97] Randall D. Beer, R. D. Quinn, R. E. Ritzmann, and R. E. Chiel. Biologically inspired approaches to robotics. *Communications of the ACM R. D., Beer; R. D., Quinn; H. J., Chiel; R. E., Ritzmann*, 40:30–38, 1997.

[Bro89] Rodney Brooks. A robot that walks; emergent behaviors from a carefully evolved network. *Neural Computation*, 1:253–262, 1989.

[CB97] H. J. Chiel and R. D. Beer. The brain has a body: adaptive behavior emerges from interactions of nervous system, body and environment. *Trends in Neuroscience*, 20(12):553–557, 1997.

[CV04] Sonia Chernova and Manuela Veloso. An evolutionary approach to gait learning for four-legged robots. In *In Proceedings of IROS'04*, September 2004.

[DA01] P. Dayan and L. F. Abbott. *Theoretical Neuroscience*. MIT Press, 2001.

[Del99] Fred Delcomyn. Walking robots and the central and peripheral control of locomotion in insects. *Autonomous Robots*, 7:259–270, 1999.

[Dü04] Uwe Düffert. Modellierung und Optimierung von Roboterbewegung. Master's thesis, Humboldt-Universität zu Berlin, 2004.

[ea99] G. S Hornby et al. Autonomous evolution of gaits with the sony quadruped robot. In *Proceedings of 1999 Genetic and Evolutionary Computation Conference (GECCO)*, 1999.

[EFL+04] Elger, Friederici, Luhmann, von der Malsburg, Menzel, Monyer, Fösler, Roth, and Scheich. Das Manifest. *Gehirn und Geist*, 6:30–34, 2004.

[FKC03] Y. Fukuoka, H. Kimura, and A. Cohen. Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts. *The International Journal of Robotics Research*, 22(3-4):187–202, March-April 2003.

[Ful93] Robert J. Full. *Biological Neural Networks in Invertebrate Neuroethology and Robotics*, chapter Integration of Individual Leg Dynamics with Whole body Movement in Arthropod Locomotion., page pp. 9. Academic Press, 1993.

[Ger05] 2005. http://www.germanteam.org/.

[GT204] Germanteam robocup 2004, 2004.

[Har93] I. Harvey. *The Artificial Evolution of Adaptive Behaviour*. PhD thesis, The University Of Sussex, 1993.

[HDI00] B. Hengst and D. et al. D. Ibbotson. The unsw united 2000 sony legged robot software system. Technical report, School of Computer Science and Engineering University of New South Wales, Sydney, 2000.

[HFK86] S. Hirose, Y. Fukuda, and H. Kikuchi. The gait control system of a quadruped walking vehicle. *Advanced Robotics*, 1:289–323, 1986.

[Hil89] Milton Hildebrand. The quadrupedal gaits of vertebrates. *BioScie*, 39:766–77, 1989.

[Hoo00] S. L. Hooper. Central pattern generators. *Curr. Biol.*, 10:176–177, 2000.

[HPW+05] I. Harvey, E. Di Paolo, R. Wood, M. Quinn, and E. Tuci. Evolutionary robotics: A new scientific tool for studying cognition. *Artificial Life*, 11(1-2):79–98, 2005.

[HSYF05] G. S. Hornby, S. Takamura S, T. Yamamoto, and M. Fujita. Autonomous evolution of dynamic gaits with two quadruped robots. *IEEE Transactions on Robotics*, 21:402– 410, June 2005.

[Jac98] Nick Jacobi. Running across the reality gap: Octopod locomotion evolved in a minimal simulation. In *Proceedings of the First European Workshop on Evolutionary Robotics*, 1998.

[Kel95] J. A. Scott Kelso. *Dynamic Patterns The Self-Organization Of Brain And Behavior*. MIT Press, 1995.

[KFK01] H. Kimura, Y. Fukuoka, and K. Konaga. Adaptive dynamic walking of a quadruped robot using a neural system model. *Advanced Robotics*, 15:859–876, 2001.

[Kim] Kimura. Legged robots of the world. http://www.kimura.is.uec.ac.jp/faculties/legged-robots.html.

[KS04] Nate Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 2619–2624, May 2004.

[KU03] M. S. Kim and W. Uther. Automatic gait optimisation for quadruped robots. In *Proc. Int. Conf. Australasian Conference on Robotics and Automation, ACRA*, 2003.

[KZP04] Bernhard Klaassen, Keyan Zahedi, and Frank Pasemann. A modular approach to construction and control of walking robots. In *Robotik 2004*, VDI-Berichte, pages 633–640. VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, Verein Deutscher Ingenieure, 2004.

[lAD00] W. llg, J. Albiez, and R. Dillmann. Adaptive posture control of a four-legged walking machine using some principles of mammalian locomotion. In *Proceedings of the International Symposium on Adaptive Motion of Animals and Machines*, 2000.

[Leg] http://www.ai.mit.edu/projects/leglab/background/milestones.html.

[LM68] R. Liston and R. Mosher. A versatile walking truck. In *Proceedings of the Transportation Engineering Conference*, 1968.

[Mah03] Björn Mahn. Entwicklung von neurokontrollern für eine holonome roboterplattform. Master's thesis, Fachhochschule Oldenburg, 2003.

[Mod] Open-r sdk model information for ers-7. http://openr.aibo.com.

[ode05] 2005. http://ode.org/.

[Pas93] Frank Pasemann. Dynamics of a single model neuron. *International Journal of Bifurcation and Chaos*, 2:271–278, 1993.

[Pas96] Frank Pasemann. *Interne Repräsentationen - Neue Konzepte der Hirnforschung*, chapter Repräsentation ohne Repräsentation - Überlegungen zu einer Neurodynamik modularer kognitiver Systeme, pages 42–91. Suhrkamp, 1996.

[Pas02] Frank Pasemann. Complex dynamics and the structure of small neural networks. *Network : Computation in neural systems*, 13(2):195 – 216, 2002.

[PD97] Frank Pasemann and Ulf Dieckmann. Evolved neurocontrollers for pole-balancing. In *In Biological and Artificial Computation: From Neuroscience to Technology*, 1997.

[PHZ03] Frank Pasemann, Manfred Hild, and Keyan Zahedi. So(2)-networks as neural oscillators. In *IWANN (1)*, pages 144–151, 2003.

[PS99] Rolf Pfeifer and Christian Scheier. *Understanding Intelligence*. Cambridge, Massachusetts: MIT Press, 1999.

[PSHL01] Frank Pasemann, Uli Steinmetz, Martin Hülse, and Bruno Lara. Robot control and the evolution of modular neurodynamics. *Theory in Biosciences*, 120:311–326, 2001.

[Rai86] Marc H. Raibert. *Legged Robots That Balance*. MIT Press, 1986.

[Ree99] Richard Reeve. *Generating walking behaviours in legged robots*. PhD thesis, Universtity of Edinburgh, 1999.

[Rid99] Christian Ridderström. Legged locomotion control — a literature survey. Technical Report TRITA-MMK 1999:27, Dept. of Machine Design, Royal Institute of Technology, S-100 44 Stockholm, Sweden, November 1999. ISSN 1400-1179.

[Rö04] Thomas Röfer. Evolutionary gait-optimization using a fitness function based on proprioception. In *8th International Workshop on RoboCup 2004 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence. Springer, 2004.

[Rob] www.robocup.org.

[Sim94] Karl Sims. Evolving virtual creatures. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22, New York, NY, USA, 1994. ACM Press.

[SW89] Shin-Min Song and Kenneth J. Waldron. *Machines That Walk*. The MIT Press, 1989.

[vT04] Arndt von Twickel. Obstacle perception by scorpions and robots. Master's thesis, University of Bonn, 2004.

[wal]

[WT92] David Wettergreen and Chuck Thorpe. Gait generation for legged robots. In *Proceedings of the IROS '92 Conference*, volume 2, pages 1413 – 1420, July 1992.

[ZON02] Fabio Zonfrilli, Giuseppe Oriolo, and Daniele Nardi. A biped locomotion strategy for the quadruped robot sony ers-210. In *International Conference on Robotics and Automation*, 2002.