Thesis for the degree of Doctor of Philosophy

Generation and Optimization of Motor Behaviors in Real and Simulated Robots

KRISTER WOLFF

Department of Applied Mechanics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2006

Generation and Optimization of Motor Behaviors in Real and Simulated Robots

KRISTER WOLFF
ISBN 91-7291-867-5

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 2548
ISSN 0346-718x

Department of Applied Mechanics
Chalmers University of Technology
SE–412 96 Göteborg
Sweden
Telephone: +46 (0)31–772 1000

Printed by Chalmers Reproservice
Göteborg, Sweden 2006

In memory of my father

Generation and Optimization of Motor Behaviors in Real and Simulated Robots

KRISTER WOLFF
Department of Applied Mechanics
Chalmers University of Technology

Abstract

In this thesis, the problems of generating and optimizing motor behaviors for both simulated and real, physical robots have been investigated, using the paradigms of evolutionary robotics and behavior-based robotics.

Specifically, three main topics have been considered: (1) On-line evolutionary optimization of hand-coded gaits for real, physical bipedal robots. The evolved gaits significantly outperformed the hand-coded gaits, reaching up to 65% higher speed. (2) Evolution of bipedal gait controllers in simulators. First, linear genetic programming was used with two different simulated bipedal robots. In both these cases, the gait controller was evolved starting from programs consisting of random sequences of basic instructions. The best evolved programs generated stable bipedal locomotion, keeping the robot upright and moving indefinitely. However, the evolved gaits were not very human-like. Thus, a different approach, inspired by the neural mechanisms involved in the locomotion of biological organisms, was tried. Here, both the structure and parameters of a central pattern generator network, controlling the locomotion of a simulated robot, were optimized using a genetic algorithm. The evolved controllers generated a stable human-like gait and were also able to handle gait transitions. (3) Behavior selection in autonomous robots, using the utility function method. In particular, the performance of the method as a function of the polynomial degree of the utility functions was investigated. It was found that adequate behavior selection systems can be found rapidly for low polynomial degrees (1-2), but also that the best solutions can only be obtained by using a higher polynomial degree (3-4). Furthermore, the performance of different evolutionary algorithms in connection with the utility function method was also investigated and, somewhat surprisingly, it was found that the standard method, employing a simple genetic algorithm, generally outperformed the modified methods.

Keywords: autonomous robots, bipedal robots, evolutionary robotics, behavior selection, behavior-based robotics, linear genetic programming.

List of publications

This thesis is based on the work contained in the following papers, referred to by Roman numerals in the text:

I. Evolution of efficient gait with humanoids using visual feedback
K. Wolff and P. Nordin. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pages 99–106, Tokyo, Japan, November 22–24, 2001.

II. Learning biped locomotion from first principles on a simulated humanoid robot using linear genetic programming
K. Wolff and P. Nordin. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2723 of LNCS, pages 495–506, Chicago, IL, USA, July 12-16, 2003.

III. Evolution of biped locomotion using linear genetic programming
Submitted to *Autonomous Robots*.

IV. Behavioral selection using the utility function method: A case study involving a simple guard robot
M. Wahde, J. Pettersson, H. Sandholt and K. Wolff. In *Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment*, pages 261–266, Fukui, Japan, September 20-22, 2005.

V. Behavioral selection in domestic assistance robots: A comparison of different methods for optimization of utility functions
J. Pettersson, D. Sandberg, K. Wolff, and M. Wahde. In *Proceedings of the 2006 IEEE International Conference on Systems, Man, and Cybernetics*, pages 4904–4909, Taipei, Taiwan, October 8–11, 2006.

VI. Structural evolution of central pattern generators for bipedal walking in 3D simulation
K. Wolff, J. Pettersson, A. Heralic, and M. Wahde. In *Proceedings of the 2006 IEEE International Conference on Systems, Man, and Cybernetics*, pages 227–234, Taipei, Taiwan, October 8–11, 2006.

VII. Evolutionary optimization of a bipedal gait in a physical robot
K. Wolff, D. Sandberg, and M. Wahde. Submitted to *IEEE Transactions on Robotics*.

Acknowledgements

I would like to express my gratitude to all those who have made the writing of this thesis possible: First of all I would like to thank my supervisors Peter Nordin, Kristian Lindgren, and Mattias Wahde. Peter and Kristian took care of me during the first half of my life as a doctoral student at Chalmers, and Mattias helped me all the way through the second half it.

I would also like to thank Göran Wendin at the Department of Microtechnology and Nanoscience, who took a great economic responsibility for my well-being during the (rather long) transition period from the Department of Physics, via the Department of Microtechnology and Nanoscience, to the Department of Applied Mechanics. Thanks also to the Carl Trygger Foundation for partially funding my research project.

I would also like to thank all my present and former colleagues and coworkers at the Department of Applied Mechanics, and all my former colleagues at the Departments of Microtechnology and Nanoscience, and Physical Resource Theory, for creating nice and enjoyable working environments at Chalmers.

I would also like to thank all the members of my family for always supporting me. Finally, I wish to give a special thanks to Eva-Lena for always being by my side. Thank you.

Technical terms used in the thesis

Contents

Chapter 1

Introduction and motivation

This thesis has two main themes, namely (1) the generation of efficient **gaits** for **bipedal robots**, and (2) **behavior selection** in **autonomous robots**. These two themes are united by the fact that **bipedal gaits** are **motor behaviors**, which can be subject to behavioral selection in robots. Furthermore, the methods used in these studies strongly draw upon biological inspiration, in particular **evolutionary algorithms** (EAs).

1.1 Research objective

The main objective with the work presented in this thesis has been to investigate the application of certain biologically inspired computation methods in the field of autonomous robots, with particular emphasis on bipedal walking robots. A particularly important aim has been to bridge the gap between simulations and physical robots, by studying the feasibility of applying evolutionary algorithms directly to physical robots, rather than only evolving in simulations and then attempting to transfer the results to physical robots.

Specifically, the following three topics have been considered: (1) Online evolutionary optimization of hand-coded gaits for real, physical bipedal robots (see Papers I and VII). (2) Generation of bipedal gait controllers in simulators, by means of artificial evolution (see Papers II, III, and VI). (3) Investigation of a method for behavior selection in autonomous robots (see Papers IV and V).

1.2 Autonomous robots

The work described in this thesis applies to autonomous robots. An **autonomous robot** is intended to move around freely in **unstructured environments**, and to perform desired tasks without continuous human guidance [7]. Whereas autonomous robots have appeared rather recently, stationary robots, by contrast, have been utilized in industry since the early 1960s in repetitive and monotonous tasks, such as spray painting, welding, and assembly [86]. Manipulators, e.g. **robot arms**, such as the SCARA type [120] **manipulator**, are common examples of stationary robots. Such robots are rarely capable of modifying their behavior to cope with unpredictable events, and therefore require a highly structured and controlled working environment, called a **robot cell** [70], in order to carry out their tasks in a safe and reliable way.

Unstructured environments change rapidly, in an unpredictable way. Autonomous robots thus cannot rely on pre-defined maps of the environment, like industrial robots generally do. Instead, an autonomous robot must be able to modify its behavior according to the changes that occur in the environment. Such adaptability is one of the most important traits of biological organisms, such as humans, and it is therefore logical to base the development of autonomous robots on concepts from biology. In particular, the fields of **neuroscience**[1], **genetics**[2], and **ethology**[3], have been used as sources of inspiration in the development of **behavior-based robotics** (BBR), which is a commonly used approach in the field of autonomous robots [2, 16, 18, 19, 21, 31, 32, 80, 81].

Autonomous robots exist in many shapes, with **wheeled robots** and **walking robots** being the two main categories. Lawn mower robots and vacuum cleaning robots [43] are examples of autonomous, wheeled robots, which are available on the market today. Other potential applications of wheeled robots include e.g. guarding and surveillance, transportation, and search and rescue missions [112]. A simple guard robot has been the topic of Papers IV and V.

Walking robots exist mainly as **bipedal robots** [56, 65, 109, 133] and **quadrupedal robots** [44, 45], but there are walking robots with more than four legs as well [73, 99]. In this thesis, however, only walking robots with two legs have been considered (see Papers I, II, III, VI, and VII).

[1] A research discipline that studies the nervous system of biological organisms.
[2] The scientific study of genes and heredity.
[3] The study of the behavior of animals in their natural environment.

1.3 Motivation

The underlying paradigm of all the work presented in this thesis is the behavior-based approach [2], which is rooted in biology and, unlike methods based on classical artificial intelligence [90, 108], is well suited for coping with rapidly changing (unstructured) environments. As mentioned above, the ability to function in such environments is particularly important for autonomous robots, which are expected to operate together with people and must thus be able to handle unpredictable events. Biologically inspired methods allow, for example, development of bipedal gaits and other motor behaviors directly in **hardware**, i.e. in actual robots (see Papers I and VII), so that detailed modelling of the kinematics and dynamics of the robot, and the difficult problem of transferring results from simulations to hardware, can be avoided. In other situations, where the approach just mentioned is not applicable, for example due to long evaluation times in hardware, simulations (see Papers II, III, IV, V, and VI) constitute an important first step towards implementation in real robots.

The traditional approaches to **bipedal gait synthesis**, which are based on classical control theory, rely on the calculation of **reference trajectories**, such as trajectories of joint angles, for the robot to follow [59, 78, 134]. Reference trajectories can rarely be specified in unstructured environments. The biologically inspired methods for bipedal gait synthesis suggested in this thesis (see Papers I, II, III, VI, and VII) do not require specification of any reference trajectories.

In the biologically inspired BBR approach, which has become a dominating paradigm in robotics control in recent years, a key issue is the problem of activating an appropriate behavior at all times. As long as the repertoire of behaviors in a robot is relatively small, the method for selection among the behaviors can be manually designed, but when the number of behaviors and the complexity of the problem grow larger, that task becomes very cumbersome.

In order to address the problems associated with **behavioral organization** in complex problem domains, the **utility function** (UF) method has been developed [12, 84, 85, 127]. In the UF method, behavior selection is based on the value of utility functions. The evolution of such functions is an integral part of the method. In many other behavior selection methods the user has to specify manually a large number of parameters, which is a notoriously difficult task [13, 127]. In the UF method the user is released from that task, since an EA is used for evolving most parameters (see Papers IV and V).

1.4 Contributions

The most important contributions of this thesis can be summarized as follows:

(1) It is possible to obtain significant improvements of rudimentary bipedal gaits, by applying the process of artificial evolution (i.e. EAs) in real, physical robots. In particular, Papers I and VII concern experiments regarding optimization of hand-coded gait patterns, by means of an EA.

(2) The program for motor control of a simulated bipedal robot with a high number of degrees of freedom (DOFs) can be evolved without a kinematic model or any other *a priori* knowledge on how to walk (see Papers II and III). However, while enabling the robot to walk, the evolved control programs presented in Papers II and III did not produce a very human-like gait.

(3) By contrast, the use of control programs based on central pattern generators (CPGs) has been shown, in Paper VI, to generate human-like gaits which, moreover, can be easily adjusted with regard to walking speed and can also handle gait transitions. It has also been shown that evolutionary optimization methods are able to generate the structure *and* optimize the parameters of the CPG network controlling the robot.

(4) All of the above-mentioned cases were centered on gait generation. While gaits constitute a very important behavior for bipedal robots, a complete robot should also comprise other behaviors than those directly related to walking. In such cases, behavior selection becomes an important problem and, in Papers IV and V, some properties of the utility function (UF) method for behavior selection have been studied, for the case of wheeled robots. It was concluded that the UF method can generate better behavior selection systems if the **polynomial degree** of the utility functions is increased. On the other hand, for lower polynomial degrees, solutions can be found more rapidly (see Paper IV). The performance of different EAs in connection with the UF method was also investigated. It was shown that an ordinary **genetic algorithm** (GA) with fixed-length chromosomes performs at least as well as modified evolutionary methods in which the chromosomes are allowed to vary in size (see Paper V).

The author of the thesis was the sole contributor to Paper III, and the main contributor to Papers II and VII. In Papers I and VI, all authors contributed approximately equally to the papers, whereas the author of the thesis made more limited contributions to Papers IV and V.

Chapter 2

Behavior-based robotics

Intelligence may be defined as *the ability of a creature (such as an animal or robot) to survive and strive towards its goals in an unstructured environment*. Using this definition, **machine intelligence**, which is also referred to as **artificial intelligence**, could become a reality through the incremental construction of intelligent systems, starting from simple ones (roughly corresponding to insects), and gradually moving towards more complex systems. This approach to achieve machine intelligence emerged in the 1980s, and it was termed **behavior-based robotics** (BBR). This chapter gives a brief introduction to BBR, which constitutes an important cornerstone for the work presented in this thesis.

2.1 Approaches to intelligent systems

In the BBR approach, which was pioneered by Rodney Brooks [18, 19, 20], intelligent behavior is an emergent property arising from simple components, or processing units, operating concurrently, see Figure 2.1. This approach thus rejects the reliance of traditional symbolic AI upon constructing an explicit internal model of reality. In BBR in general, each component or part consists of a number of **task-achieving behaviors**, each designed (usually by hand) for a specific task. The activation of certain behaviors is triggered by specific sensor input [18], or, as described in Papers V and IV, by internal variables as well. By contrast, in the classical AI approach, the problem of controlling a robot in a real-world environment is formulated according to a vertical sense-plan-act structure [2], see Figure 2.2. Hence, this approach is sometimes also referred to as a **top-down** approach, while the BBR approach is referred to as a **bottom-up** approach.

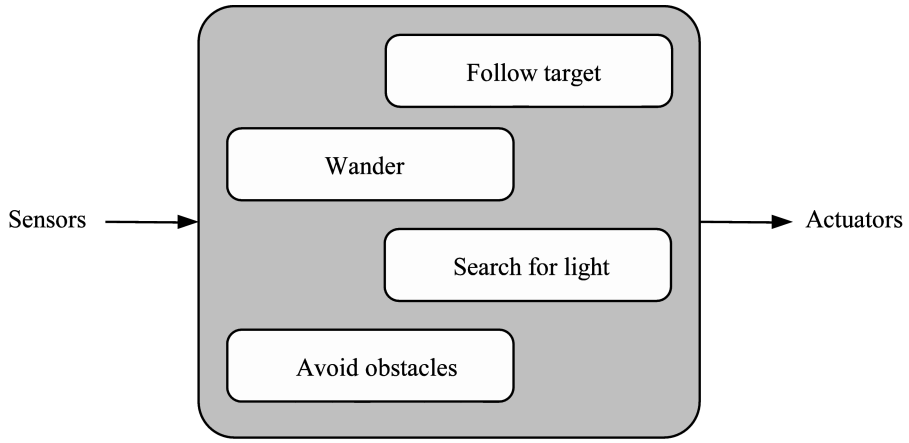

Figure 2.1: An example, comprising four behaviors, of a typical structure for a control architecture in the bottom-up approach (BBR).

2.2 Behaviors

In the BBR paradigm, the artificial brain of a robot contains a repertoire of behaviors. In isolation, such a behavior can be quite simple. Consider as an example the basic **obstacle avoidance** behavior, which can be formulated as follows: *if obstacle detected in front of the robot* [by the sensors] *then turn left for one second*. This behavior can easily be programmed by hand into the brain of a robot, since it is formulated as a simple if-then-else rule. In the UF method, see Papers IV and V, behaviors are normally constructed using combinations of such rules.

Rules like the one exemplified above, encoding a behavior for obstacle avoidance, may of course be evolved by an EA as well. Other possible representations of behaviors include generalized finite state machines (GFSMs) [101, 128], and recurrent neural networks (RNNs) [100], which are well suited when generating robotic behaviors using EAs.

Furthermore, when combining a simple behavior, such as the one described above, with one or several other basic behaviors, such as e.g. **follow target** or **charge battery**, it is possible to generate robotic brains capable of demonstrating complex overall behaviors. That is, tasks like patrolling an arena that contains obstacles and charging stations, as described in Paper IV, would then be possible to achieve for a robot.

However, in order to generate a complex, overall behavior by combining several simple behaviors in a robotic brain, some mechanism for behavior coordination, or **behavior selection** (see below), is needed.

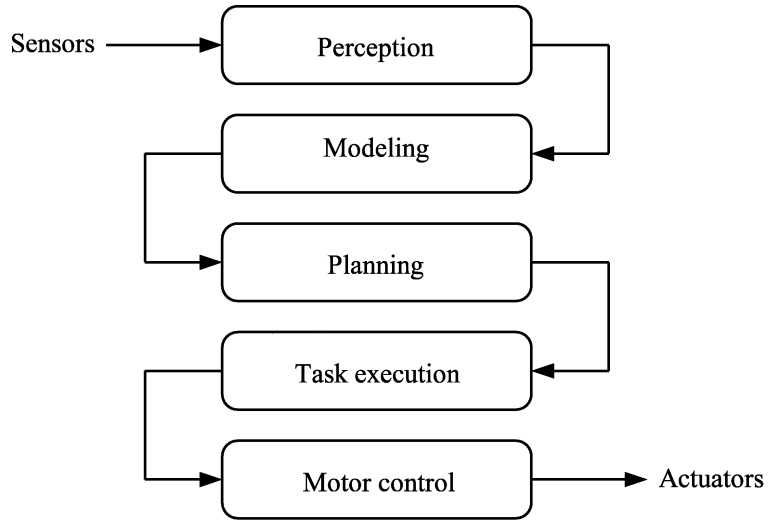

Figure 2.2: Decomposition of a robot control system into functional modules in the top-down approach (classical AI).

2.3 Behavior selection

A major issue in BBR is the process of determining which behavior should be activated at any given time, a problem which is commonly referred to as the **behavior selection problem**. In order to address this issue, a number of behavior selection methods have been proposed, see e.g. [103] for a review. It is common to divide methods of behavior selection into two major categories: **Arbitration methods** and **cooperative methods**, depending on the procedure used for selecting behaviors [121]. In arbitration methods, only one specific behavior is active at any given time. In cooperative methods on the other hand, the robot's action is a weighted sum of the actions suggested by several behaviors.

However, in many of the methods suggested to date, it is required that the user should specify the architecture and parameters by hand. Furthermore, as discussed above, behavior-based robots are usually expected to operate in unstructured environments. Thus, tuning parameters of the behavior selection system by hand is not an easy task. In order to overcome some of these obvious drawbacks of the hand-coded behavior selection methods, the **utility function method** (UF method) has been developed at Chalmers University of Technology, see [127] and Papers IV and V.

2.3.1 Utility function method

The UF method is an arbitration method in which behavior selection is based on the value of utility functions that are obtained through evolution. The concept of utility is inherited from economic theory and game theory [92] in which it is used as a common currency for comparing the outcome of possible situations. However, the concept of utility has recently been introduced in robotics as well [85, 127].

In the UF method, each behavior is associated with a utility function, whose variables are a subset of the state variables of the robot. There are three kinds of state variables: External variables (i.e. the readings of external sensors), internal physical variables (such as e.g. the battery level), and internal abstract variables (also called hormone variables), which roughly correspond to hormones in biological organisms. Thus, hormone variables are used for representing emotions, in a simplified fashion. For example, the detection of an object in front of the robot (triggering *fear*) may raise the level of a hormone variable which, in turn, may lead to an increase in the utility of an **obstacle avoidance** behavior, similar to the preparation for a fight-or-flight response caused by the release of adrenaline (epinephrine) in a biological organism finding itself in a fearful situation.

In general, each utility function is given by a polynomial ansatz, as shown in Papers IV and V. Behavior selection is a very straightforward procedure in the UF method: at all times, the behavior whose utility function takes the highest value is activated. The issue is how to specify the utility functions, so as to generate purposeful behavior selection. In the UF method, this is done using an EA, as described in more detail in Subsection 3.2.2. As in any EA, specifying the fitness function is crucial. In the UF method, the fitness is associated with the execution of a given **task behavior**. Other behaviors are then considered as **auxiliary behaviors**, i.e. behaviors which are also needed, such as **battery charging**, but do not increase the fitness of the robot. Once the fitness function has been defined, the task of the EA is to set the coefficients of the polynomial utility functions so as to maximize fitness. Note, however, that a polynomial representation of utility functions is not a fundamental restriction in the UF method.

Normally, each utility function is represented by a **complete polynomial**, including all terms up to and including a pre-specified degree d, as described in Paper IV. In that case, the structure of the utility functions remains intact during the optimization process. However, a different approach has been investigated in Paper V, where **non-complete polynomials** were used. This approach enabled structural modifications of the non-complete polynomials. Initially, it was assumed that this would lead to better perfor-

mance of the evolutionary optimization process, but that proved not to be the case. In fact, the **standard UF** method, which uses complete polynomials, outperformed the modified methods with non-complete polynomials; see Paper V, and Section 5.3 for further details.

Chapter 3

Evolutionary robotics

When designing a robot intented for operation in a realistic, unstructured environment, it is normally known what the robot should do, but perhaps not always *how* it should do it. **Evolutionary robotics** (ER) is an approach in which robots are considered as autonomous artificial organisms that develop their behaviors and skills in close interaction with their environment. Strongly inspired by biology and ethology, ER makes use of techniques from the field of **evolutionary computation** to generate robotic brains and, sometimes, bodies. Due to the large number of generations normally required for EAs, the ER approach often makes use of simulators in order to evolve robotic brains. The transition to real robots of brains evolved in simulators, however, is a far from trivial [68]. In all the work presented in this thesis, i.e. in Papers I through VII, the ER methodology has been applied.

3.1 Background and introduction

In general, the basic idea behind ER can be summarized as follows [95]: An initial population of randomly generated chromosomes, or individuals in the EA terminology (see below), is created. From each chromosome, a robotic control system can be decoded (and, sometimes, also a description of the robot's morphology). The resulting control system is then used in a robot, simulated or actual, in a given environment. Each robot is left free to move around in the environment, while its performance on various tasks is monitored. The fittest robots, according to an evaluation criterion set by the experimenter, are allowed to reproduce, with some probability p, by generating slightly modified copies of their genotypes. These copies are then inserted

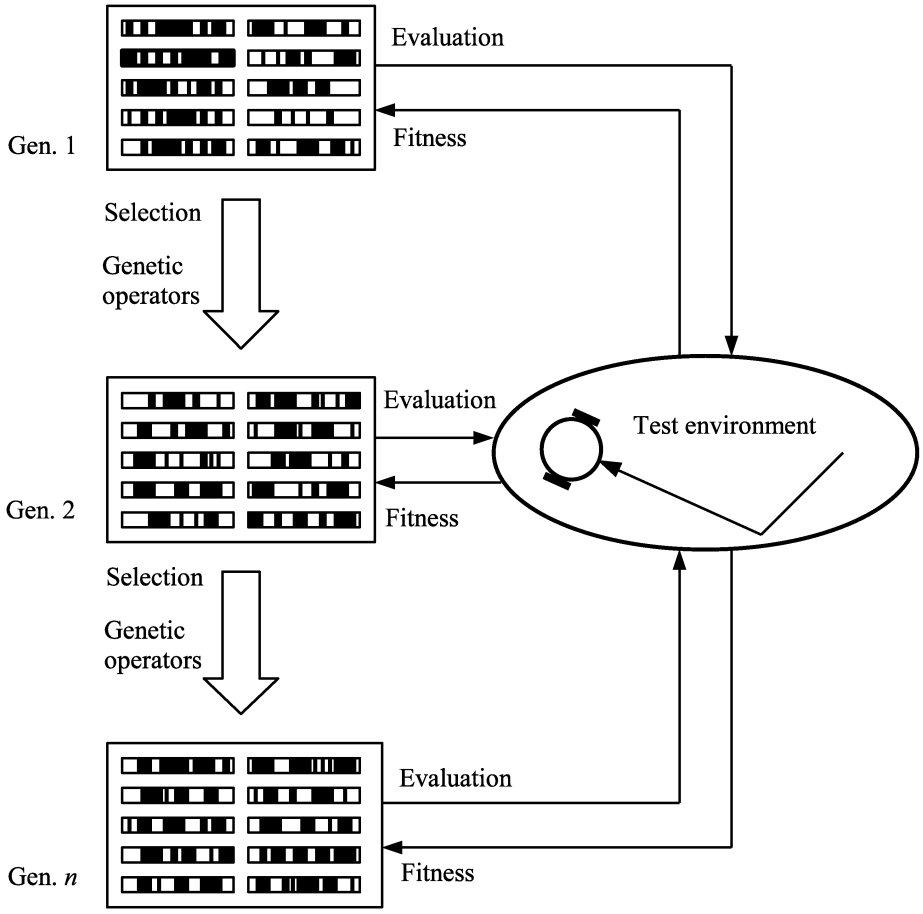

Figure 3.1: Schematic depiction of the ER concept. An initial population of chromosomes is generated randomly. From each chromosome, a robotic control system is decoded. The resulting control system is then evaluated in a robot in a given environment.

in the population, and the process is iterated until a stopping criterion, as defined by the experimenter, has been reached, see Figure 3.1. A detailed description of an ER application for bipedal gait generation is given in Paper III (Algorithm 2.3).

In an early example of an ER investigation, Sims [113, 114] simultaneously evolved both the bodies and the brains of artificial creatures, in computer simulation. More recently, both the morphology and the controllers of modular, physical robots have been generated using the ER methodology [104]. However, a more common approach to ER is to focus evolution on the generation of the robotic brain only, as e.g. in [41, 88, 107]. In ER, the

designer of a robotic system has, of course, complete freedom to choose which parts of the system should be evolved, and which parts should be designed by other means, e.g. by hand. Papers I through VII in this thesis have been focused on evolving the control system, or parts thereof, using a fixed body structure for the robot.

3.2 Evolutionary algorithms

Fundamental in the ER approach is the concept of biologically inspired computation, and especially evolutionary algorithms (EAs). EAs are a class of search and optimization algorithms, which derive their behaviors from the mechanisms of natural (or Darwinian) evolution. However, it is important to note that EAs are not accurate models of biological evolution, which, instead, is merely used as a metaphor and a source of inspiration when developing computational models. Thus, EAs are examples of biologically inspired computation methods.

The use of EAs, rather than traditional engineering optimization methods, in connection with the problems of generating or optimizing robot controllers, is motivated mainly by the following facts: (1) Classical engineering optimization methods generally rely on gradients (i.e. derivatives) of the objective function to guide the search process [29]. In the problems considered in this thesis, gradient information is rarely available, so that traditional methods cannot be applied. (2) Traditional engineering methods often tend to converge to local optima in the search space [29]. EAs, on the other hand, generally cope well with multidimensional search spaces with many local optima [3]. Note, however, that there is no guarantee for finding the (globally) optimal solution with an EA. (3) Many classical optimization methods have been designed to solve specific classes of search and optimization problems. Even though a particular traditional method may work very well [29], its use is limited to those kinds of problems for which it has been developed. EAs, by contrast, can easily be applied to a wide variety of complex optimization problems [42], with a minimal customization effort needed in order to adapt the algorithm to the problem at hand.

Furthermore, since the 1960s, the principles of natural evolution have been successfully utilized on computers to search for solutions to many different problems. Several research subfields, such as **genetic algorithms** (GAs) [57] and **genetic programming** (GP) [76] have emerged. Although they differ from each other in many aspects, they all mimic natural evolution. In the following subsections, some of the concepts of this field, of relevance to the topics of this thesis, will be discussed.

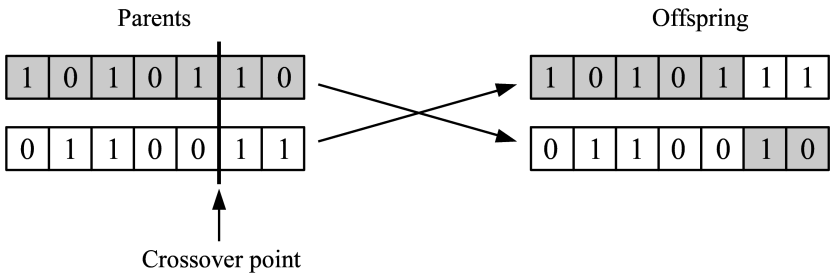

Figure 3.2: Crossover in a standard GA.

3.2.1 EA fundamentals

EAs constitute a vast topic, and only a brief discussion will be given here. See e.g. [3, 25, 89] for further details on the topic. However, two particular topics, i.e. a standard GA used in the UF method in Papers IV and V, and a GP variant known as **linear genetic programming** (LGP) used in Papers II and III, will be described in some detail.

The first decision that must be made when applying an EA concerns the **representation** of a solution. In standard GAs solution candidates are represented as fixed-length strings (chromosomes) of zeros and ones. In LGP, a linear program representation structure is used (see Subsection 3.2.3 and Papers II and III).

In EAs, an assembly of solution candidates, or individuals, is simply called a **population**. The standard procedure in EAs of using a population of candidate solutions, rather than a single candidate solution, is beneficial since it makes possible a parallel search through the space of possible solutions [47].

The **variation operator** is meant to increase the diversity in the population of solution candidates, i.e. its function is to ensure that new aspects of a problem are considered. In EAs, this operator is referred to as **mutation**.

The **conservation operators** are intended to decrease the diversity in the population, i.e. to consolidate what has already been learned. **Recombination**, or **crossover**, of two or more individuals is the primary tool for achieving this goal. Common recombination methods are one-point, two-point, or n-point crossover between two individuals. One-point crossover is illustrated for the case of GAs in Figure 3.2. Commonly, a subset of the population is transferred without change to the next generation, a procedure

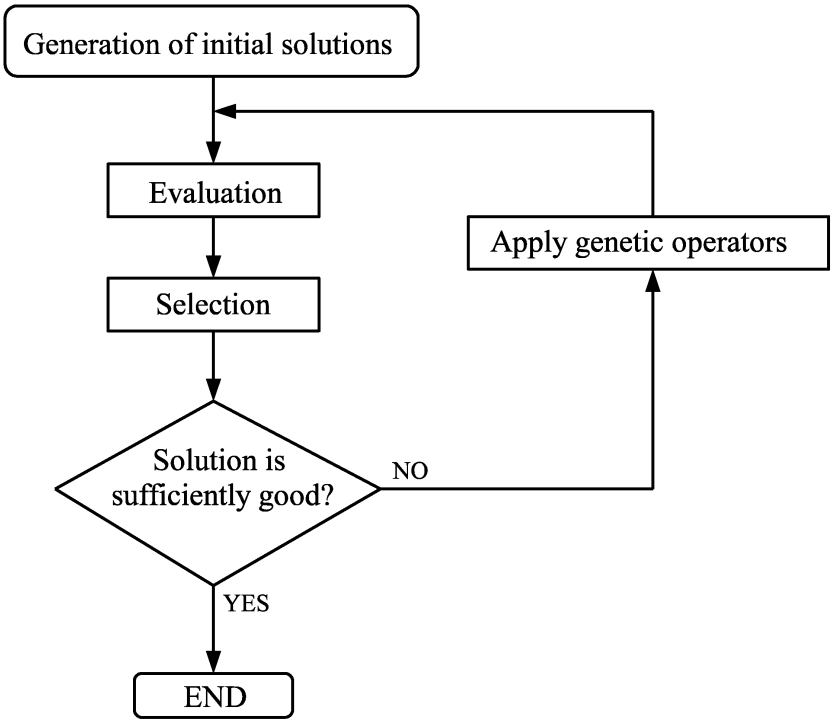

Figure 3.3: An illustration of the flow in a generic EA.

referred to as **elitism**.

In order to drive the evolutionary search process forward, there must be some **quality differentials**, as obtained from a graded **fitness function** (or **objective function**), which distinguishes a good solution candidate from a poor one. A discussion of fitness functions is given in Section 3.3 below.

In order to improve the solutions found by an EA, poor individuals have to be replaced by better ones, on the basis of the fitness function. Based on Darwin's tradition, this procedure is called **selection** [3]. A commonly used selection method in EAs is **fitness-proportional selection**. This method is also referred to as **roulette-wheel selection**, due to an analogy with a roulette wheel assigning a slot for an individual, sized in proportion to its fitness [25]. In the work presented in this thesis, however, another selection scheme has been used, namely **tournament selection** (see Papers I through VII). In tournament selection, a subset of the population is considered: q individuals are randomly picked from the population to compete against each other, and the best individual is selected with some probability $p_{\text{sel}} > \frac{1}{2}$. This process is repeated k times, and the selected individuals are allowed to mate

and pass on traits to the gene pool. The number q is called the tournament size. Typical values of q range from 2 up to around 10% of the population size. By varying the tournament size, the selection pressure can be adjusted. A small tournament size causes low selection pressure and a large tournament size causes high selection pressure [3]. The typical flow of a generic evolutionary algorithm is summarized in Algorithm 2.1 in Paper III, and in Figure 3.3.

3.2.2 Evolution of utility functions

In this subsection, an example application of a standard GA will be described, namely the evolution of utility functions (see Papers IV and V). In the UF method (see Subsection 2.3.1 and [127]), the selection of behaviors is based on utility functions, which are normally given in the form of polynomials. The detailed overall behavior of the robot is thus determined by the exact shapes of the utility functions. The task of the EA is to find optimal utility functions through the specification of polynomial coefficients.

The EA is initialized by randomly generating a population. Each individual of the population corresponds to a chromosome encoding the utility functions, one for each behavior. For example, a complete utility function polynomial U_c of two variables and degree 2 would take the following general form:

$$U_c(s_1, s_2) = a_{00} + a_{10}s_1 + a_{01}s_2 + a_{20}s_1^2 + a_{11}s_1s_2 + a_{02}s_2^2 \qquad (3.1)$$

where a_{ij} are constants to be determined by the EA. Such a polynomial ansatz has been used in Papers IV and V, but in the latter case also non-complete polynomials U_{nc}, i.e. polynomials lacking certain terms, have been used. In Paper V, the assumption was that the EA would gain in performance through the use of non-complete polynomials, since the search for such polynomials effectively takes place in a space of lower dimension than if complete polynomials were used.

Then the process of decoding and evaluating individuals, performing tournament selection and reproduction (by means of crossover and mutation), follows the standard EA procedure, as described above.

In its standard formulation, the UF method is based on evolutionary optimization of utility functions [127], but the UF method does not necessarily have to be tied to an EA. One could, of course, consider alternative methods for the optimization of the utility functions, or even let the user specify them by hand. However, hand-tuning of a large number of parameters is usually not an option, unless the problem is very simple. Furthermore, many other

optimization methods require that the optimization problem at hand should be mathematically well-posed, in order for the optimization method to be applicable. Otherwise, the problem needs to be re-formulated for numerical optimization [94]. As mentioned above, EAs are generally very flexible and efficient in discontinuous, multidimensional search spaces with many local optima. To set up an EA for the search of optimal solutions in a specific problem domain usually requires a minimum of adjustment and hand-tuning of the algorithm. For these reasons, an EA has been adopted as the preferred optimization tool in the UF method.

3.2.3 Linear genetic programming

When going beyond parametric optimization of a fixed structure, and focusing on the generation of the structure itself, the branch of EAs known as GP is a very flexible and powerful method, something that has been demonstrated in a wide range of scientific and engineering optimization problems [77].

In LGP, a program, or individual, consists of a linear list of instructions. Each **instruction** includes an operation on **source registers** and an assignment of the result to a **destination register** [17], see Figure 3.4 for a conceptual description of LGP. In the LGP implementations used in the work described in this thesis, 2-register and 3-register instructions have been used. 3-register instructions work on 2 source registers and assign the result in a third (destination) register, e.g. $r_i := r_j + r_k$. In 2-register instructions, the operator either requires only one operand, e.g. $r_i := \sin r_j$, or the destination register acts as a second operand, e.g. $r_i := r_j + r_i$.

The concept of the so called **virtual register machine** (VRM) [60] is useful when implementing LGP. A VRM consists of a set of m registers, each of which can hold a floating point number, and a set of n instructions. The registers constitute the machine's mutable memory, and all program input and output is communicated through the registers. Additionally, registers may also be used to supply the program with constants, which can be synthesized in otherwise unused registers. A **register state** $\mathbf{S_r}$ is a vector of m floating point values. Program inputs are supplied in the initial state $\mathbf{S}_{r,i}$ and outputs are taken from certain registers (the **output registers**) of the final register state $\mathbf{S}_{r,f}$. Besides the required **input registers**, additional **calculation registers** can be provided. Note that the LGP structure facilitates the use of multiple program outputs. By contrast, functional expressions like GP trees calculate one output only [17].

The implementation of LGP used here is explained in more detail in Section 2.2 of Paper III. In particular, note Example 2.1 in the paper, which

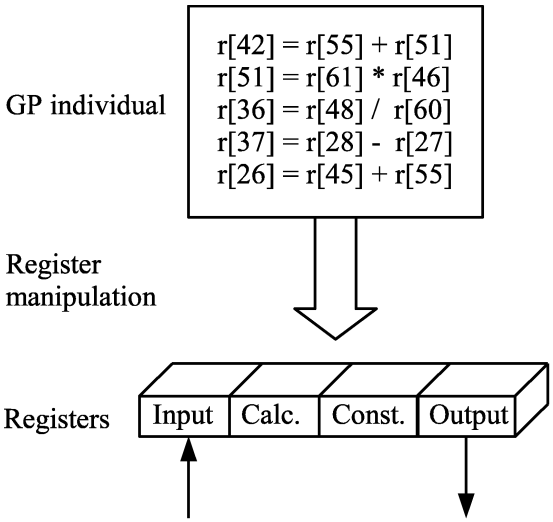

Figure 3.4: Schematic description of the evaluation of an individual in LGP. Input to the program (GP individual) is supplied in the input registers. The constant registers are supplied with values at initialization. When executed by the VRM, the GP individual manipulates the contents of the calculation registers. When the program execution has terminated (i.e. when the evaluation reaches the end of the program), the output is supplied in the output registers.

shows a variable-length genetic program in C notation. An early version of this LGP implementation is furthermore shown in Section 3.2 of Paper II. In these implementations, the internal representation of a genetic program (chromosome), as operated upon by the genetic operators, is an array of integers, as shown e.g. in Example 2.2 of Paper III. In this example, each row corresponds to an instruction, which, in runtime, must be translated by an interpreter in order to be applied to the problem domain. Program execution starts with the topmost instruction, and proceeds down the list, until the last instruction has been executed. The details of the decoding procedure are explained in Subsection 2.2.4 of Paper III. The process of random initialization of the linear chromosomes is furthermore described in Algorithm 2.2 in Paper III.

In the case of Papers II and III, steady-state tournament selection was used in order to select which individuals should be allowed to reproduce, and which individuals should be eliminated from the population.

The genetic operators, crossover and mutation, were used in a similar way in Papers II and III, to transform the initial, random populations in an evolutionary search process, into individuals that met the objective. Stan-

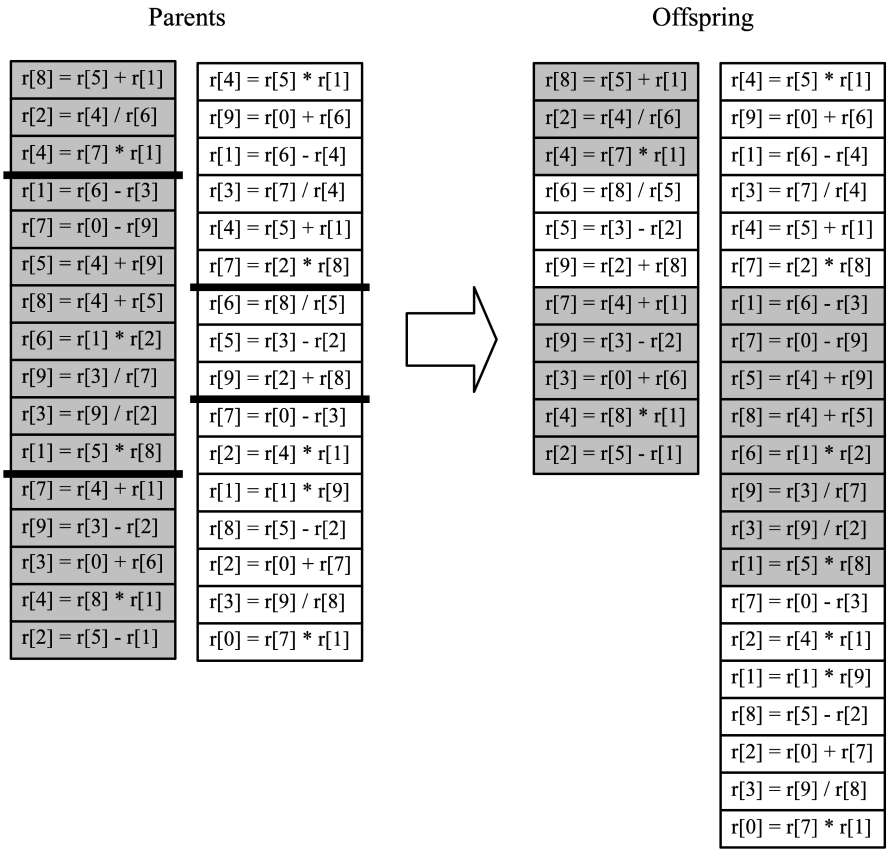

Figure 3.5: Two-point crossover in LGP.

dard crossover in LGP works by swapping linear genome segments of parent individuals. Two crossover points in the first parent's genome are randomly chosen, and then two other crossover points in the other parent's genome are chosen. The instructions between the crossover points are swapped, as shown in Figure 3.5. The mutation operator works on two different levels; it first selects which instruction should be mutated, and then it makes one or more changes to that instruction. There are normally three types of change that can occur, namely (1) a change in any of the registers to another randomly chosen register, (2) a change of the operator in the instruction to another randomly chosen operator, or (3) a change of a constant to another randomly chosen constant.

Finally, as in any case involving EAs to search for solutions in a complex problem domain, finding a good fitness measure that guides the evolution in the desired direction is crucial. This issue will now be discussed further.

3.3 Objective function aspects

When defining the fitness (objective) function there are certain issues to consider, which depend on the type of optimization problem to which the EA is applied. In the case of numerical problems, the fitness function is usually given by an explicit mathematical function. Consider e.g. the task of using an EA to find the maximum value of the function $f(x, y)$ on the domain $x, y \in [-d, d]$, where d is an arbitrary, positive real number, and the function $f(x, y)$ is given by:

$$f(x, y) = \frac{\sin x \sin y}{xy} \tag{3.2}$$

As depicted in Figure 3.6, the function $f(x, y)$ has many local maxima and minima. Furthermore, it can easily be shown that:

$$\lim_{x,y\to 0} f(x, y) = 1 \quad \text{and} \lim_{x,y\to\pm\infty} f(x, y) = 0$$

Functions like this one are often used as benchmarks when evaluating the performance of different optimization algorithms, e.g. EAs. The objective function should be such that the closer the individuals come to the global maximum, the higher fitness they get. In the case of function optimization, an obvious choice of fitness function would be the function itself, such that $F(i) = f(x, y)$ for the fitness of individual i.

In the case of more open-ended problems[1], such as evolving controllers for autonomous robots, as e.g. in Papers II, III, and IV there exists no such explicit mathematical formula, which could be adopted as the fitness function. Instead, the optimization algorithm uses a fitness function based on a simulation of a physical robot system. Or, as in the case of Papers I and VII, the individuals are actually evaluated on a real, physical robot. In such cases, the designer has to formulate the goal(s) of the robot in terms of mathematical functions that depend on quantities that can be measured by the robot (see e.g. [58, 87]), which indeed is a daunting task that must be carried out with extraordinary care. Among EA practitioners, it is a well-known fact that in real-world applications a large part of the time spent on a project is invested in defining the optimization problem [122], i.e. specifying the fitness function. In the work described in this thesis, that difficulty was apparent too.

[1] In this text, an optimization problem in which the structure of the optimal solution is unknown a priori, is referred to as open-ended. Thus, in such problems, the search for an optimum involves not only parametric changes, but also structural changes, of the system being optimized.

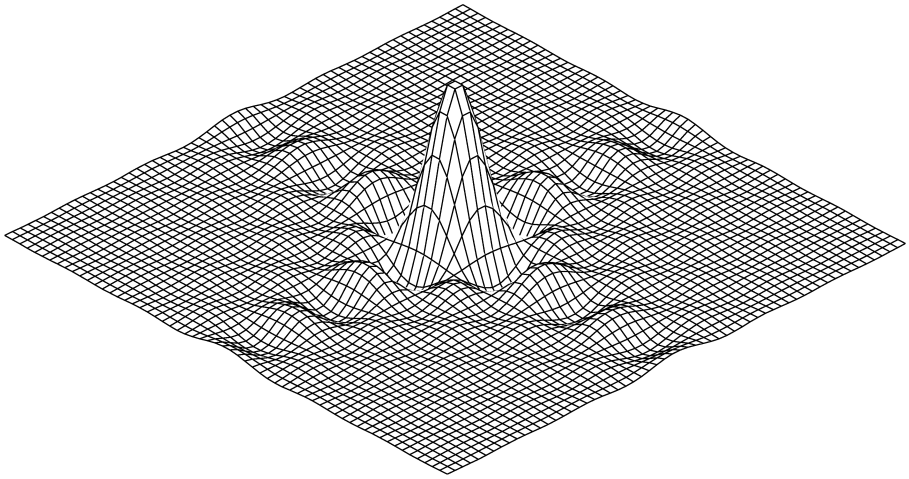

Figure 3.6: The function $f(x, y)$ depicted on the interval $x, y \in [-20, 20]$.

3.3.1 Fitness in LGP evolution

Especially in Paper III, much time was spent on finding a proper fitness measure, which could guide the evolution in the desired direction, as described in Subsection 3.1.1 of that paper. According to the aim of the evolutionary optimization task, i.e. to evolve controllers that make the simulated robot produce efficient locomotion behavior, a function that measures the distance covered by the robot during a trial over a fixed period of time, seems a natural choice. Furthermore, a human-like gait requires an upright posture of the robot. Therefore, an additional quantity, *height above ground of the robot's head*, described in Paper II, was used initially in the fitness measure. However, when introducing this measure, no improvements were seen in the robot's performance (even though the fitness values were numerically larger). Instead, cheating individuals started to emerge, as described in Subsection 3.1.1 in Paper III. Those individuals stood idle until just before the evaluation period ended, and then took a large leap forward.

Another example of cheating behavior emerged in the very beginning of the project described in Paper III. Due to the complex nature of the simulation package used [115], extensive hand-tuning of several numerical pa-

rameters of the software, related to the performance of the simulation, was necessary. Then, it turned out that some of the parameters were not set properly. Behaviors emerged from the evolution that caused the simulated robot literally to explode, with its body disintegrating. Due to the actual implementation, some of these individuals received unnaturally high fitness scores. This example shows how an EA can exploit opportunities, and thus evolve improper behavior of the individuals. Once the actual system parameters were tuned properly, this awkward behavior ceased to emerge. The fitness function was then finally defined as shown in Equation (2) in Paper III.

In this project, the intention was to favor **anthropomorphic**, i.e. human-like, gaits. Since human bipedal locomotion is very energy-efficient, compared to the gaits of most bipedal robots [27], an energy discharging function, described in Subsection 3.1.2 of Paper III, was added to the simulation: Each individual was supplied with a finite amount of energy, which was gradually used up as the robot moved. A few other strategies for guiding the evolution towards anthropomorphic gaits were investigated as well, as described in Paper III. However, the energy discharging function was the only addition to the original fitness function (e.g. Equation (2) in Paper III) that helped to improve the final result, although fully anthropomorphic gait was not obtained. The results of this investigation are further discussed in Section 5.2.

3.3.2 Fitness in CPG evolution

In the case of evolution of central pattern generators for walking, as described in Paper VI, the fitness measure was taken as the distance walked by the robot in the initial forward direction, x, decreased by the sideways deviation y. That is, the fitness F of individual i was given by the following equation:

$$F(i) = x - |y| \tag{3.3}$$

where x is the distance walked by the robot, and y is the sideways deviation. Additions to this fitness function, such as the energy discharging function in the case of LGP evolution described in the previous subsection, were not tried. Instead, a support structure, as described in Paper VI, was added in order to guide the evolution towards the desired goal. By contrast with the LGP evolution, in this case, the goal of generating human-like bipedal gait for the simulated robot was indeed reached. The study presented in Paper VI is further discussed in Section 5.5.

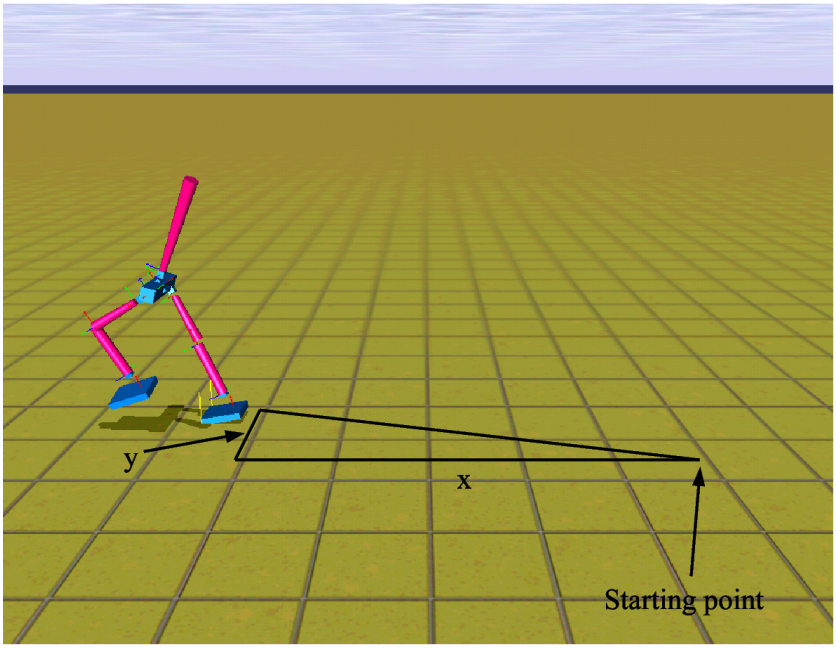

Figure 3.7: An illustration of the fitness measure used for CPG evolution.

3.3.3 Fitness in UF method

In the case of evolving behavior selection, as described in Papers IV and V, the fitness measure was simply taken as the time spent in the navigation behavior, since the task of the robot was to cover as much as possible of the arena. However, in order to avoid behavioral dithering, i.e. rapid switching between behaviors, the robot only received a fitness increase if it spent at least one full second executing the task behavior, i.e. navigation.

3.4 Evolution in robots versus simulation

In the work described in this thesis, evolution of robot controllers has been studied mainly using realistic, physical simulators, see Papers II, III, IV, V, and VI. Furthermore, in Papers I and VII evolution in real, physical robots has been investigated as well. Apart from using these two common approaches to ER, a third methodology also exists, where the entire evolutionary process takes place on a *population* of physical robots [40, 130]. However, this third approach was not used in this thesis, mainly due to the costs involved in acquiring and maintaining a population of different physical robots.

As described in Papers I and VII, evolution in real, physical hardware is indeed achievable. In general, evolution in hardware is much more challenging than evolution in simulators, for several reasons: First, evolution in real robots can be very demanding for the hardware (i.e., the robots), as described in Papers I and II. By contrast, in simulations, this problem simply does not occur. Second, the process of evolution in a simulator can relatively easily be parallelized, given that an appropriate computational resource is available. For example, in the case of evolving locomotion controllers described in Paper II, a variant of GP was implemented according to the parallelization scheme of the asynchronous island model [38]. In this model, the population is divided into a number of subpopulations, or **demes**, where each deme is assigned to a separate processor, and individuals are allowed to migrate from one deme to another, during evolution. A corresponding parallelization in the case of evolution in real, physical robots would be more difficult and costly. It would require multiple instances of the robot, as well as duplicate experimental environments. Third, evaluation of individuals in simulators can often be performed several times faster than real-time, which is not the case for evaluation of individuals in real robots: Evolution in physical robots is very time-consuming, something that normally restricts the number of evaluated generations considerably.

While evolution in simulators is more convenient from the experimentalist's viewpoint than evolution in physical robots, the simulation approach presents other problems. The main issue concerns whether the controllers obtained from the simulation can be transferred to a real, physical robot. This problem is referred to as the **reality gap** [69]. In order to overcome the difficulties associated with the reality gap, an approach named minimal simulations has been proposed [68]. However, this approach has not been used in here: Instead, in the investigations involving physical robots (see Papers I and VII), the evolutionary process has been executed directly in the robot, so as to acheive results that are directly applicable without the need for a tedious transition procedure.

Chapter 4

Bipedal locomotion

When considering bipedal walking it is of importance to take into account both the biomechanical aspects, as well as the neurophysiological foundations. This chapter begins with a brief discussion of **bipedal locomotion** from a biomechanical viewpoint, followed by a review of the neurophysiological underpinnings of the locomotor system of vertebrates. The aim of the studies described in Papers I, II, III, VI, and VII has been to investigate biologically inspired methods, as an alternative to classical control methods, for generating (or optimizing, in the case of Papers I and VII) gaits for bipedal robots. Both biologically inspired methods and classical methods are briefly described in the final section of this chapter.

4.1 Bipedal walking

The process of bipedal walking can be seen as continuously falling forward in a controlled way. At the beginning of a step, the free leg swings forward with bent knee and lifted foot in front of the supporting leg and places the heel on the ground. The body maintains its speed, and the forward leg then lifts the body up and provides support. In the next moment, the supporting leg is utilized as a lever for the propulsion; the leg stretches at the hip and the knee and the rear part of the foot sole bends away from the ground. Eventually, the body weight rests on the middle part of the foot sole. By rapidly bending the knee and the hip of the supporting leg and stretching the foot sole, the supporting leg is lifted from the ground as the body weight simultaneously shifts to the opposite leg, which under the described procedure swings forward and touches the ground. During walking, the body weight shifts from one supporting leg to the other, a process that

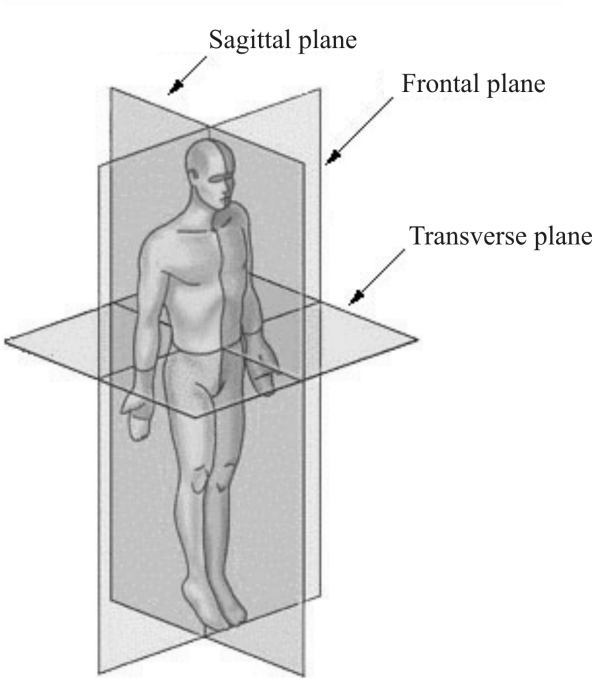

Figure 4.1: Depiction of the human anatomy reference planes: The **sagittal plane** is a vertical plane running from front to back through the body, the **frontal plane** is a vertical plane running from side to side through the body, perpendicular to the sagittal plaane, and the **transverse plane** is a plane parallel to the ground and perpendicular to the sagittal and frontal planes. Reproduced with permission from http://www.training.seer.cancer.gov; funded by the U.S. National Cancer Institute's Surveillance, Epidemiology and End Results (SEER) Program, via contract number N01-CN-67006, with Emory University, Atlanta SEER Cancer Registry, Atlanta, Georgia, U.S.A.

also causes the synchronous arm oscillations. During running, the supporting foot leaves the ground before the swing foot touches the ground.

4.1.1 Gaits

The cyclic pattern of foot placement during straight-line locomotion is called a **regular gait**, or a **gait** for short. An **irregular gait**, on the other hand, is used to handle specific situations such as e.g. side steps to avoid collisions with obstacles [86].

The advance of one leg from the moment when the foot leaves the ground,

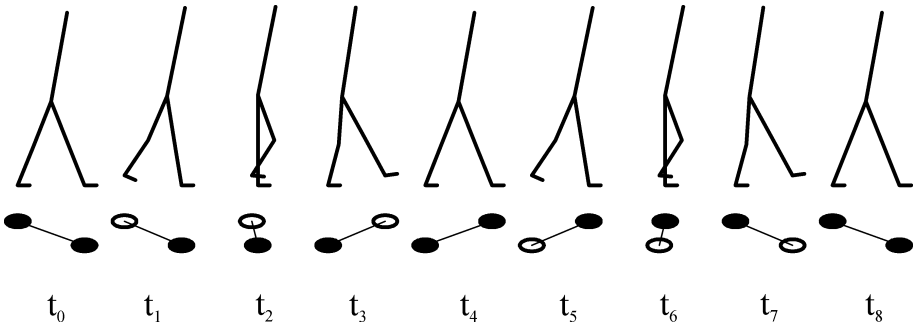

Figure 4.2: A stick figure illustrating a complete biped walking cycle, or a stride. It starts with a double-support phase at time t_0, followed by a single-support period (t_1, t_2, t_3), during which the right leg is in swing motion. At time t_4 there is another double-support phase, once again followed by a single-support period (t_5, t_6, t_7), during which the left leg is in swing motion. Eventually, the stride completes at time t_8, at which point a new double-support period begins. A filled ellipse underneath the stick figure indicates that the corresponding foot, right or left, is in contact with the ground.

until it touches the ground again, is called a **step**. A step with one of the legs immediately followed by a step with the other leg is referred to as a **stride**, which is the basic cycle of a bipedal gait. That is, a stride constitutes the whole cycle of movements from one occurrence of a specific leg movement to its repetition. A complete stride is depicted in Figure 4.2 (t_0-t_8).

In gait analysis it is convenient to divide the stride into two main phases; the **single-support phase** and the **double-support phase**. During the single-support phase one foot is in contact with the ground and the other foot is in swing motion, being transferred from the back to the front position. In the double-support phase, both feet simultaneously touch the ground. The so called **support polygon**, or support area, is formed by the intersection of all convex sets containing the set of all contact points between the ground and the feet (double-support phase, see Figure 4.3), or foot (single-support phase).

The standard human bipedal gait types are called **walk** and **run**. In the **walking gait**, the corresponding muscle groups of each leg are activated periodically, which leads to a half-period phase shift of the leg movements in the gait cycle. The **running gait** closely resembles walking, except that there is a significant flight phase once every beat.

Bipedal gaits involve a large range of complex movements of the musculoskeletal system of the organism. From a biomechanical point of view, those movements are built from opposing sets of skeletal muscles, termed flexor and

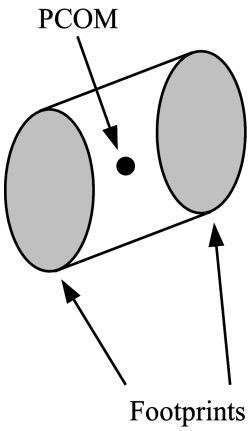

Figure 4.3: Illustration of the support area in the double-support phase. A static gait requires that the (vertical) **projection of the center of mass** (PCOM) of the robot should lie within the support area formed by the feet on the ground.

extensor muscles. When contracted, the **flexor** bends a joint, thus decreasing the angle between components of a limb. This action, such as bending the knee or elbow, is known as **flexion**. The opposite action, which opens a joint, is called **extension**, and it is accomplished by an **extensor** muscle. Hence, the movement of body structures is accomplished by alternating contraction of muscles. A more detailed biomechanical analysis of bipedal gaits can be found in e.g. [131].

4.1.2 Gait stability

A good bipedal gait is characterized by the attainment of a stable periodic motion of the bipedal system. In order to maintain stable locomotion, the joint variables (angular displacement and velocity), of the bipedal system must follow a steady cyclic trajectory [50]. From a dynamical systems point of view, a cyclic motion is characterized by the presence of a **limit cycle attractor** in the phase space of the bipedal system. Attractors are bounded subsets of the phase space, to which regions of initial conditions converge as time evolves. As an example, a limit cycle of the left knee joint variables of the simulated robot used in the experiments described in Paper VI is depicted in Figure 4.4. If the gait of the bipedal system is slightly perturbed, by e.g. some external force applied to it, some of the phase trajectories of the system can change their paths. However, a stable limit cycle will attract adjacent phase trajectories, and the system will return asymptotically to the original limit

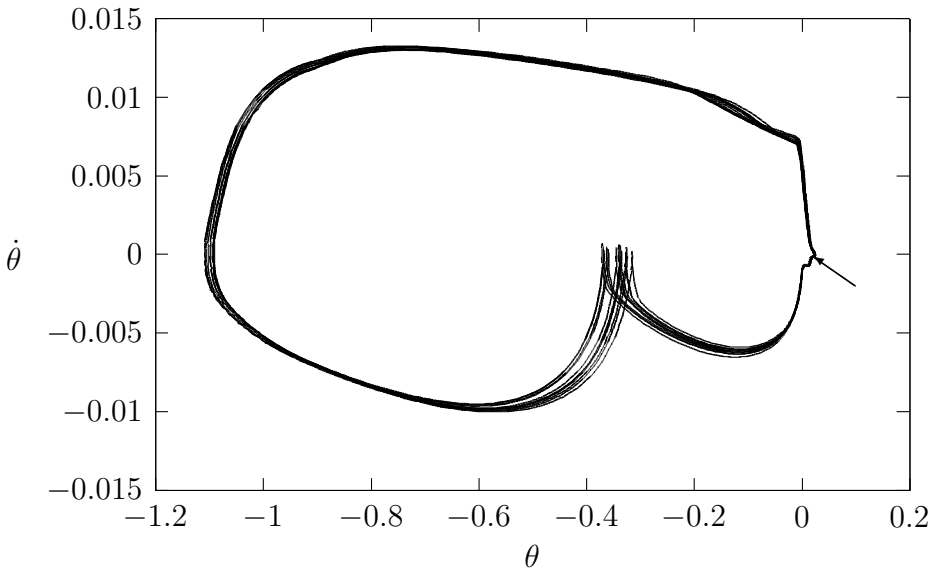

Figure 4.4: Phase portrait of a periodic gait. The figure shows the joint angular velocity $\dot{\theta}$, plotted against the angular displacement variable θ of the left knee of the simulated robot used in Paper VI. This figure corresponds to the angular motion of the knee joint, shown as $\theta_2(t)$ in Figure 5.7 below. One cycle in this figure corresponds to a complete stride of the robot. The starting point of the phase trajectory is indicated by the arrow in the figure, and the trajectory evolves in the clockwise direction. Here, the phase space plot is depicted for the interval $t \in [5, 11]$, where the gait has converged to a stable state.

cycle [1]. The **basin of attraction** of the limit cycle is the surrounding region in which this attracting feature is valid. Therefore, in order for a gait controller to be robust, its associated limit cycle should have a sufficiently wide basin of attraction. Since walking stability of a bipedal system is crucial to its performance, the concept of gait stability has been studied in terms of dynamical systems theory [30, 50, 61, 93, 102]. Furthermore, energy-optimal control laws for bipedal systems have been synthesized [9, 10], using optimal control theory and non-linear programming [11].

4.1.3 Gait transitions

In human walking, an increase in the locomotion speed is attained by a rise in both the frequency and amplitude of the movement. When a certain speed has been reached, the gait changes from walking to running. Within the framework of dynamical systems theory, a smooth transition of the gait

pattern can be seen as **bifurcation**[1] between different types of global limit cycles of the system [102, 119]. In this thesis, gait transitions of bipedal systems are discussed in Subsection 5.5.2.

4.2 Motor systems

In this section, the foundations of the systems generating the signals required for muscular control will be discussed.

4.2.1 Motor system hierarchy

In vertebrate animals the **central nervous system** (CNS) consists of the brain and the spinal cord. Generally, animal brains are structured into a number of different regions, each of which is devoted to specific functionality [71]. The **forebrain**, which is the largest part of the brain, controls higher level cognition, basic survival behaviors, homeostasis, and the display of emotions. The **brain stem**, which comprises the midbrain and the hindbrain, connects the brain to the spinal cord, allowing for communication between the brain and the rest of the body. The **midbrain** is responsible for processing sensory information, and it connects the forebrain to the brain stem. The **hindbrain** co-ordinates motor activity, posture and equilibrium, and it regulates unconscious but essential functions such as breathing. Finally, the **spinal cord** extends from the brain stem, and functions as a sensory and motor conduit between the brain and the body [2]. Given the (rather coarse-grained) outline of the brain structure above, one would perhaps assume that the structure of the brain is fixed once and for all. However, on a finer scale the specialization of the different regions of the brain is not completely hard-wired [71]. An injured brain can, to some extent, rewire itself in order to compensate for the damage.

In higher animals in general, the **motor system**, which is responsible for locomotion and other motor activities, is organized in a hierarchical fashion, as depicted in Figure 4.5. The same three key components are present in the nervous systems of different species, despite the variety of body forms and types of locomotion [111]. First, **central pattern generators** (CPGs) in the lower centers of the spinal cord contain neural circuits for generating coordinated rhythmic outputs of the motor neurons. CPGs are activated and controlled by descending fibers from the second key element, the higher motor centers in the cerebral cortex of the forebrain. Third, there are **feedback**

[1] In the study of dynamical systems, a bifurcation occurs when a change made to specific parameter values of a system causes a qualitative change in its dynamic behavior [98].

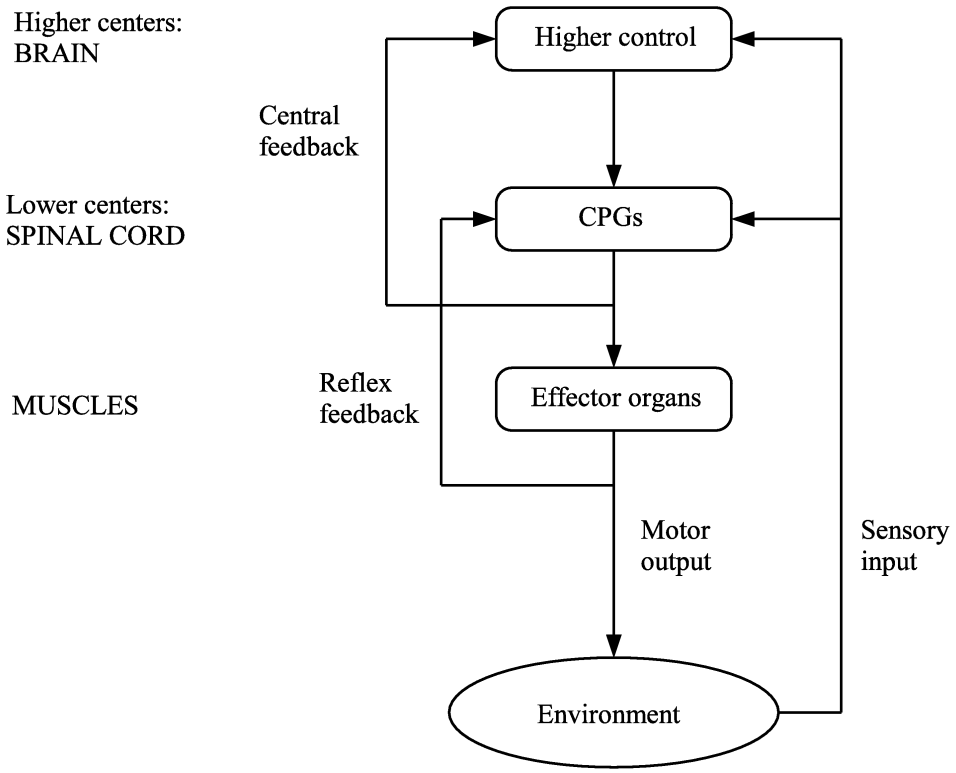

Figure 4.5: Schematic picture of the motor hierarchy in higher organisms. (After [111]).

circuits which return information from the muscles, and from the external environment. Furthermore, **central feedback** circuits pass on information within the nervous system itself, from lower to higher centers. Central feedback, which consists of a copy of the CPG output, is compared with sensory input by the higher centers of the brain, to determine how closely the generated CPG output followed the output needed to reach some behavioral goal, see Figure 4.5.

This hierarchical motor system organization allows for the generation and control of different kinds of movements of the musculo-skeletal system. The entirely automatic and involuntary discrete movements referred to as **reflexes** link stimulus to response without directly involving the higher centers of the brain. Stereotyped **rhythmic movements**, like walking and swimming, are typically voluntary only in the initiation and termination phases of the movements, but almost automatic in between. Unlike an involuntary action, a voluntary **goal-oriented action** is induced by a response to external stimuli, which have been processed by the brain. With this **hierarchical**

organization of the motor system, direct control of reflexes and other involuntary movements is delegated to the lower levels of the system, and the higher levels control voluntary actions without having to specify explicitly the details of the motor patterns.

As indicated above, CPGs are neural circuits capable of producing oscillatory output given tonic (i.e. non-oscillating) input. The aim of the work described in Paper VI has been to design artificial counterparts to biological CPGs, with associated feedback networks, for a three-dimensional simulated bipedal robot. The ultimate research goal is to implement a similar CPG structure on a real physical bipedal robot. The remainder of this chapter will treat the CPGs as part of the motor control system, and for now ignore the higher levels of the control system.

4.2.2 CPGs in biological organisms

There exists indeed biological evidence for the presence of central pattern generators in both lower and higher animals. The **lamprey**, which is one of the earliest and simplest vertebrate animals, swims by propagating an undulation along its body, produced by an alternating activation of motor neurons on the left and right sides of the segments along the body. The lamprey has a brain stem and spinal cord with all basic vertebrate features, but with orders of magnitude fewer nerve cells of each type than higher vertebrates. Therefore, it has served as a prototype vertebrate for the detailed analysis of the nervous system, including the lamprey CPG, in a number of neurophysiological studies [51, 52, 53, 54, 55]. Furthermore, a review work supporting the notion of biological CPGs in a variety of species, is given in [33]: In some early experiments by Brown [22, 23], it was shown that cats with transected spinal cord and with cut dorsal roots still showed rhythmic alternating contractions in ankle flexors and extensors. This was the basis of the concept of a spinal locomotor center, which Brown termed the **half-center model** [24]. The concept is further discussed in Subsection 4.2.3. Further evidence for a spinal cord CPG structure in vertebrates was obtained by experiments in which rhythmic output was generated, although movement-related afferent input[2] was completely eliminated by blocking of the movement. This can be achieved either by injection of neuro-muscular relaxants, or by transection of the efferent nerves. By recording the output of efferent nerves, rhythmic activity was demonstrated in both the hindlimbs and the forelimbs of cats. This phenomenon is referred to as **fictive locomotion**, and it closely re-

[2] An *afferent* nerve carries information towards the CNS, whereas an *efferent* nerve carries information away from the CNS.

sembles the locomotion pattern that can be recorded from the intact animal. When the movement is blocked, rhythmic sensory input is absent but static afferent information (e.g. hip position) is present and can influence the CPG. The occurrence of fictive locomotion is evidence that CPG structures in the isolated spinal cord are capable of generating rhythmic output in the absence of any signals from efferent descending sources as well as movement-related afferent sources [33]. It is pointed out in [33], however, that one cannot completely exclude the possibility that some of the locomotor output is not centrally generated, but could to some extent originate from stretch reflexes.

To this author's knowledge, there is only evidence by inference of the existence of a human CPG. The strongest evidence comes from studies of newborns, in whom descending supraspinal control is not yet fully developed, see e.g. [135] and references therein. Furthermore, advances made in the rehabilitation of patients with spinal cord lesions support the notion of a human CPG: Treadmill training is considered by many to rely on the adequate afferent activation of a human CPG [33]. In view of the results of the many extensive studies on the subject, it seems likely that primates in general, and humans in particular, would have a CPG-like structure.

In humans, however, CPGs play a less significant role for walking than in simpler species, such as those mentioned above. In addition to the generation of a coordinated, rhythmical activation pattern of the extensors and flexors of the legs, human bipedal gait requires more advanced control of posture and balance during walking, which is provided by sensory inputs [96]. CPGs alone are thus not sufficient in order to generate stable bipedal gaits that can handle large physical perturbations and environmental changes.

In the CPG study presented in this thesis, feedback regarding certain joint angles as well as foot-to-ground contact, was utilized to some extent (see Subsection II-D. in Paper VI). The primary aim of the study described in Paper VI was the generation of CPG networks capable of producing a stable gait for a 14 DOF simulated bipedal robot in a *static* environment. Thus, the feedback signals were considered less important in this study. However, a deeper investigation of how different kinds of feedback can influence the generated gaits, in the case of CPG-based bipedal gait control, would be an interesting topic for future study.

4.2.3 Biological CPG models

Based on biological findings, three main CPG models have been derived [111]. In order to account for the alternating activation of flexor and extensor muscles of the limbs of the cat during walking, the **half-center model** was suggested [24]. Each pool of motor neurons for flexor or extensor muscles is

activated by a corresponding half-center of **interneurons**[3]. Another set of neurons provides a steady excitatory drive to these interneurons. Further, inhibitory connections between each half-center of interneurons ensure that when one half-center is active, the other is being suppressed. It was hypothesized that, as activity in the first half-center progressed, a process of fatigue would build up in the inhibitory connections between the two half-centers, thereby switching activity from one half-center to the other [24].

The **closed-loop model** was originally proposed for the **salamander** [74]. To some extent it resembles the half-center model, but the interneurons are organized in a closed loop of inhibitory connections. There are corresponding pools of motor neurons activated, or inhibited, in sequence, allowing for a finer differentiation in the activation of the individual muscles.

In the **pacemaker model** rhythms result as a cell membrane property, involving complex ionic current interplay, of a group of pacemaker cells[4]. These cells undergo rhythmic excitation as a result of intrinsic membrane mechanisms, involving complex interaction of ionic currents. The pacemaker cells drive flexor motor neurons directly, and bring about concurrent inhibition of extensor motor neurons through inhibitory interneurons [111].

4.2.4 Computational CPG models

Several approaches for computational modeling of the characteristics of CPGs are known from the literature. In the pilot study of the work described in Paper VI, three commonly used neural models and a non-linear oscillator, suitable as candidates for developing a CPG network for bipedal walking, have been identified. In this subsection, those CPG models will be further discussed. The investigation process, in which each CPG model was made a part of a randomly generated network, will then be described in Section 5.4.

Drawing upon neurophysiological work on the lamprey spinal cord [54], numerical simulations of the local spinal network were performed by Grillner and co-workers, using model neurons ranging from biophysically realistic models describing the most important membrane currents and other mechanisms of importance [36, 129], to simple connectionist-type non-spiking neurons [34]. The network was based on an earlier, experimentally established connectivity of the lamprey spinal CPG [129]. The use of the biophysical models makes it possible to compare the simulation results directly with corresponding experimental data. The advantage of using the simpler model,

[3]Refers to neurons that send signals only to neurons and not to other body parts (such as muscles).

[4]The heart rate is controlled by electrical impulses, which are generated by pacemaker cells.

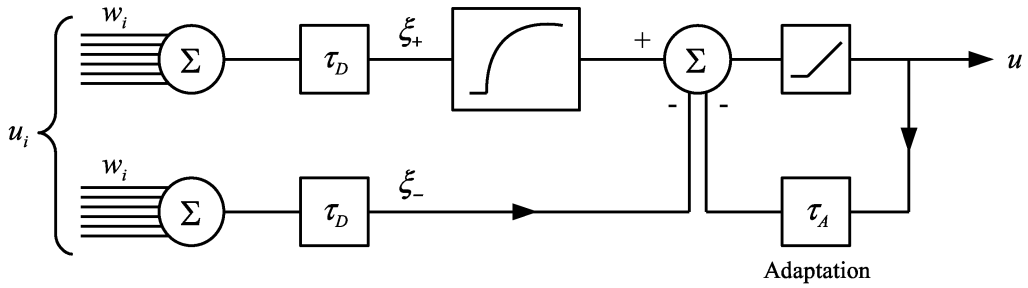

Figure 4.6: Schematic depiction of the mechanisms included in Ekeberg's neuron model. (After [34]).

on the other hand, is the weak dependence of certain parameters, which are hard to measure experimentally [35]. Ekeberg [34] showed that the simpler model could be used to produce swimming behavior for a lamprey in a two-dimensional hydromechanical simulation. An extended version of the **Ekeberg model** was later adopted for locomotion control of the salamander, in simulation [62]. In particular, the transition from aquatic to terrestrial locomotion, which has been an important step in vertebrate evolution, was studied. However, a different neuron model was used in that paper. In the pilot study of Paper VI, an implementation of the neuronal model as described in [34] was chosen for further investigation. In this model, the delayed values for the excitatory synaptic input, ξ_+, the inhibitory synaptic input, ξ_-, and the neuron output, ϑ, are calculated from first-order differential equations:

$$\dot{\xi}_+ = \frac{1}{\tau_D}\left(\sum_{i\in\Psi_+} w_i u_i - \xi_+\right) \tag{4.1}$$

$$\dot{\xi}_- = \frac{1}{\tau_D}\left(\sum_{i\in\Psi_-} w_i u_i - \xi_-\right) \tag{4.2}$$

$$\dot{\vartheta} = \frac{1}{\tau_A}(u - \vartheta) \tag{4.3}$$

In Equations (4.1) and (4.2), Ψ_+ and Ψ_- are the sets of incoming excitatory and inhibitory synapses, respectively, w_i is the strength of synapse i and u_i is the output value from the corresponding presynaptic neuron. Now, the

output from the model neuron is given by the following equation:

$$u = \begin{cases} 1 - \exp\left(\left(\Theta - \xi_+\right)\Gamma\right) - \xi_- - \mu\vartheta & \text{if positive} \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

In this equation, Θ is the threshold for activation, Γ is a gain constant, and μ controls the degree of adaptation, see Figure 4.6. The neural model acts primarily as a leaky integrator with saturating transfer function, having an output value between 0 and 1, as indicated by Equation (4.4). The excitatory and inhibitory synaptic inputs, ξ_+ and ξ_- respectively, are added in Equation (4.4) and both are subject to a dendritic delay[5] with a time constant τ_D, see Equations (4.1) and (4.2). Spike frequency adaptation is also included as a delayed negative feedback with another time constant τ_A, see Equation (4.3).

The next candidate model (in the pilot study) for CPG development was based on a **continuous-time recurrent neural network** (CTRNN) neuronal model, as described in [26]. CTRNNs are networks formed by neurons of the following general form:

$$\tau_i \dot{y}_i = -y_i + \sigma\left(\sum_{j=1}^{N} w_{ij} y_j + \theta_i\right), \quad i = 1, \ldots, N \tag{4.5}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.6}$$

where N is the number of neurons in the network, y_i is the state of neuron i, $\dot{y}_i$ denotes the time rate of change of this state, τ_i is the neuron's membrane time constant, w_{ij} is the strength of the connection from neuron j to neuron i, and θ_i is a bias term. In a neurobiological interpretation, the state y can be interpreted as the nerve cell's mean membrane potential, while its output $\sigma(x)$ is associated with the short-term average firing frequency [4]. CTRNNs have been analyzed as neuronal models for CPGs [4] and employed as CPGs for a simple locomotion task in [6], where it was demonstrated that even small neural circuits (containing one or two neurons) are capable of a variety of dynamical behaviors. Furthermore, CTRNNs have been employed as neurobiological models of leeches [79] and of the nematode *C. elegans* [39].

The most commonly used neuronal CPG model is the one formulated by Matsuoka [83], which is a mathematical description of the half-center model. In its simplest form, an oscillator unit consists of two neurons arranged in mutual inhibition, as depicted in Figure 1 in Paper VI. The neurons in the

[5]Dendrites are short branches connecting neurons to each other.

half-center model are described by the following differential equations [119]:

$$\tau_u \dot{u}_i = -u_i - \beta v_i + \sum_{j=1}^{N} w_{ij} y_j + u_0 \quad (4.7)$$

$$\tau_v \dot{v}_i = -v_i + y_i \quad (4.8)$$

$$y_i = \max(0, u_i) \quad (4.9)$$

where u_i is the inner state of neuron i, v_i is an auxiliary variable measuring the degree of self-inhibition (modulated by the parameter β) of neuron i, τ_u and τ_v are time constants, u_0 is an external tonic (non-oscillating) input, w_{ij} are the weights connecting neuron j to neuron i, and, finally, y_i is the output of neuron i. Two such neurons arranged in a network of mutual inhibition (a half-center model), form an oscillator, in which the amplitude of the oscillation is proportional to the tonic input u_0. The oscillator is driven by this input, referred to as the **bias**, making the CPG unit produce an oscillatory output signal, whose amplitude is proportional to the strength of the bias signal. The frequency of the oscillator can be controlled by changing the values of the two time constants τ_u and τ_v. This is illustrated in Figure 4.7. If an external oscillatory input is applied to the input of an Matsuoka oscillator, the CPG can lock onto its frequency. Then, when the external input is removed, the CPG smoothly returns to its original oscillation frequency. This property, referred to as entrainment, is relevant for the usefulness of the Matsuoka oscillator for adaptive locomotion [119]. Furthermore, the **Matsuoka oscillator** model has proved to be very useful in various motor control tasks. It has been employed for bipedal locomotion in 2D [37, 119] and 3D [66], quadruped locomotion [46], and rhythmic arm movements [132]. For reasons that will be clarified in Section 5.4, the Matsuoka oscillator model was adopted for the development of CPG networks for bipedal walking in Paper VI.

In contrast to the three CPG approaches mentioned above, there are also approaches (referred to as **nonlinear oscillators** [28]) that do not attempt to model real, biological neurons. Instead, using nonlinear oscillators can be thought of as a purely mathematical approach to model the signals required to produce locomotion behaviors in a robot. A nonlinear oscillator CPG unit can be represented by two first order differential equations, as described e.g. in [28]. Based on the equations of a harmonic oscillator, where $\tau\dot{v} = -x$, and $\tau\dot{x} = v$, the proposed model consists of several oscillators

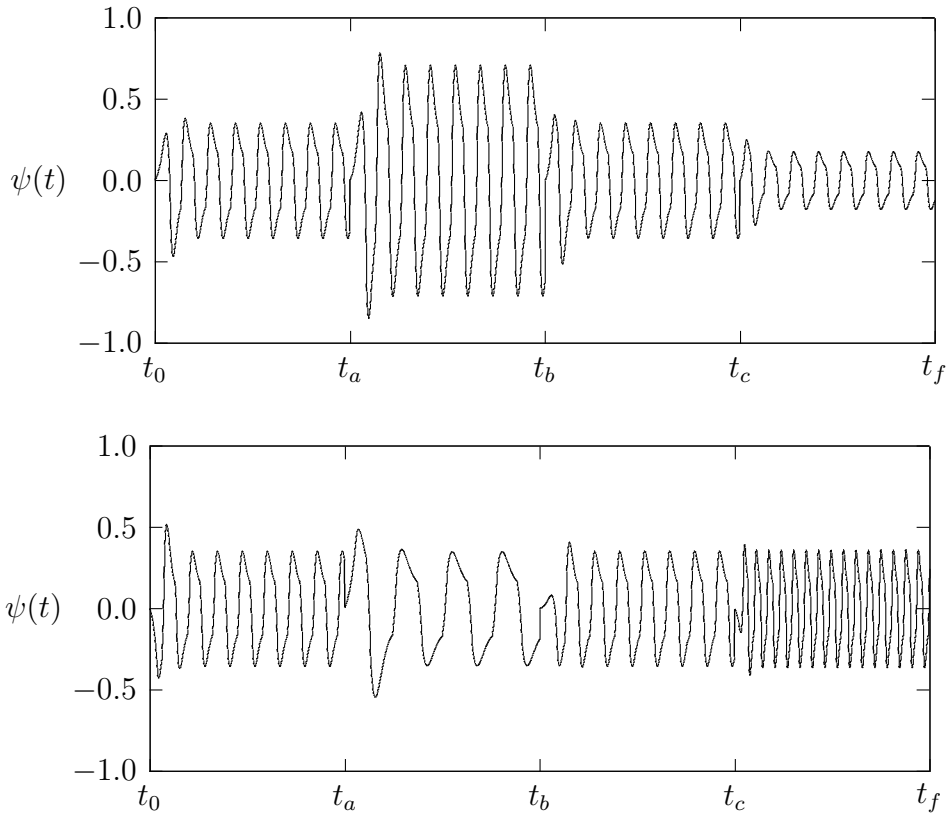

Figure 4.7: Changing the behavior of the Matsuoka oscillator output signal $\psi(t)$. *Top panel:* By varying the bias u_0, the amplitude is changed. At time t_a the bias was changed from its original value u_0 to $2u_0$, at time t_b it was changed back to u_0 again, and at time t_c the bias value was set to $\frac{u_0}{2}$. *Bottom panel:* Changing the frequency by varying the time constants τ_u and τ_v. At time t_a the time constants were changed from their original values τ_i to $2\tau_i$. At time t_b the time constants were changed back to their original values, and at time t_c their values were set to $\frac{\tau_i}{2}$.

linked together, as described by the following equations:

$$\tau \dot{v}_i = -\alpha \frac{x_i^2 + v_i^2 - E_i}{E_i} - x_i + \sum_j (a_{ij} x_j + b_{ij} v_j) \tag{4.10}$$

$$\tau \dot{x}_i = v_i \tag{4.11}$$

The energy parameter, E_i, controls the amplitude of each oscillator, which is given by $x_i^2 + v_i^2$, where the first term approximates the oscillator's potential energy, and the second term its kinetic energy. The oscillator has a sinusoidal

limit cycle and the speed by which this limit cycle is reached is controlled by the parameter α. The frequency of the oscillation is determined by the time parameter τ, and the coupling constants a_{ij} and b_{ij} are used when several oscillators are connected to form a network. The output of oscillator i is given by x_i. This nonlinear oscillator CPG model was originally proposed in [63], where it was used to model salamander locomotion in simulation. It has also been used to model snakelike and lampreylike locomotion in a real robot [28], and locomotion in a modular robot [15]. Furthermore, it was utilized for bipedal locomotion in simulation [91], but with less successful results. For the pilot study for Paper VI, the nonlinear oscillator model as described in [28] was adopted.

4.3 Bipedal walking in robots

In this section, the different views of the implementation of bipedal gaits in robots will be discussed briefly. In a conventional, engineering approach, controllers based on classical control theory are used, but a shift towards biologically inspired methods is growing rapidly in the field of robotics research [72].

4.3.1 Static and dynamic gaits

Bipedal gaits for robots are usually divided into two main classes: **static gaits** and **dynamic gaits**, which are described in Section 1 in Paper III. As pointed out there, the most fundamental issue when implementing bipedal locomotion in a robot is how to preserve the balance of the bipedal system during walking. In static walking, walking balance is achieved if the projection of the robot's centre-of-mass (PCOM) on the ground plane remains within the convex hull of the support polygon at all times. In dynamic walking one must also account for dynamical effects, which arise due to the velocities and accelerations of the components of the bipedal system. In order to address this issue, certain stability criteria have been proposed, such as the **zero moment point** (ZMP) [124, 125, 126]. The ZMP is defined as the contact point between the ground and the foot sole of the supporting leg where the torques around the horizontal axes, generated by all forces acting on the robot, are equal to zero. During a dynamically balanced gait, the ZMP can only move within the supporting area. More recently, other related concepts have been introduced, such as the **foot rotation indicator** (FRI) point [48], and the **zero rate of change of angular momentum** (ZRAM) [49], as described in Paper III. Dynamic walking is usually more

natural-looking and energy-efficient, and provides for higher walking speed than static walking, but the control problem of dynamic walking is also much harder.

4.3.2 Trajectory tracking gait synthesis

Conventionally, there are two main approaches to bipedal gait synthesis: off-line trajectory generation, and on-line motion planning. Both these methods rely on the calculation of reference trajectories, such as e.g. trajectories of joint angles, for the robot to follow. An **off-line controller** assumes that there exists an adequate dynamic model of the robot and its environment, which can be used to derive a body motion that adheres to a stability criterion, such as e.g. the ZMP. Only if the ZMP stays within the allowable region, can the generated gait be implemented on the actual robot. An **on-line controller**, on the other hand, uses limited knowledge of the kinematics and dynamics of the robot and its environment. Instead, simplified models are used to describe the relationship between input and output. This method relies much on feedback information. Compared to off-line controllers, on-line controllers generally achieve higher walking robustness at the cost of an increase in demand for computational resources, since on-line controllers solve the system equations during operation. For further details, see Section 1 in Paper III, and references therein.

A drawback of control policies based on classical control theory, such as the ones outlined above, is that when a robot is moving in a realistic, dynamically changing environment, the physical conditions will never be possible to predict completely. Hence, reference trajectories can rarely be specified in such environments. A control policy based on conventional control theory will lead to lack of flexibility in an unpredictable environment [118]. Therefore, alternative **biologically inspired computational methods**, which in general do not require any reference trajectory when generating bipedal gaits, have been proposed [5, 8, 72, 105].

4.3.3 Biologically inspired gait synthesis

A common approach in biologically inspired control of walking robots is based on connectionist theory, i.e. **artificial neural networks** (ANNs). A review of such methods can be found in [72]. Another approach follows the paradigm of artificial evolution, i.e. the use of **evolutionary algorithms**, to optimize a given controller, which may consist of **recurrent neural networks** (RNNs)

[106], **finite state machines** (FSMs) [101], or any other control structure of sufficient degree of flexibility [14]. The controller may also consist of a structure coded by hand, as in Paper I. A third approach uses biologically inspired computational methods, such as GP, to generate certain structures, or programs, for locomotion control of robots see [137] and Papers II and III. In some cases the evolutionary optimization (or generation) of program structures may be applied to a certain component of the overall controller, as e.g. in [97], where a feedback network was generated using GP. However, to the author's knowledge, there exist only a few examples, such as those mentioned in Section 1 of Paper III, which go beyond parametric optimization and generate also the structure of the bipedal walking controller. It should also be noted that in the work described in this thesis, in particular in Papers II, III, VI, and VII, contributions are made to this specific subfield of robotics research. In Papers II and III, **linear genetic programming** (LGP) was used to generate locomotion controllers from first principles for simulated bipedal robots, and in Paper VI both the network structure and the parameters of a CPG network were evolved, using a GA as the optimization method. Finally, in Paper VII, both the structure and the parameters of the gait control program were modified by evolution.

When starting the work described in Papers II and III, it was hypothesized that GP would provide a good basis for automatic creation of locomotion controllers for bipedal robots. The outcome of these experiments is further discussed in Section 5.2 of this thesis.

An alternative approach to gait synthesis was investigated in Paper VI. As discussed previously in this chapter there is a strong motivation, based on biological findings in animals, for adopting CPGs as the foundation of a controller for bipedal walking. CPGs can easily generate the rhythmic output patterns required for bipedal locomotion, and they provide an appropriate structure for both parametric and structural evolution. Moreover, CPGs exhibit certain properties of adaptation to the environment: Both the nervous system, composed of coupled neural oscillators, and the musculo-skeletal system have their own nonlinear oscillatory dynamics. The latter is caused by the effects of gravity and inertia. Furthermore, it has been demonstrated that, during locomotion, some recursive dynamics occurs between the neural control system and the musculo-skeletal system. This phenomenon, termed **mutual entrainment**, emerges spontaneously from the cooperation among the system's components in a self-organized way [119]. That is, natural periodic motion, set close to the natural (resonant) frequency of the mechanical body, is achieved by the entrainment of the CPG to a mechanical resonance. In particular, the CPGs are entrained to a mechanical resonance by sensory feedback. The feedback is non-essential for the rhythmic pattern generation

itself, but rather modifies the oscillations in order to achieve adaptation to environmental changes [67]. It has been hypothesized that the principle of mutual entrainment through sensory feedback is responsible for the global stability of the movement in real time [118, 119]. In such cases, the neural, musculo-skeletal, and sensory systems behave cooperatively to adapt to unpredictable changes of the environment, as demonstrated in the case of obstacle avoidance [116].

Chapter 5

Applications

This chapter aims at summarizing the papers included in this thesis and emphasizing the results achieved. In Section 5.1 experiments are described, in which it was demonstrated that online evolutionary optimization of bipedal gaits in real, physical robots (see Papers I and VII) is indeed achievable. Furthermore, in Section 5.2, an evolution-based approach for automated generation of gait controller programs for simulated bipedal robots is described (see also Papers II and III). Next, in Section 5.3, two case studies of the properties of the UF method for behavior selection are presented (see also Papers IV and V). Finally, Sections 5.4 and 5.5 describe a simulation-based case study in which it was shown that the process of artificial evolution (e.g. an EA) can be used to concurrently design a CPG network structure *and* also optimize its parameters, in order to produce efficient bipedal walking behavior for a robot (see Paper VI).

5.1 Evolution in physical robots

The main purpose of the experiments regarding online evolution of gaits, as described in Papers I and VII, was to investigate whether artificial evolution is a useful approach to the problem of bipedal gait optimization in physical robots, despite the fact that evolution carried out directly in hardware imposes some restrictions on the setup of the EA; that is, the evaluation time of the individuals, the total number of generations, and the population size are very limited, as compared to the case in where evolution is carried out in a simulator. Two case studies of online optimization of hand-coded gaits for physical bipedal robots have been carried out (see Papers I and VII).

In the first case, described in Paper I, the robot used in the experiment

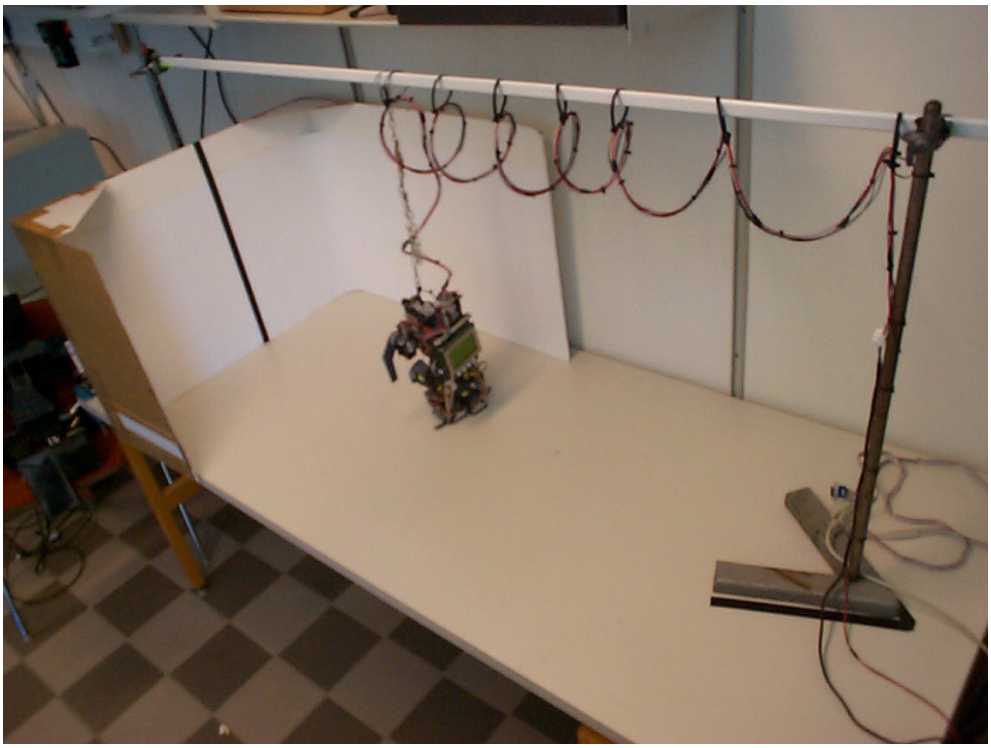

Figure 5.1: Experimental setup of the first evolutionary gait experiment carried out in a bipedal robot (Paper I). The cables suspended in the horizontal beam above the robot were used for external power supply during the experiment.

was a simplified, scaled model of a full-size bipedal robot, having 12 DOFs, and onboard power supply and controller. Intended for research applications, it was developed by Chalmers University of Technology in co-operation with Dortmund University in Germany [136]. For sensing, the robot has a vision system and a range sensor. The actuators of the robot are standard off-the-shelf RC servo motors[1], which are commanded to a specific angular position by pulse-code modulated input signals sent from the microcontroller. Each servo is assigned a target position by addressing it an integer value within the interval $I \in [0, 255]$, from the microcontroller. Further details regarding the hardware implementation can be obtained from Paper I and [136].

Initially, a hand-coded gait program was designed for this robot. This program consists of a number of vectors $\mathbf{S}_i$, such that each vector describes

[1]Two different sizes of servo motors were used. The four motors for the ankle and hip joints were of a strong type with an output torque of 0.87 Nm, and for the other eight joints, servo motors with an output torque of 0.38 Nm were used.

an **internal state** of the robot, i.e. a set of servo target positions, or joint angles. The robot can thus be commanded to a certain position (in joint space) by addressing the servos the values of such a **state vector**. The controller is designed such that when two different state vectors, $\mathbf{S}_a$ and $\mathbf{S}_b$, are sent serially to the robot, it will make a continuous motion from state a to state b. Furthermore, by adding more vectors, the robot can be made to move continuously over an arbitrary number of states, to some final state n.

The hand-coded gait program was developed by manually specifying a set of state vectors $\mathbf{S}_i$, such that a complete stride of the robot is described by the set of state vectors. By repeatedly sending the set of state vectors to the robot, an arbitrary number of strides is executed by the robot. However, manually specifying a gait in this manner is both difficult and time consuming, and there is no guarantee that the obtained gait is optimal in any respect. An approach where the hand-coded gait was optimized by means of an EA, was therefore adopted.

In the evolutionary gait experiment described in Paper I, an EA ran on the robot's onboard controller. During the experiment, the robot was contained in an arena shown in Figure 5.1. The EA operated on the (integer) values of the gait control program (i.e. the set of state vectors defining a stride), in order to optimize the gait of the robot with respect to locomotion speed and stability. Note that the structure defining a gait program was fixed during the evolution. A fitness function was specified, taking into account both the walking speed and the straightness of the locomotion path; see Section 4.2 of Paper I. For an evaluated individual, the fitness value was based on measurements obtained from the robot's onboard sensors only. This fitness function takes into account the objectives for the EA, mentioned above. However, one could in principle have included other quantities in the fitness function as well, such as e.g. a subjective judgement of the quality of the gait, given by the experimenter.

Since the evolution took place in a real, physical robot, only a relatively small population size could be managed; thus a population of thirty individuals was used. Furthermore, due to wear of the servos of the robot (several servos were replaced up to three times) during the evolutionary experiment, the EA could only be run for a small number of generations; in this particular experiment, nine generations[2] were executed. Figure 5.2 shows how the EA modified the phase space trajectories of certain joints. However, the EA evolved an individual that outperformed the manually developed

[2] *Generations* in this context refers **generation equivalents**. In a steady state EA, a generation equivalent has been completed when N new individuals have been created, where N is the population size.

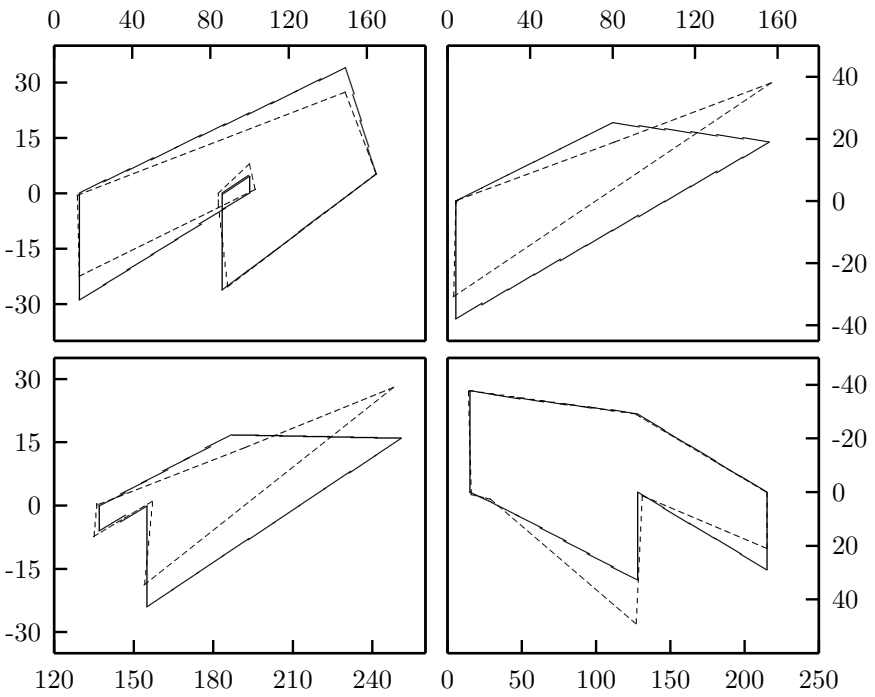

Figure 5.2: Phase portraits of the hand-coded gait (solid lines), and of the evolved gait (dashed lines), respectively. In the figure the joints' angular velocity $\dot{\theta}$ (vertical axis) is plotted against the angular displacement $\theta(t)$ for four of the joints, namely; right knee (*top right panel*), right ankle (*top left panel*), waist (*bottom right panel*), and left ankle (*bottom left panel*). These trajectories are not based on actual joint angles measured on the robot. Instead, they were generated directly from the set of state vectors defining a stride of the robot.

gait program. The best evoved individual received a fitness value of 0.1707. The hand-coded individual was tested in the same way, and received a fitness value, averaged over three trials, of 0.1051. The increase in fitness was mostly due to a straighter path of locomotion: When tested, the hand-coded individual had a sidewise deviation of 0.33 m over a 1.0 m distance, from the desired straight path of locomotion. The corresponding value for the best evolved individual was less than 0.05 m. The straightness of the robot's locomotion path was thus strongly improved.

In the second case study involving a physical robot, a small, commercially available humanoid robot [75], with 17 DOFs, was used (see Paper VII and Figure 5.3). As in the case described above, a hand-coded gait (termed the *standard gait*), consisting of a sequence of set angles (states) for the servo

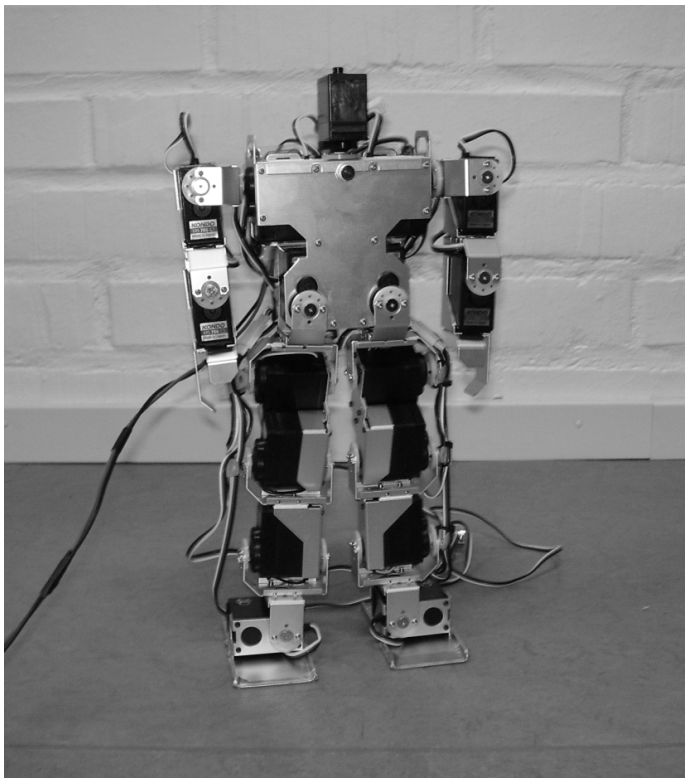

Figure 5.3: The bipedal robot used in connection with Paper VII.

motors, and ramping times for the transition between states, was subject to an evolutionary optimization process. Note, however, that in this study *structural modifications* were allowed, as well as *parametric optimization*. By contrast, in Paper I only parametric optimization was enabled. In Paper VII, the parametric mutations modified the actual value of a given gene, and the structural mutations inserted a new state between two consecutive states in an individual (see Section II-D in Paper VII).

The standard gait was used as starting point for the optimization: An initial population of 30 individuals was generated, using the standard gait as a seed. That is, the genomes of all individuals were first set so as to encode the standard gait. Next, the individuals were mutated, using the same procedure that normally takes place after the evaluation of a generation, i.e. when creating the next generation. The resulting set of individuals then formed the initial population. Each new generation of individuals was evaluated using the actual humanoid robot, which was contained in an arena, as shown in Figure 3 in Paper VII. The fitness function was set proportional to the

inverse of the time T taken for the robot to traverse the arena, i.e. fitness was proportional to the locomotion speed (see Equation (4) in Paper VII). Furthermore, the fitness values were normalized in such a way that fitness for the standard gait individual equalled 1.0, on average. By contrast with the former case, the robot was not equipped with any sensors. Thus, measurements for obtaining fitness values had to be performed using an external device, namely a photocell, which registred the passage of the finish line (see Figure 3 in paper VII).

In case of failure, i.e. if the robot fell over, or deviated too much from the intended path such that it would not pass the finish line at all, the evaluation was manually interrupted. In that case, individuals that fell over were given a fitness of 0, whereas individuals that walked in a strongly curved path (such that they would not break the light beam at the finish line), were somewhat arbitrarily given fitness 0.1. It should be noted that individuals that reached the finish line obtained fitness values much higher than 0.1. Using the method described above for assigning fitness values, the locomotion speed combined with a strong punishment for instability has been taken as the measure of quality of a gait. Thus, gaits that are stable and fast, and that make the robot walk along a straight-line path have been favored. The outcome of this experiment showed that an improvement in walking speed of approximately 65 % could be achieved. The sideways deviation over the evaluation distance L for the best-evolved individual was found to be 0.09 m, on average, which was roughly the same as for the standard gait.

In both case studies described in this section, the resulting robot gaits were improved unambiguously in a measurable and numerically quantifiable way. However, the major conclusions from these two experiments go beyond these detailed performance improvements: It has been shown that it is indeed possible to obtain significant improvements of functioning but non-optimized motor behaviors (gaits in these two cases), using an EA with a real, physical bipedal robot. Furthermore, as noted in Paper VII, the sideways deviation of the best gait always occurred in the same direction, as a result of individual variations of the particular components and their placement used in that robot. It is unlikely that such an effect would have been correctly modelled, at least without extensive and time-consuming system identification, in a simulation. Thus it may be concluded that, in cases where a generated gait is supposed to be applied in a specific physical robot, evolving in a physical robot is to be preferred to evolution in simulation, despite the great difficulties involved. However, as will be illustrated below, simulations may still play an important role, for example in investigations where the aim is to investigate a particular optimization method.

5.2 LGP for bipedal gait generation

The main purpose of the studies presented in Papers II and III was to investigate the application of an evolutionary method, based on LGP (see Subsection 3.2.3), for generation of human-like bipedal gaits, from first principles, for simulated bipedal systems with a high number of DOFs. Furthermore, it should be noted that (1) in each of these two case studies, the brain[3] of the robot was evolved starting from programs consisting of random sequences of basic instructions and (2) these studies are examples of totally model-free evolution, i.e. there was no model of the bipedal system, neither any *a priori* knowledge on how to walk, available to the evolved controllers.

In these experiments, two different bipedal models were used in simulation, namely (1) a bipedal model with 12 DOFs resembling an actual robot, see Figure 2 in Paper II, and (2) a generic bipedal model with 26 DOFs, see Figures 1 and 2 in Paper III. In Subsection 2.1.3 in Paper III the details regarding how the GP individual controls the robot are described in detail.

In Paper II, the controller was of the open-loop type, i.e. there was no feedback from the environment available to the control program. The only information available to the controller was the current internal state of the robot, i.e. the set of current joint angles (which were actually computed by the controller in the previous time step). In the other case (see Paper III), the simulated robot was equipped with a number of sensors as well: The robot had a three-axis **accelerometer** in its head, measuring linear accelerations, and one accelerometer in each foot. Furthermore, there were three **gyroscopes** in its head, measuring angular accelerations.

In the case of Paper II, the fitness function was specified such that it took into account the covered distance during the evaluation, and the height of the robot's head during walking, see Section 3.4 in that paper. In Paper III, a fitness function taking into account the covered distance during evaluation was specified. Moreover, a number of additional constraints, such as e.g. an energy discharging function, were investigated as well, see Sections 3.1 and 3.3 in Paper III. The addition of the energy discharging function was motivated by the fact that human bipedal locomotion is very energy-efficient, compared to the gaits of most bipedal robots (see Subsection 3.3.1 above) and, indeed, this addition made it possible to improve the results.

In both these cases, i.e. in Papers II and III, it proved to be possible to evolve programs that produce bipedal walking for the simulated robots. Inspection of the best individuals, after the evolutionary runs were terminated, showed that several of the best evolved individuals of Paper III are capable of

[3]The brain, in this context, refers to a gait-generating controller program.

generating bipedal locomotion several times longer than the evaluation time of 36 s. One of the best individuals was actually able to produce forward bipedal locomotion for more than 20 minutes.

The major conclusion that can be drawn from these two simulation-based case studies is, however, that the generation of such a complex behavior as bipedal walking for a system with a high number of DOFs, starting with a brain consisting of random sequences of basic instructions, is indeed realizible using artificial evolution, although fully anthropomorphic gait was never obtained in this case.

A possible explanation for the somewhat poor results with respect to gait quality obtained in Papers II and III, compared to the results obtained in Paper VI (see Section 5.5), could be the choice of sensor feedback: As mentioned above, in the first instance of LGP for gait generation (see Paper II) the controller program was of the open-loop type, i.e. no external sensor input was used at all. In the second case (see Paper III) the feedback used consisted of an artificial balancing sense, along with measurements of the robot's current joint angles. By contrast, in the case of CPG networks for bipedal gait generation (see Paper VI) the simulated robot was equipped with a touch sensor in each foot, which indicated whether there was ground contact or not. Furthermore, feedback measuring the waist angle, thigh angle, and leg angle, all relative to the vertical axis, was introduced. However, no balancing sense, as in Paper III, was used in the case of CPGs. Apparently, the touch feedback turned out to be more important (and perhaps also easier for the controller to interpret and use) for the generation of stable gaits, than the balancing sense. Furthermore, it is known that tactile feedback from the foot, indicating foot-to-ground contact, is essential for human locomotion [123]. Thus, the introduction of the above-mentioned touch sensors on the feet, in the case of the CPGs (Paper VI), is well-motivated.

No touch sensor was used in connection with the LGP-based controller (see Papers II and III). Including such a sensor might be an interesting topic for a future study of the LGP approach. It should also be noted that the starting point of evolution was different in the two cases (Papers II and III vs. Paper VI): When using CPGs, the type of controlling mechanism for each joint was decided upon prior to evolution. Each joint was assigned a Matsuoka half-center oscillator that automatically produces useful, oscillatory output signals (see Paper VI). This is a very different starting point compared to the random sequences of instructions used in the LGP case.

5.3 Evolution of utility functions

In Papers IV and V a method for behavior selection in autonomous robots was investigated. In Paper IV, the UF method was applied in a case study of a simple simulated guard robot, which should patrol an arena, see Figure 5.4. The main purpose of this simulation study, apart from demonstrating the feasibility of the method in a realistic robot task, was to study how the choice of polynomial degree of the utility functions influenced the performance of the UF method. Furthermore, the performance for various utility function specifications was investigated (see Paper V).

In Paper IV, the robot was differentially steered, with two DC motors. The arena contained several obstacles, as well as battery charging stations. The robotic brain had five behaviors, accessible for activation by the behavior selection system. Those behaviors were **straight-line navigation**, **obstacle avoidance**, **energy maintenance**, **corner seeking**, and **battery charging**. According to the concept of the UF method, each behavior is associated with a utility function, see Subsection 2.3.1. For each time step, it is determined which of the utility functions takes the highest value. Accordingly, that behavior is activated. The task of the robot was to cover as much of the arena as possible, thus a suitable fitness measure is the time spent in **straight-line navigation** behavior. In this behavior, the motor torques were set to equal, positive values.

In the simulation experiment, each individual was allowed a maximum simulation time of 100 s. However, the evaluation of a simulated robot was terminated directly in case of collisions with obstacles, or if the on-board battery became fully discharged. Four different polynomial degrees were investigated, namely 1, 2, 3, and 4, for which the number of polynomial coefficients equals 18, 44, 89, and 160, respectively, see Equation (2) in Paper IV. In each run 10,000 individuals were normally evaluated, and it was found that the best fitness values varied very little between the different polynomial degrees. However, a series of extended runs were carried out as well, where the number of evaluated individuals was on the order of 50,000 to 100,000. Then, the best fitness values varied significantly between the different polynomial degrees, in favor of the highest polynomial degree, see Section 6 of Paper IV. The conclusion is that, at least for the problem considered in Paper IV, better solutions could be found using a higher (i.e. 3 or 4) polynomial degree; on the other hand, the increase in the size of the search space for the larger polynomial degrees makes it difficult to find such solutions.

In Paper V, the performance of different EAs in connection with the UF method was investigated. In this simulation experiment, the robot used was similar to the one used in Paper IV, but the arena was slightly mod-

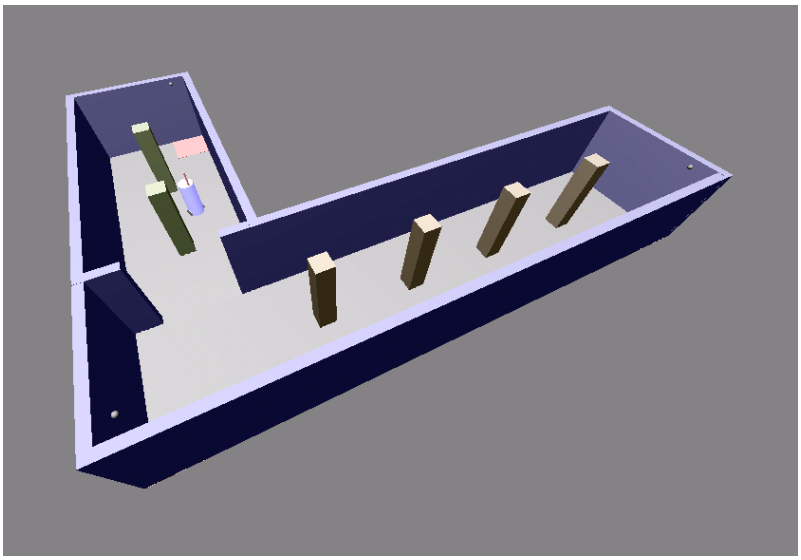

Figure 5.4: The arena patrolled by the guard robot in Paper IV.

ified to resemble a small apartment with three rooms. In this case study, the robotic brain had three behaviors, namely **straight-line navigation**, **obstacle avoidance**, and **battery charging**. The task of the robot was to explore the environment; thus the time spent in **straight-line navigation** was chosen as the fitness measure, as in the case above (Paper IV). For each behavior, a corresponding utility function was defined, see Section III of Paper V. Normally, each utility function is represented by a complete polynomial, including all terms up to and including a pre-specified degree. In that paper, however, the standard UF method was compared with a different approach where non-complete polynomials were used; for the latter, structural modifications (i.e. modifications of the number of polynomial terms) were allowed. The results of the simulation experiments described in Paper V suggest that the performance of the **standard UF** method is slightly better than that of the modified methods (in which the chromosomes are allowed to vary in size), see Section V of that paper. This was a somewhat surprising result, since it could be assumed that the optimization of non-complete polynomials would be faster than the optimization of complete polynomials, given the smaller number of coefficients in the former. A possible explanation, based on a simple toy problem, is given at the end of Paper V. The example shows that, in certain cases, an increase in the dimensionality of the search space (i.e. the number of coefficients that must be determined), may in fact *increase* the likelihood of finding the optimal solution.

5.4 Evaluation of CPG models

The primary goal of the work described in Paper VI was to demonstrate that the process of artificial evolution can be used to concurrently design a CPG network structure *and* to optimize its parameters, in order to produce efficient bipedal walking behavior for a robot in a fully three-dimensional simulated environment. In order to reach the goal, it is necessary to have a suitable CPG unit as a basis of the work. The initial task of this research project was therefore to identify such a unit, which could serve as the basis for the evolutionary development of a locomotion controller. There are several requirements that the CPG units forming the network should meet: In a population of randomly assembled CPG networks with arbitrary interconnections, a large portion of the randomly generated networks should relatively easily produce oscillations. In other words, the process of evolutionary optimization of the networks must not destroy their properties, but instead increase their chances of producing useful oscillations. Furthermore, since the purpose is to create bipedal locomotion controllers, the generated networks should be capable of producing output signals that closely resemble those alternating activation signals necessary to generate bipedal gaits. Finally, in full structual evolution, the size of the CPG networks should be allowed to vary. In order to meet these demands, the four CPG models described in Subsection 4.2.4 were evaluated. In the two following subsections, the evaluation procedure will be described.

5.4.1 Oscillations in random networks

When generating a CPG network for locomotion by means of evolution, a desirable property of the CPGs, as mentioned above, is that the likelihood of producing oscillations must be high, given random initial parameters. In order to investigate this property of the four different CPG models, 1,000 randomly generated oscillators of each type were created. An oscillator network was encoded in a genome consisting of five chromosomes, where each chromosome encoded a single neuron. The genes of each chromosome then specified each parameter of the neuron, i.e. whether the neuron should be active or not, whether it was an internal neuron or an output neuron, the sign of the output, which weights were present, and the values of those weights. The networks were evaluated by measuring their respective output signals. If the output signal amplitude of a certain network exceeded a value of 0.1, defined as peak-to-peak value, it was considered as oscillating.

With randomly initiated parameters, the nonlinear oscillator demonstrated the highest likelihood to produce oscillations. More than 90 % of the ran-

domly generated nonlinear oscillator networks were able to produce oscillations. The second best model was the Matsuoka oscillator, which produced oscillations with a 30 % likelihood. Both the CTRNN model and the Ekeberg model showed less than 1 % oscillating networks. Results similar to these were also confirmed in [64] in the case of Ekeberg's model, and in [82] for the CTRNNs. In the case of CTRNNs, it was suggested (see [82]) that the initial population should be seeded with oscillating networks initially, which seems reasonable, but in the investigation presented here, the main interest was to examine the intrinsic propensity for oscillations in randomly created networks, rather than gaining high performance in the initial stage. Therefore, the seeding approach was not tested.

5.4.2 Evolution towards a target signal

Bipedal gait control using CPGs, as described in Paper VI and elsewhere in this thesis, is possible in (at least) two different ways: The CPGs can be used to generate torques, which will then be directly applied to the robot's joints. Alternatively, the CPGs could be used for producing sequences of desired joint angles for the robot to follow. In the latter approach, however, it would be necessary to utilize some local control policy at each joint, e.g. a PD controller, in order to produce the torques for the motors, from the given joint angles. In order to investigate the ability of the CPG models presented earlier to produce such desired joint angles, and not just arbitrary oscillations, CPG networks were evolved towards predefined target signals. At the time of the experiments, no locomotion data from the simulated robot existed, therefore data from real human locomotion were used [110]. The data consisted of measured joint angles from the hip, knee, and foot joints, acquired during normal human walking. In addition, two artificially generated test signals, similar to a saw tooth wave and a spiking wave, respectively, were used as target signals as well.

Using the same encoding as described in the above subsection, a standard GA was utilized to evolve networks of the different CPGs towards the given target signals. Full structural evolution was enabled, with a maximum of five CPG units in each network. A fixed mutation rate was used, but no crossover operator. The fitness was defined as the inverse of the least squares error between the generated output signal and the desired target signal. A total of 10 runs were made for each signal type (hip, knee, ankle, saw tooth, and spiking wave forms), for each CPG type. For the hip joint signal, 300 individuals were evolved for 100 generations, and for the remaining signals, 500 individuals were evolved for 500 generations.

The hip joint target signal shows a compelling similarity with an ordi-

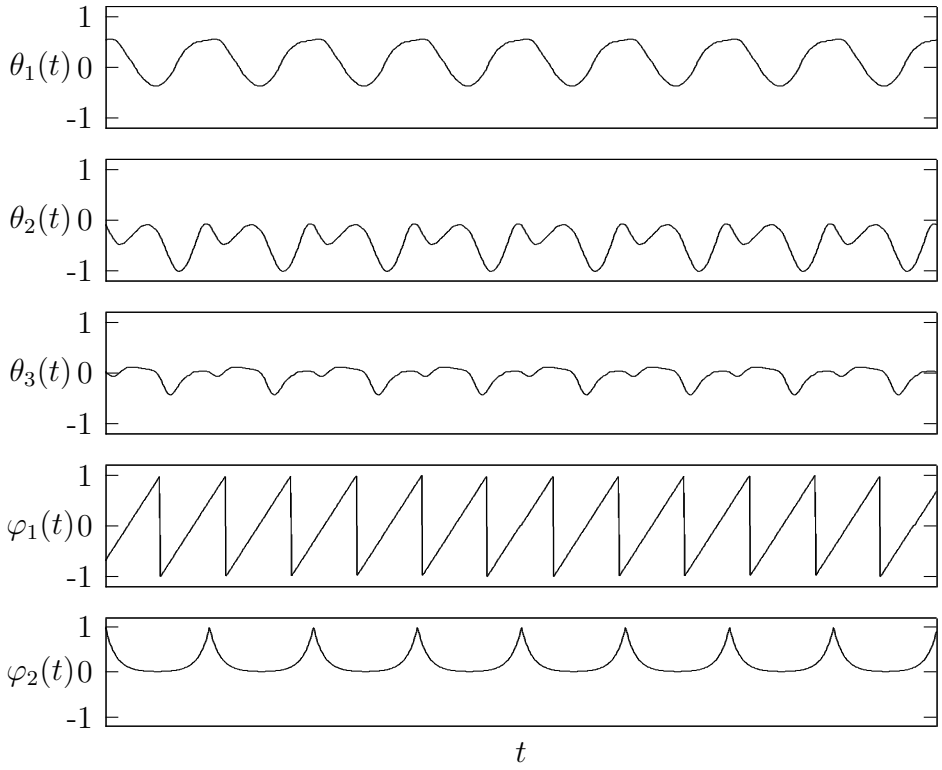

Figure 5.5: Captured data from real human locomotion, displaying joint angles of the hip (θ_1), knee (θ_2), and ankle (θ_3) joints, together with two artificially generating test signals (φ_1 and φ_2). In the case of data from human locomotion, i.e. θ_1, θ_2, and θ_3 above, the data shown here were measured from the graphs of Figure 2 in [110].

nary sine wave, and the knee and ankle joint signals could in principle be constructed given sinusoidal signals with a few different frequencies.

In this investigation the CTRNN-based CPG networks and the networks based on Ekeberg's model were not able to produce any useful signals at all. The Matsuoka oscillator and the nonlinear oscillator, on the other hand, showed similar performance except in the case of the saw tooth signal and the spiking signal. In that case, the Matsuoka model outperformed the nonlinear oscillator significantly. Therefore, the Matsuoka oscillator seemed to be the most suitable for generating signals which cannot be reproduced with a few sinusoidal oscillations. Hence, the Matsuoka oscillator unit, as described by Taga [116, 117, 119], was adopted for development of CPG networks for bipedal locomotion, see Paper VI.

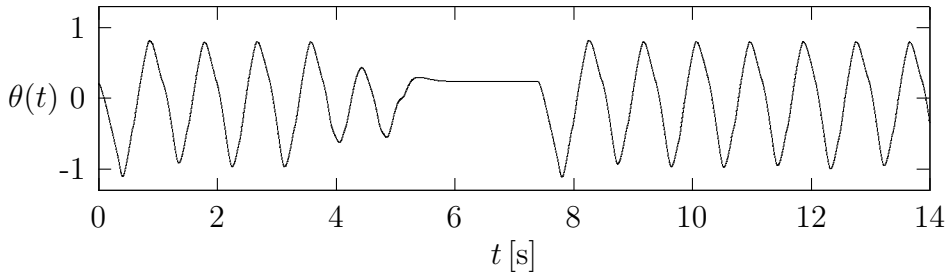

Figure 5.6: Angular motion of the left hip joint during a stop-and-go procedure.

5.5 Evolution of CPGs for locomotion

With the Matsuoka CPG model an elementary network structure can be constructed, where each joint is assigned a half-center oscillator, as depicted in Figure 3 in Paper VI. By using such a structure, straightforward control of flexion and extension can be achieved. As described in Subsection 5.4.2, one could either interpret the CPG output signals as motor torques directly, or as desired joint angles for the robot's joints to follow. The former approach was used in Paper VI, where it is described in detail. Due to time limitations, the latter approach had to be investigated after the submission of Paper VI.

5.5.1 Angle-generating CPG network

Using the two-point support (described in Paper VI) for a duration of two seconds, as well as the same feedback structure, and all joints except hip_3 unlocked, an approach was also examined where the CPG network was used to generate desired joint angles. Then, the appropriate motor torques were generated via a set of standard PD regulators.

With this approach, the results were significantly improved, both in terms of locomotion speed and stability of the best individuals, and in terms of evolution time. For instance, the best individual in the case of angle-generating CPGs reached a distance of 52.46 m, and had an average locomotion speed of 1.31 m/s. The corresponding values for the best individual in the case of torque-generating CPGs was a distance of 35.56 m, and an average locomotion speed of 0.90 m/s, as shown in the 10^{th} row of Table 1 in Paper VI. Thus, the improvement when using angle generating CPGs was approximately 48 and 46 per cent, respectively. Furthermore, individuals of good walking stability emerged quite early in the evolution: Around genera-

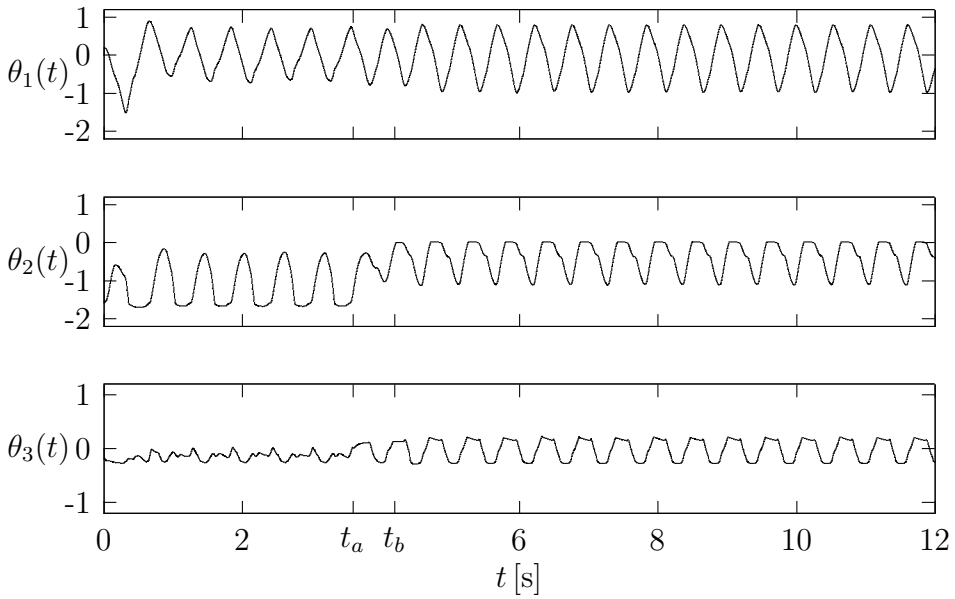

Figure 5.7: Angular motions of the hip (θ_1), knee (θ_2), and ankle (θ_3) joints of the left leg during a gait transition. The gait transition occurs between times t_a and t_b. At time t_a the CPG network is replaced by another one, and the bias u_0 is slightly reduced. The bias value is then gradually increased again towards its original value, which it reaches at time t_b. In order to enable a smooth gait transition when the CPG network is changed, the bias signal must be reduced and then gradually increased, as described above. If, instead, it is kept constant during the network switching, spikes of abnormal magnitude would occur in the output signals of the CPG network, as can be seen in Figure 4.7, where the bias and time parameters were changed momentarily.

tion 25, an individual capable of walking the whole evaluation time of 20 s without falling emerged. In the case of torque-generating CPGs, such an individual did not emerge until after 800 generations. Furthermore, the best individual equipped with angle-generating CPGs was able to walk, without falling, during a testing time of 20 minutes, but the best individual equipped with torque-generating CPGs did not walk more than 43 s when tested. Obviously, adding a PD regulator to each joint does improve the possibility of evolving good bipedal gaits.

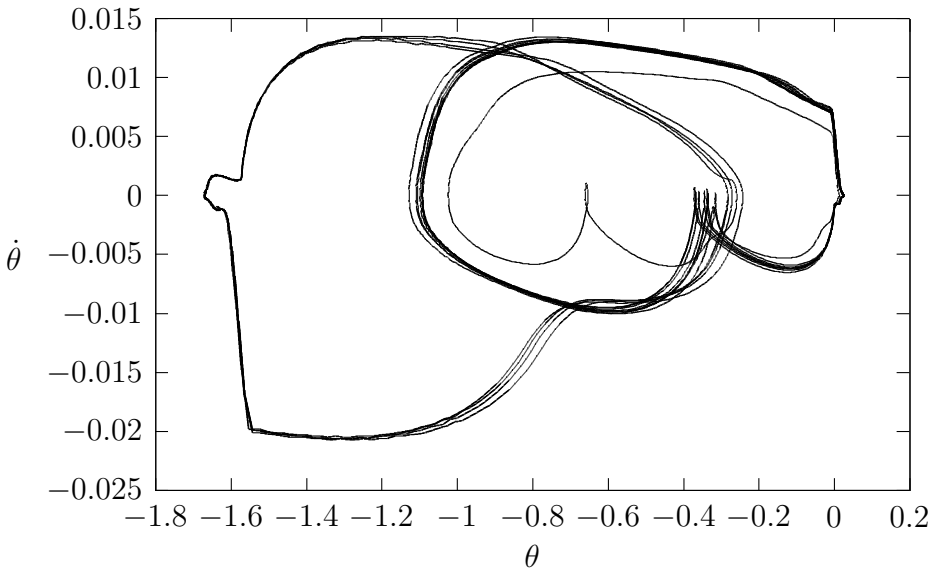

Figure 5.8: Phase portrait of a gait during a transition. In this figure the joint variable angular velocity $\dot{\theta}$ is plotted against the angular displacement variable $\theta(t)$ of the left knee of the simulated robot used in Paper VI. (The variable θ corresponds to θ_2 in Figure 5.7). As shown here, the gait pattern transition can be interpreted as a bifurcation between different types of limit cycles. Compared to Figure 4.4, where the phase portrait is shown after the gait transition, this figure also shows the phase portrait before the transition. Furthermore, the single trajectory generated during the actual transition is depicted as well.

5.5.2 Robustness and gait transitions

When applied to a real physical robot, the obtained CPG networks should be able to handle physical perturbations in order to be useful. Furthermore, the CPG networks should enable gait transitions, i.e. changing the robot's gait from slow to fast, stopping a gait and starting again etc. In order to address these issues, some additional tests of the CPG networks have been carried out.

During evolution the initial output values of the CPGs were always set to zero. When the best individuals were tested for random initialization of these parameters instead, only the angle-generating CPG network was able to ensure stability of the robot during a full evaluation time. The torque-generating networks were not able to maintain the robot's balance during walking for more than some fraction of the evaluation time, when the initial conditions were set at random.

In order to investigate the networks' ability of controlling the speed and the pattern of locomotion, modifications of the time parameters, τ_u and τ_v, and the bias u_0 were considered. When the bias was slightly changed from its original value, the torque-generating CPG networks were not able to maintain the robot's balance at all. When applying the same modification to the angle-generating networks, the locomotion speed could be decreased by as much as 50%. However, in order to ensure walking stability, the bias had to remain at the original value of 1.0 for the hip_2 joints.

Since on-line manipulation of the robot's motion was shown to be possible, using the angle-generating network, two gait transition procedures were successfully implemented. First, a stop-and-go routine was tested, and second, a complete gait transition was realized by smoothly changing the CPG network to another one, see Figures 5.6, 5.7, and 5.8.

5.5.3 Discussion and comparison with LGP

The results of these investigations, as described above, clearly show that the process of artificial evolution can be used for designing functioning CPG network structures, as well as optimizing the parameters thereof, in order to achieve efficient and stable bipedal walking for a simulated robot. Moreover, the best generated gaits show anthropomorphic shape, and the speed of locomotion was indeed realistic, i.e. almost 1 m/s for a robot with a height of approximately 0.75 m (see Paper VI). These results are clearly superior to the results obtained with LGP (see Papers II and III). Furthermore, it has been shown that, with the generated CPG networks, the walking motion of the robot can be controlled and manipulated in a real-time fashion, by tuning certain parameters.

In the beginning of the evolutionary process, different problems arose in the two cases (LGP vs. CPGs): In the case of LGP, a behavior emerged in which the robot stood up, idle until only a few moments of evaluation time was left, and then took a large leap forward. In that way, the individual could get rather good fitness, even though it did not walk. Furthermore, when the robot eventually started to walk in a more continuous fashion, the gaits did not look very human-like at all. Therefore, the energy discharging function was introduced, as mentioned in Subsection 3.3.1. In the case of CPGs, another problem appeared, namely that the robot could not keep its balance at all in the beginning of the evaluation period. As soon as the robot started to move, it fell over. In order to help the robot to keep its balance in the beginning, the support structure was introduced as discussed above. In that way, walking behavior started to emerge.

Chapter 6

Conclusions and future work

6.1 Conclusions

The major conclusions that can be drawn from the work described in this thesis can be summarized as follows:

(1) It is indeed possible to obtain significant improvements of rudimentary bipedal gaits for physical bipedal robots, by applying an EA to a hand-coded gait pattern (see Papers II and VII). Although evolution of gaits in real hardware is rather complicated, that procedure is well motivated by the fact that physical robot hardware is afflicted with individual variations and minor defects that are very hard to capture in a simulation, as pointed out in Section 5.1. Furthermore, despite the fact that evolution carried out directly in hardware imposes some restrictions on the setup of the EA (see Section 5.1), as compared to the case in which evolution is performed in a simulator, evolution of gaits in physical robots is realizable.

(2) In Papers II and III it was shown that it is possible to evolve the structure of a gait-generating controller program for a simulated bipedal robot with a high number of DOFs, starting without any *a priori* knowledge on how to walk. Adding an energy discharging function, so that the robot is given a finite amount of energy to be used for walking, helped to improve the final result, although fully anthropomorphic gait was never obtained with this (LGP) approach.

(3) By contrast, in Paper VI, it was shown that controllers based on CPGs can generate anthropomorphic gaits which, moreover, can be easily adjusted with regard to walking speed and gait transitions. It has also be shown that evolution-based optimization techniques can be successfully applied in the design process of the CPG network structures for bipedal gait synthesis (see Paper VI). In this approach, both the structure *and* the parameters of the

CPG networks were subject to evolutionary optimization.

(4) Finally, in Papers IV and V some properties of the UF method for behavior selection were studied for the case of wheeled robots: As described in Paper IV, the UF method can generate better behavior selection systems if the polynomial degree of the utility functions is increased. On the other hand, for lower polynomial degrees, solutions can be found more rapidly. Furthermore, in Paper V the performance of different EAs in connection with the UF method was studied. It was shown that an ordinary genetic algorithm with fixed-length chromosomes performs at least as well as modified evolutionary methods in which the chromosomes are allowed to vary in size.

6.2 Future work

In the studies involving real, physical robots (see Papers I and VII), there is indeed room for several improvements: Since continuous human guidance of the process is necessary, the long evaluation times pose a problem. It would therefore be beneficial to extend the experimental setup with some device for automatic positioning of the robot between the evaluations, in order to decrease the burden for the experimenter. Furthermore, continuous surveillance of the arena and the robot by means of a fixed camera could be used to monitor the robot's locomotion path, which could then be included in the fitness measure. Another interesting possibility would be to base the fitness value on the heritage of a given individual, i.e. to include a term based on its ancestors' fitness values, perhaps by using a discounting factor for individuals from earlier generations. The elimination of a good individual, based on a single failure, could then be avoided, without having to carry out time-consuming re-evaluations of each individual.

In the case of generation of bipedal gaits using LGP, see Papers II and III, the obtained gaits certainly need improvements, although it was demonstrated that the method could generate locomotion behavior. One possible improvement, which has already been investigated briefly, is to use two programs for controlling the robot: In this case, one program controls the robot in the initial start-up moment, and a second, cyclic program controls the robot during its continued motion. In the current setup, described in Paper III, the point at which the second program takes control over the robot is pre-defined. A natural extension would be to introduce utility functions and to use the UF method in order to determine the transition point. Moreover, in that case it would be possible to use a repertoire of gait programs in the robot, allowing it to cope with different situations occurring in its environment. Furthermore, including similar feedback as in the case of CPGs

(see Paper VI) might help to improve the generated bipedal gaits.

In the CPG experiment, described in Paper VI, the connection paths of the feedback network were pre-defined. A topic for further work would be to investigate whether the structure of the feedback network could be evolved instead. Since the feedback paths were specified in an *ad hoc* manner, there is no guarantee that these paths are optimal. Exploration of that issue would be a suitable task for an EA, and might lead to increased performance of the CPG network controller.

Furthermore, investigating how different kinds of feedback, such as information regarding foot joint angles, can influence the quality of the generated gaits would be a relevant topic for future study, particularly in view of the importance of sensory feedback in the case of human bipedal walking (see Subsection 4.2.2). Another topic in the CPG approach that needs further development concerns the gait transitions, described in Section 5.5. In Paper VI, gait transitions were accomplished by switching between two different CPG networks controlling the robot. However, the gait transition maneuver required quite some manual tuning of the CPG parameters in order to work fully. In a robot intended for real-world applications, the transition must be carried out in a more automatic fashion. As in the LGP case described above, a logical possibility regarding gait transitions in the case of CPG networks would be to use utility functions in order to determine the transition point.

The studies of the UF method, described in Papers V and IV, suggests a promising potential for the method in behavior selection for autonomous robots. However, the transfer of results obtained with the UF method to real, physical robots needs to be further investigated. For that reason, the method will be implemented in a real-world application during the spring of 2007, a project in which the author will participate. The task to which the UF method will be applied concerns a robot for transportation and delivery.

Bibliography

[1] Adolfsson, J. *Passive control of mechanical systems: Bipedal walking and autobalancing*. PhD thesis, KTH, Mechanics, 2001.

[2] Arkin, R. C. *Behavior-Based Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, Cambridge, MA, and London, England, 1998.

[3] Banzhaf, W., Nordin, P., Keller, R., and Francone, F. *Genetic Programming – An Introduction: On the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1998.

[4] Beer, R. D. On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4):469–509, 1995.

[5] Beer, R. D., Quinn, R. D., Chiel, H. J., and Ritzmann, R. E. Biologically inspired approaches to robotics: What can we learn from insects? *Communications of the ACM*, 40(3):30–38, 1997.

[6] Beer, R. D., Chiel, H. J., and Gallagher, J. C. Evolution and analysis of model cpgs for walking: II. general principles and individual variability. *Journal of Computational Neuroscience*, 7(2):119–147, 1999.

[7] Bekey, G. A. *Autonomous Robots: From Biological Inspiration to Implementation and Control (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[8] Bekey, G. A. Biologically inspired control of autonomous robots. *Robotics and Autonomous Systems*, 18(1–2):21–31, 1996.

[9] Berbyuk, V., Boström, A., Lytwyn, B., and Peterson, B. Optimization of control laws of the bipedal locomotion systems. In *Proceedings of*

the International Conference on Advances in Computational Multibody Dynamics, pages 713–728, Lisbon, Portugal, 1999.

[10] Berbyuk, V., Lytwyn, B., and Demydyuk, M. Energy-optimal control of underactuated bipedal locomotion systems. In *Proceedings of the ECCOMAS Thematic Conference Multibody Dynamics 2005 on Advances in Computational Multibody Dynamics*, pages 1–15, Madrid, Spain, 21–24 June 2005.

[11] Betts, J. *Practical Methods for optimal control using nonlinear programming*. SIAM, Philadelphia, PA, USA, 2001.

[12] Blumberg, B. Action-selection in hamsterdam: Lessons from ethology. In *SAB94: Proceedings of the third international conference on Simulation of adaptive behavior : from animals to animats 3*, pages 108–117, Cambridge, MA, USA, 1994. MIT Press.

[13] Blumberg, B. *Old Tricks, New Dogs: Ethology and Interactive Creatures*. PhD thesis, The Media Lab, MIT, Cambridge MA, USA, 1996.

[14] Boeing, A., Hanham, S., and Bräunl, T. Evolving autonomous biped control from simulation to reality. In *Proceedings of the Second International Conference on Autonomous Robots and Agents, ICARA'04*, Palmerston North, New Zealand, 2004.

[15] Bourquin, Y. Self-organization of locomotion in modular robots. Master's thesis, Computer and Communication Sciences, BIRG, EPFL, Lausanne, Switzerland, 2004.

[16] Braitenberg, V. *Vehicles: Experiments in Synthetic Psychology*. The MIT Press, 1984.

[17] Brameier, M. *On Linear Genetic Programming*. PhD thesis, Dortmund University, Dortmund, Germany, 2003.

[18] Brooks, R. A. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[19] Brooks, R. A. Intelligence without representation. *Artificial Intelligence*, 47(1–3):139–159, 1991.

[20] Brooks, R. A. Intelligence without reason. Technical report, Cambridge, MA, USA, 1991.

[21] Brooks, R. A. *Cambrian intelligence: The early history of the new AI.* MIT Press, Cambridge, MA, USA, 1999.

[22] Brown, T. G. The intrinsic factors in the act of progression in the mammal. *Proc R Soc London Ser*, 84:308–319, 1911.

[23] Brown, T. G. The factors in rhythmic activity of the nervous system. *Proc R Soc London Ser*, 85:278–289, 1912.

[24] Brown, T. G. On the nature of the fundamental activity of the nervous centers: Together with an analysis of the conditioning of rhythmic activity in progression, and a theory of the evolution of function in the nervous system. *Journal of Physiology*, 48:18–46, 1914.

[25] Bäck, T. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms.* Oxford University Press, Oxford, UK, 1996.

[26] Chiel, H. J., Beer, R. D., and Gallagher, J. C. Evolution and analysis of model cpgs for walking: I. dynamical modules. *Journal of Computational Neuroscience*, 7(2):99–118, 1999.

[27] Collins, S. H., Ruina, A. L., Tedrake, R., and Wisse, M. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307:1082–1085, 2005.

[28] Crespi, A., Badertscher, A., Guignard, A., and Ijspeert, A. J. An amphibious robot capable of snake and lamprey-like locomotion. In *Proceedings of the 35th International Symposium on Robotics (ISR'04)*, Paris, France, 2004.

[29] Deb, K. Genetic algorithm in search and optimization: The technique and applications. In *Proceedings of the International Workshop on Soft Computing and Intelligent Systems*, pages 58–87, Calcutta, India, 1998.

[30] Dingwell, J. B., Cusumano, J. P., Cavanagh, P. R., and Sternad, D. Local dynamic stability versus kinematic variability of continuous overground and treadmill walking. *Biomechanical Engineering*, 123(1):27–32, 2001.

[31] Dorigo, M. and Colombetti, M. *Robot Shaping: An Experiment in Behavior Engineering.* The MIT Press, 1997.

[32] Dorigo, M. and Schnepf, U. Genetics-based machine learning and behaviour based robotics: A new synthesis. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(1):141–154, 1993.

[33] Duysens, J. and Van de Crommert, H. W. A. A. Neural control of locomotion; Part 1: The central pattern generator from cats to humans. *Gait and Posture*, 7(2):131–141, 1998.

[34] Ekeberg, Ö. A combined neuronal and mechanical model of fish swimming. *Biological Cybernetics*, 69(5-6):363–374, October 1993.

[35] Ekeberg, Ö. and Grillner, S. Simulations of neuromuscular control in lamprey swimming. *Phil. Trans. R. Soc. Lond. B*, 354(1385):895–902, 1999.

[36] Ekeberg, Ö., Wallén, P., Lansner, A., Tråvén, H., Brodin, L., and Grillner, S. A computer based model for realistic simulations of neural networks. I: The single neuron and synaptic interaction. *Biological Cybernetics*, 65(2):81–90, 1991.

[37] Endo, G., Morimoto, J., Nakanishi, J., and Cheng, G. An empirical exploration of a neural oscillator for biped locomotion control. In *Proceedings of the International Conference on Robotics and Automation (ICRA'04)*, volume 3, pages 3036–3042, 2004.

[38] Fernández, F., Galeano, G., and Gómez, J. A. Comparing synchronous and asynchronous parallel and distributed genetic programming models. In *Proceedings of the 5th european conference on genetic programming, EuroGP'05*, pages 326–336, Kinsale, Ireland, 2002.

[39] Ferrée, T., Marcotte, B., and Lockery, S. Neural network models of chemotaxis in the nematode *caenorhabditis elegans*. In *Advances in Neural Information Processing Systems 9, NIPS*, pages 55–61, 1996.

[40] Ficici, S., Watson, R., and Pollack, J. Embodied evolution: A response to challenges in evolutionary robotics. In *Proceedings of the Eighth European Workshop on Learning Robots*, pages 14–22, 1999.

[41] Floreano, D. and Mondada, F. Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 26:396–407, 1996.

[42] Fogel, D. B. An introduction to simulated evolutionary optimization. *IEEE Transactions on Neural Networks*, 5(1):3–14, January 1994.

[43] Forlizzi, J. and DiSalvo, C. Service robots in the domestic environment: A study of the roomba vacuum in the home. In *Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction, HRI'06*, pages 258–265, New York, NY, USA, 2006.

[44] Fujita, M. Digital creatures for future entertainment robotics. In *Proceedings of the International Conference on Robotics and Automation, ICRA'00*, pages 801–806, San Francisco, CA, USA, 2000.

[45] Fujita, M. and Kitano, H. Development of an autonomous quadruped robot for robot entertainment. *Autonomous Robots*, 5(1):7–18, 1998.

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

[47] Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1989.

[48] Goswami, A. Postural stability of biped robots and the foot-rotation indicator (FRI) point. *The International Journal of Robotics Research*, 18:523–533, 1999.

[49] Goswami, A. and Kallem, V. Rate of change of angular momentum and balance maintenance of biped robots. In *Proceedings of the International Conference on Robotics and Automation, ICRA'04*, volume 4, pages 3785–3790, 2004.

[50] Goswami, A., Espiau, B., and Keramane, A. Limit cycle and their stability in a passive bipedal gait. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 246–251, Minneapolis, MN, USA, April 1996.

[51] Grillner, S. Neurological bases on rhythmic motor acts in vertebrates. *Science*, 228:143–149, 1985.

[52] Grillner, S. Neural networks for vertebrate locomotion. *Scientific American*, 274:64–69, 1996.

[53] Grillner, S. and Zangger, P. How detailed is the central pattern generator for locomotion? *Brain Res*, 88:367–371, 1975.

[54] Grillner, S., Wallén, P., Brodin, L., and Lansner, A. Neuronal network generating locomotor behavior in lamprey: Circuitry, transmitters, membrane properties, and simulation. *Annual Review of Neuroscience*, 14(1):169–199, 1991.

[55] Grillner, S., Deliagina, T., Ekeberg, Ö., El Manira, A., Hill, R., Lansner, A., Orlovsky, G., and Wallen, P. Neural networks that coordinate locomotion and body orientation in lamprey. *Trends in Neurosciences*, 18(6):270–279, 1995.

[56] Hirai, K. The Honda humanoid robot: Development and future perspective. *Industrial Robot: An International Journal*, 26(4):260–266, 1999.

[57] Holland, J. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, MI, USA, 1975.

[58] Hornby, G. S., Takamura, S., Yokono, J., Hanagata, O., Yamamoto, T., and Fujita, M. Evolving robust gaits with aibo. In *Proceedings of the International Conference on Robotics and Automation, ICRA'00*, volume 3, pages 3040–3045, San Francisco, CA, USA, 2000.

[59] Huang, Q. and Nakamura, Y. Sensory reflex control for humanoid walking. *IEEE Transactions on Robotics and Automation*, 21(5):977–984, 2005.

[60] Huelsbergen, L. Toward simulated evolution of machine–language iteration. In *Proceedings of the '96 Conference on Genetic Programming*, pages 315–320, Stanford University, CA, USA, 1996.

[61] Hyon, S. and Emura, T. Symmetric walking control: Invariance and global stability. In *Proceedings of the International Conference on Robotics and Automation, ICRA'05*, pages 1443–1450, 2005.

[62] Ijspeert, A. J. A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander. *Biological Cybernetics*, 84(5):331–348, 2001.

[63] Ijspeert, A. J. and Cabelguen, J. M. Gait transitions from swimming to walking: investigation of salamander locomotion control using nonlinear oscillators. In *Proceedings of the 2nd International Symposium on Adaptive Motion of Animals and Machines, AMAM'03*, Kyoto, Japan, 2003.

[64] Ijspeert, A. J., Hallam, J., and Willshaw, D. Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology. *Adaptive Behavior*, 7(2):151–172, 1999.

[65] Ishida, T. Development of a small biped entertainment robot qrio. In *Proceedings of the International Symposium on Micro-Nanomechatronics and Human Science, and The Fourth Symposium on Micro-Nanomechatronics for Information-Based Society*, pages 23–28, November 2004.

[66] Ishiguro, A., Fujii, A., and Hotz, P. E. Neuromodulated control of bipedal locomotion using a polymorphic cpg circuit. *Adaptive Behavior*, 11(1):7–17, 2003.

[67] Iwasaki, T. and Zheng, M. Sensory feedback mechanism underlying entrainment of central pattern generator to mechanical resonance. *Biol Cybernetics*, 94(4):245–261, 2006.

[68] Jakobi, N. *Minimal Simulations for Evolutionary Robotics*. PhD thesis, COGS, 1998.

[69] Jakobi, N., Husbands, P., and Harvey, I. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Advances in Artificial Life: 3rd European Conference on Artificial Life, ECAL'95*, LNCS 929, pages 704–720. Springer, 1995.

[70] Kamoun, H., Hall, N. G., and Sriskandarajah, C. Scheduling in robotic cells: Heuristics and cell design. *Operations Research*, 47(6):821–835, 1999.

[71] Kandel, E. R., Schwartz, J. H., and Jessell, T. M. *Principles of Neural Science*. McGraw-Hill, New York, NY, USA, 4th edition, 2000.

[72] Katic, D. and Vukobratovic, M. Survey of intelligent control techniques for humanoid robots. *Intelligent and Robotic Systems*, 37(2):117–141, 2003.

[73] Klaassen, B., Linnemann, R., Spenneberg, D., and Kirchner, F. Biomimetic walking robot scorpion: Control and modeling. *Robotics and Autonomous Systems*, 41(2–3):69–76, 2002.

[74] Kling, U. and Székely, G. Simulation of rhythmic nervous activities. I. function of networks with cyclic inhibitions. *Kybernetik*, 5(3):89–103, September 1968.

[75] Kondo. `http://www.kondo-robot.com/`.

[76] Koza, J. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* The MIT Press, Cambridge, MA, USA, 1992.

[77] Koza, J. R., Keane, M. A., Yu, J., Bennett, F. H., and Mydlowec, W. Automatic creation of human-competitive programs and controllers by means of genetic programming. *Genetic Programming and Evolvable Machines*, 1(1-2):121–164, 2000.

[78] Li, Q., Takanishi, A., and Kato, I. Learning control of compensative trunk motion for biped walking robot based on zmp stability criterion. In *Proceedings of the '92 International Conference on Intelligent Systems*, pages 597–603, 1992.

[79] Lockery, S. R. and Kristian, W. B. Distributed processing of sensory information in the leech. I. input-output relations of the local bending reflex. *J Neurosci.*, 10(6):1811–1815, 1990.

[80] Maes, P. Learning behavior networks from experience. In Varela, F. J. and Bourgine, P., editors, *Proceedings of the First European Conference on Artificial Life*, Paris, France, 1992.

[81] Mahadevan, S. and Connell, J. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55: 311–365, 1992.

[82] Mathayomchan, B. and Beer, R. D. Center-crossing recurrent neural networks for the evolution of rhythmic behavior. *Neural Comput.*, 14 (9):2043–2051, 2002.

[83] Matsuoka, K. Mechanisms of frequency and pattern control in the neural rhythm generators. *Biol Cybern*, 47(2–3):345–353, 1987.

[84] McFarland, D. *Animal Behavior*. Addison-Wesley, 1999.

[85] McFarland, D. and Bösser, T. *Intelligent Behavior in Animals and Robots*. The MIT Press, Cambridge MA, USA, 1993.

[86] McKerrow, P. J. *Introduction to Robotics*. Addison-Wesley Publishing Co., Inc., Boston, MA, USA, 1991.

[87] Miglino, O., Lund, H. H., and Nolfi, S. Evolving mobile robots in simulated and real environments. *Artificial Life*, 2(4):417–434, 1995.

[88] Miglino, O., Nafasi, K., and Taylor, C. E. Selection for wandering behavior in a small robot. *Artificial Life*, 2(1):101–116, 1995.

[89] Mitchell, M. *An introduction to genetic algorithms*. MIT Press, Cambridge, MA, USA, 1996.

[90] Mitchell, T. *Machine Learning*. McGraw Hill, 1997.

[91] Mojon, S. Using nonlinear oscillators to control the locomotion of a simulated biped robot. Master's thesis, Computer and Communication Sciences, BIRG, EPFL, Lausanne, Switzerland, 2004.

[92] Morgenstern, O. and von Neumann, J. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[93] Nakanishi, J., Morimoto, J., Endo, G., Cheng, G., Schaal, S., and Kawato, M. An empirical exploration of phase resetting for robust biped locomotion with dynamical movement primitives. In *Proceedings of the international conference on Intelligent Robots and Systems, IROS'04*, volume 1, pages 919–924, 2004.

[94] Nash, S. and Sofer, A. *Linear and nonlinear programming*. McGraw Hill, 1996.

[95] Nolfi, S. and Floreano, D. *Evolutionary Robotics. The Biology, Intelligence, and Technology of Self-organizing Machines*. MIT Press, Cambridge, MA, USA, 2001.

[96] Ogihara, N. and Yamazaki, N. Generation of human bipedal locomotion by a bio-mimetic neuro-musculo-skeletal model. *Biological Cybernetics*, 84(1):1–11, 2001.

[97] Ok, S., Miyashita, K., and Hase, K. Evolving bipedal locomotion with genetic programming —a preliminary report—. In *Proceedings of the Congress on Evolutionary Computation, CEC'01*, pages 1025–1032, Seoul, South Korea, 2001.

[98] Ott, E. *Chaos in Dynamical Systems*. Cambridge University Press, Cambridge, UK, 1993.

[99] Parker, G. B. The incremental evolution of gaits for hexapod robots. In *Proceedings of the Genetic and Evolutionary Computation Conference*, San Francisco, CA, USA, 2001.

[100] Pettersson, J. and Wahde, M. Application of the utility function method for behavioral organization in a locomotion task. *IEEE Transactions on Evolutionary Computation*, 9(5):506–521, October 2005.

[101] Pettersson, J., Sandholt, H., and Wahde, M. A flexible evolutionary method for the generation and implementation of behaviors for humanoid robots. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, pages 279–286, Waseda University, Tokyo, Japan, 22-24 November 2001. Humanoid Robotics Institute.

[102] Pinto, C. and Golubitsky, M. Central pattern generators for bipedal locomotion. *Mathematical Biology*, 2005. Submitted.

[103] Pirjanian, P. Behavior coordination mechanisms – state-of-the-art. Technical Report 99–375, Institute of Robotics and Intelligent Systems, University of Southern California, CA, USA, 1999.

[104] Pollack, J. B., Lipson, H., Hornby, G., and Funes, P. Three generations of automatically designed robots. *Artificial Life*, 7(3):215–223, 2001.

[105] Quinn, R. D. and Espenschied, K. S. Control of a hexapod robot using a biologically inspired neural network. In *Proceedings of the workshop on "Locomotion Control in Legged Invertebrates" on Biological neural networks in invertebrate neuroethology and robotics*, pages 365–381, San Diego, CA, USA, 1993.

[106] Reil, T. and Massey, C. Biologically inspired control of physically simulated bipeds. *Theory in Biosciences*, 120(3–4):327–339, 2001.

[107] Reynolds, C. W. Evolution of corridor following behavior in a noisy world. In *Proceedings of the 3rd International Conference on Simulation of Adaptive Behaviour,* SAB'94, pages 402–410.

[108] Russell, S. J. and Norvig, P. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[109] Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., Higaki, N., and Fujimura, K. The intelligent asimo: System overview and integration. In *Proceedings of the International Conference on Intelligent Robots and System, IROS'02*, volume 3, pages 2478–2483, Lausanne, Switzerland, 2002.

[110] Sasaki, K. and Neptune, R. R. Differences in muscle function during walking and running at the same speed. *Journal of Biomechanics*, 2006. In press.

[111] Shephard, G. M. *Neurobiology*, chapter 20, pages 435–451. Oxford University Press, 3rd edition, 1994.

[112] Siegwart, R. and Nourbakhsh, I. R. *Introduction to Autonomous Mobile Robots*. Bradford Book, 2004.

[113] Sims, K. 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.

[114] Sims, K. Evolving 3D morphology and behavior by competition. *Artificial Life*, 1(4):353–372, 1994.

[115] Smith, R. Open dynamics engine (ODE). `http://ode.org/`.

[116] Taga, G. A model of the neuro-musculo-skeletal system for anticipatory adjustment of human locomotion during obstacle avoidance. *Biological Cybernetics*, 78(1):9–17, 1998.

[117] Taga, G. Nonlinear dynamics of the human motor control - real-time and anticipatory adaptation of locomotion and development of movements. In *Proc 1st Int Symp on Adaptive Motion of Animals and Machines (AMAM'00)*, 2000.

[118] Taga, G. Emergence of bipedal locomotion through entrainment among the neuro-musculo-skeletal system and the environment. *Physica D Nonlinear Phenomena*, 75:190–208, 1994.

[119] Taga, G., Yamaguchi, Y., and Shimizu, H. Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. *Biological Cybernetics*, 65:147–159, 1991.

[120] Tsai, L. W. *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*. Wiley, New York, NY, USA, 1999.

[121] Tyrell, T. *Computational Mechanisms for Action Selection*. PhD thesis, University of Edinburgh, Edinburgh, Scotland, UK, 1993.

[122] Ursem, R. K. *Models for Evolutionary Algorithms and Their Applications in System Identification and Control Optimization*. PhD thesis, EVALife, Department of Computer Science, University of Aarhus, 2003.

[123] Van Wezel, B. M. H., Ottenhoff, F. A. M., and Duysens, J. Dynamic control of location-specific information in tactile cutaneous reflexes from the foot during human walking. *The Journal of Neuroscience*, 17(10):3804–3814, 1997.

[124] Vukobratovic, M. and Juricic, D. Contribution to the synthesis of biped gait. *IEEE Transactions on Bio-Medical Engineering*, 16(1), 1969.

[125] Vukobratovic, M. and Stepanenko, J. On the stability of anthropomorphic systems. *Mathematical Biosciences*, 15:1–37, 1972.

[126] Vukobratovic, M., Frank, A. A., and Juricic, D. On the stability of biped locomotion. *IEEE Transactions on Bio-Medical Engineering*, 17 (1):25–36, 1970.

[127] Wahde, M. A method for behavioural organization for autonomous robots based on evolutionary optimization of utility functions. *Journal of Systems and Control Engineering*, 217(4):249–258, September 2003.

[128] Wahde, M. and Sandholt, H. Evolving complex behaviors on autonomous robots. In *Proceedings of the 7th UK Mechatronics Forum International Conference*, Atlanta, GA, USA, 6–8 September 2000. Pergamon Press.

[129] Wallén, P., Ekeberg, Ö., Lansner, A., Brodin, L., Tråvén, H., and Grillner, S. A computer-based model for realistic simulations of neural networks. II: The segmental network generating locomotor rhythmicity in the lamprey. *J. Neurophysiol.*, 68:1939–1950, 1992.

[130] Watson, R. A., Ficici, S. G., and Pollack, J. B. Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 335–342, Washington D.C., USA, 1999.

[131] Whittle, M. C. *Gait analysis: An introduction*. Butterworth-Heinemann, 3rd edition, 2002.

[132] Williamson, M. M. Neural control of rhythmic arm movements. *Neural Networks*, 11(7–8):1379–1394, 1998.

[133] Yamaguchi, J., Inoue, S., Nishino, D., and Takanishi, A. Development of a bipedal humanoid robot having antagonistic driven joints and three DOF trunk. In *Proceedings of the International Conference on Intelligent Robots and Systems, IROS'98*, volume 1, pages 96–101, 1998.

[134] Yamaguchi, J., Soga, E., Inoue, S., and Takanishi, A. Development of a bipedal humanoid robot-control method of whole body cooperative dynamic biped walking. In *Proceedings of the International Conference on Robotics and Automation, ICRA'99*, pages 368–374, 1999.

[135] Zehr, E. P. and Duysens, J. Regulation of arm and leg movement during human locomotion. *Neuroscientist*, 10(4):347–361, 2004.

[136] Ziegler, J., Wolff, K., Nordin, P., and Banzhaf, W. Constructing a small humanoid walking robot as a platform for the genetic evolution of walking. In Rückert, U., Sitte, J., and Witkowski, U., editors, *Proceedings of the 1st International Conference on Autonomous Minirobots for Research and Edutainment, AMiRE'01*, volume 97 of *HNI*, pages 51–59, 2001.

[137] Ziegler, J., Barnholt, J., Busch, J., and Banzhaf, W. Automatic evolution of control programs for a small humanoid walking robot. In *Proceedings of the 5th International Conference on Climbing and Walking Robots, CLAWAR'02*, 2002.

Paper I

Evolution of efficient gait with humanoids using visual feedback

In

Proceedings of the IEEE-RAS International Conference on Humanoid Robots, pages 99–106, Tokyo, Japan, November 22–24, 2001.

Evolution of Efficient Gait with Humanoids Using Visual Feedback

Krister Wolff and Peter Nordin

Department of Physical Resource Theory, Complex Systems Group
Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg, Sweden
{wolff, nordin}@fy.chalmers.se

Abstract

In this paper we present the autonomous, walking humanoids Priscilla, ELVIS and ELVINA and an experiment using evolutionary adaptive systems. We also present the anthropomorphic principles behind our humanoid project and the multistage development methodology. The adaptive evolutionary system used is a steady state evolutionary strategy running on the robot's onboard computer. Individuals are evaluated and fitness scores are automatically determined using the robots onboard digital cameras and near-infrared range sensor. The experiments are performed in order to optimize a by hand developed locomotion controller. By using this system, we evolved gait patterns that locomote the robot in a straighter path and in a more robust way, than the previously manually developed gait did.

Keywords

Evolutionary Robotics, Genetic Programming.

1. Introduction

The applications of robots with human-like dimensions and motion capabilities, humanoid robots, are plentiful. Humanoid robots constitute both one of the largest potentials and one of the largest challenges in the fields of autonomous agents and intelligent robotic control. In a world where man is the standard for almost all interactions, humanoid robots have a very large potential acting in environments created for human beings [1].

In traditional robot control programming, an internal model of the system is derived and the inverse kinematics can thus be calculated. The trajectory for movement between given points in the working area of the robot is then calculated from the inverse kinematics. Even though this still is a very common approach, we propose for several reasons the concept of genetic programming for control programming of so-called bio-inspired robots [2] as e.g. a humanoid. The traditional geometric approach to robot control, based on modeling of the robot and derivation of leg trajectories, is computationally expensive and requires fine-tuning of several parameters in the equations describing the inverse kinematics [3]. Conventional industrial robots are designed in such a way that a model can be easily derived, but for the development of bio-inspired robots, this is not a primary design principle. Thus, a model of the system is very hard to derive or to complex so that a model-based calculation of actuator commands requires too much time for reactive tasks [2]. For a robot that is conceived to operate in an actual human living environment, it is impossible for the programmer to consider all eventualities in advance. The robot is therefore required to have an adaptation mechanism that is able to cope with unexpected situations.

The anthropomorphic principle behind humanoids might be a stronger motivation factor then conventionally assumed. Consider for example the phenomenon of human left-handedness. Left-handed persons have been shown to have a shorter expected life length than right-handed persons. The standard explanation for the higher mortality rate is a higher accident frequency and the assumed explanation for this deviation is due to the fact that the world is built for right-handed people [11]. If such a minute deviation in behavior could cause accident frequencies measurable in as statistically significant mortality biases, we could expect considerable difficulties for a robot working in a human environment. The differences between human and robot will always be bigger there, than the difference between a left-handed and right-handed person. We aim at exploring and evaluating the consequences of a strong anthropomorphic principle where humanoids are built with very close correspondence with humans in terms of size, weight, geometry and motion capabilities. We have therefore devised a full-sized autonomous humanoid robot that is built around an

Proceedings of the 2001 IEEE –RAS International Conference on Humanoid Robots

accurate model of a human skeleton –the Priscilla robot.

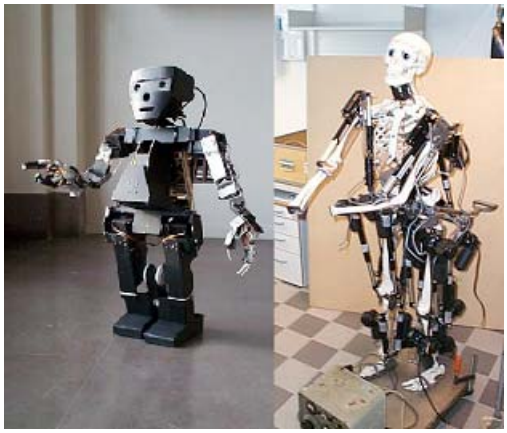

Figure 1. ELVIS (left) and Priscilla (right).

The skeleton design guarantees anthropomorphic geometry and enables close correspondence in movement capabilities.

From a methodological and developmental standpoint are the project guided by more than the anthropomorphic principle. Even though we have simulators for the Priscilla robot we strongly believe in the embodiment principle and we try to make most of our experiments on the full-size autonomous Priscilla robot. However, the over-all efficiency of our humanoid project has turned out to increase when using several smaller size prototypes. The mid-size prototype is called ELVIS and it is about 60 cm tall. The smallest size humanoid is the ELVINA type, about 25 cm tall. Two instances of ELVINA have been built to enable experiments with cooperating humanoids. In this paper we briefly introduce three humanoid robot prototypes.

A third guiding principle is the need for adaptivity when dealing with such a complex object as a humanoid in such a complex environment as everyday human life. We are furthermore using evolutionary algorithms and more specifically genetic programming as the adaptation method. Genetic programming is an efficient method for breeding symbolic structures such as computer programs and behavior definitions [4].

We present work in this paper evolving a gait pattern, using genetic programming and especially evolutionary strategies [4]. To do this, one has to choose between two main alternatives: using a real robot for the evolution, or using a simulated robot. Several experiments with simulations, with different approaches, have been reported recently. Anytime learning can make use of evolutionary computation in a learning module for the robot to adapt to changes in the robot's capabilities without the use of internal sensors [5]. A methodology for developing simulators for evolution of controllers in minimal simulations has been proposed and shown to be successful when transferred to a real, physical octopod robot [6]. This was also compared with a controller that was evolved with a real octopod robot [7]. It was found that it matched better the physical constraints of the robot hardware. Using simulation, ball-chasing behavior has been evolved and successfully transferred to a real AIBO[1] quadruped robot dog [8]. The collisions between the robot and ball had different results in the real world than in the simulated world, however it did not affect ball-chasing performance. When a high degree of accuracy is necessary, it is desirable to be able to evolve with a physical robot. We want to show that evolution of controllers with complex, physical robots can be carried out in reality, although evolving with a simulator would do it many times faster.

The first attempt in using a real, physical robot to evolve gait patterns was made at the University of Southern California [9]. Neural networks were evolved as controllers to get a tripod gait for a hexapod robot with two degrees of freedom for each leg. Recently, a group of researchers at Sony Corporation presented the results of their work with evolving locomotion controllers for dynamic gait of their quadruped robot dog AIBO [10] and [13]. These results show that evolutionary algorithms can be used on complex, physical robots to evolve non-trivial behaviors on those robots. In previous evolution with physical robots has a humanoid, biped robot not been used.

Our test problem is that of developing locomotion controllers for static gaits for our biped robots. Evolution of static walking with a biped robot is much more difficult than it is with a robot that has a greater number of legs. A static gait requires that the projection of the center of mass of the robot on the ground lie within the support polygon formed by feet on the ground [3]. This is obviously easier to fulfill with a robot that got four, six, eight or more legs. However, dealing with biped locomotion leads us into a partly different problem

[1] http://www.aibo.com

domain. When a biped robot is walking (static), it is supported only by one foot at the ground during an appreciable period of time. Only this single foot then constitutes its support polygon. For a biped robot, the area of the support polygon is relatively small, compared to the altitude of where its center of mass is located. The corresponding measure for a robot that got four or more legs is relatively larger. Therefore it is easier for a robot with many legs to maintain balance than it is for a biped robot, as the motion of walking dynamically changes the stability of the robot.

2. Robot Platforms

All the three humanoids[2] carry their main computer power onboard. ELVIS and Priscilla have a small PC laptop, while ELVINA has the EyeBot MK3 controller onboard, carrying it as a backpack. The EyeBot MK3[3] consists of a 32-bit micro-controller board with a graphics display and four push buttons for user input.

2.1 Priscilla

The Priscilla robot consists of a plastic skeleton with titanium reinforcements and linear electric actuators.

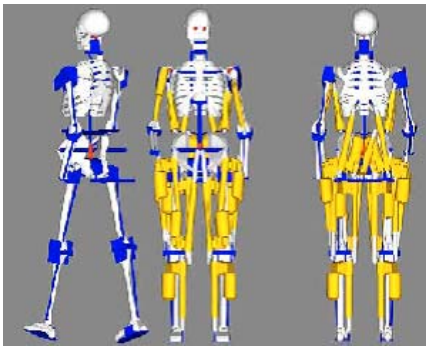

Figure 2. CAD-drawings of Priscilla showing skeleton, titanium reinforcements and actuators.

Linear electric actuators are more accurate and therefore easier to control than pneumatic actuators, which was also considered as an option when designing the robot. Even though pneumatic actuators are more powerful in an autonomous setting, linear electric actuators were chosen for the Priscilla humanoid.

One goal was to make the robot strong and fast enough to be able to walk with normal human walking speed. There are also a number of movements that the robot should be able to carry out when speed is not critical. That could be lifting the arm when holding an object in its hand, or rising from a chair. Such movements dictate other constraints on the actuators.

Once the requirements were defined, suitable actuators were chosen. For the legs Linak La30 actuators with different strokes is used and for the arms Warner Electric La1 actuators. For head movement we use standard off-the-shelf R/C servomotors as actuators, because of their convenience in connection to computer for control.

Figure 3. Picture of the actuators and servo. From top to bottom La30, La1 with stroke 0.115m and 0.06m respectively. To the right the R/C-servo.

The La 30 actuator is equipped with a passive brake that is activated automatically when the motor stops. This prevents the robot from consuming energy when not moving. The smaller La1 actuator is not available with brake. Both the La30 and La1 actuator have built-in potentiometers that provides us with the possibility to get accurate readings within the controller software of the state of each actuator at a certain time. The actuators weights, depending on the stroke, from about 1.0kg to 2.5kg. The weight is the main reason why not even more powerful actuators are used for the Priscilla humanoid.

2.2 Elvis

ELVIS is a scale model of a full-size humanoid with a height of about 60cm, built with 42 servos giving a high degree of freedom in legs, arms and

[2] http://humanoid.chalmers.se
[3] http://www.ee.uwa.edu.au/~braunl/eyebot/

hands. Microphones, cameras and touch sensors guide the robot. The imminent goals are to walk upright and to navigate through vision serving a prototype for Priscilla. Seven on board micro-controllers control the servos and sensors. ELVIS is autonomous, with onboard power supply and main processing unit, but many experiments are mainly performed with connection to a host computer. ELVIS can for instance walk fully autonomously.

2.3 Elvina

ELVINA is a simplified, scaled model of a full-size humanoid with body dimensions that mirrors the dimensions of a human [15]. The ELVINA humanoid is a fully autonomous robot with onboard power supply and computer, but many experiments are performed with external power supply. It is 28cm tall and it weights about 1490g including batteries. Each of the two legs has 5 degrees of freedom, of which 4 DOF is active and 1 DOF is passive. The head, the torso and the arms has 1 DOF each, giving a total of 14 DOF.

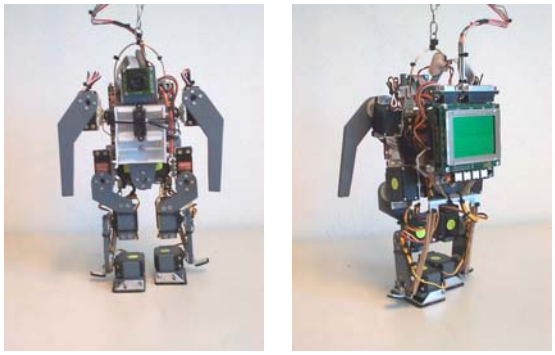

Figure 4. Pictures of the ELVINA Humanoid. Camera and PSD sensor visible (left) and controller board (right).

Vision is the most important sensor of this robot. Therefore, it is equipped with full color 24 bit digital camera, which is based on CMOS technology. The camera is directly connected to the controller board and mounted in the robot's head. The body also houses a near-infrared PSD (position sensitive detector) which is used to determine distances to nearby objects. In its present status, the robot is capable of static walking.

A single camera cannot be used to accurately measure the distance to a nearby object. This is instead achieved with the near-infrared PSD range sensor, which consists of an IR emitter and a position sensitive detector in a single package. The principle of this sensor is based on triangulation, which means that the sensor is relatively insensitive to the texture and color of the object at which it is pointed [14].

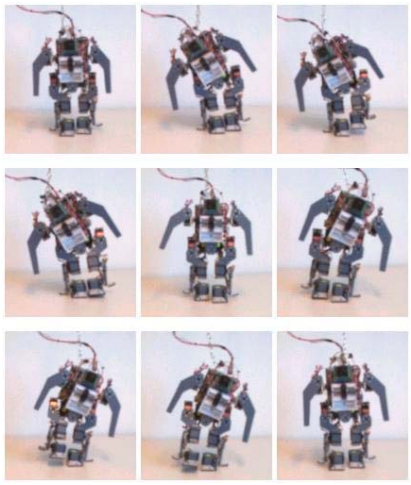

Figure 5. Series of pictures showing a complete gait cycle, from top left to bottom right.

In order to control the movements of a limb, the partial movements of all involved joints must be coordinated and synchronized to get the desired motion. For this reason, a servo locomotion module has been developed. The idea behind it is that twelve integer-valued vectors specify a complete gait cycle, each of the vectors specifying given positions of the robot's limbs. Each of the vectors consists in fact of a set of control parameters, a position value for each actuator and two time constants. The first of the vectors in the set correspond to the robot's initial position and the second vector corresponds to the second position of the robot and so on. By interpolation from the values of one vector to the values of another vector the robot's limbs is caused to smoothly move from one position to another position. The first time constant specifies how fast the limbs should move between consecutive positions and the other constant specifies time delay before the position of the limbs is updated. To obtain a complete gait cycle the set of vectors specifying it is interpolated once and the robot is made to continuously walk by iteration. All robot control programs that we have developed are implemented in C language.

3. Architecture

The philosophy behind all our robots is that the software architecture should mainly build on evolutionary algorithms and specifically genetic programming. Evolution is thus used to induce programs, functions and symbolic rules for all levels of control. Three hierarchical layers are used for control. Those are the reactive, the model building and the reasoning layer.

3.1 Reactive Layer

The first layer is a reactive layer based on on-line evolution of machine code. This method assumes that all fitness feedback is obtained directly from the actual robot. The disadvantage is that the GP individuals spend most of their time waiting for feedback from the physical environment. This results in moderate learning speed, and the constant movement shortens the life span of the hardware. The benefit of the method is its simplicity, and that the only constraints needed for the models being learned are that they should fulfil their task as a black box. This layer is used for reactive behaviors such as balancing.

3.2 Model Building Layer

To achieve higher learning speeds and more generic behaviour there is a second control layer that works with memories of past events. In this genetic reinforcement-learning framework, the system tries to evolve a model of the underlying hardware system and problem. The model maps sensor inputs and actions to a predicted goodness or fitness value. The currently best model is then used to decide what action results in optimal predicted fitness given current sensor inputs. This layer allows the genetic programming system to run at full speed without having to wait for feedback from the environment; instead it fits the programs to memories of past events. The machine code genetic programming approach used is called Automatic Induction of Machine Code GP, AIMGP [12]. AIMGP is about 40 times faster than conventional GP systems due to the absence of any interpreting steps. In addition, the system is compact, which is beneficial when working on board a real robot. The model-building layer is also used for basic control tasks.

3.3 Reasoning Layer

The third layer is a symbolic processing layer for higher "brain functions" requiring reasoning. The objective of this layer is to handle high level tasks such as navigation, safety and energy supply. This layer is built on "genetic reasoning", a method where evolution is used as an inference engine, requiring less heuristics to guide the inference procedure [12].

Each of these layers consists of modules for various tasks such as balancing, walking and image processing. Some system functions are represented as several modules spanning different layers.

4. Experiments

Several experiments has been performed on this architecture:

- Balancing, evolution of functions for balance
- Walking, evolution of efficient walking, described further in this paper
- Vision, evolution of 3-D vision
- Navigation, evolution of plans
- Audio orientation, evolution of stereo hearing
- Manipulation, evolution of eye hand co-ordination

In this section we describe an experiment with evolution of efficient walking.

4.1 Evolutionary Algorithm

The evolutionary algorithm used is a steady state evolutionary strategy [4], running on the robot's onboard computer. A population that stems from a manually developed individual is created with a uniform distribution over a given search range. Then four individuals are randomly selected from this initial population. These individuals are evaluated and their fitness is measured. The two individuals with the better fitness values are considered as parents and the two individuals with the lower fitness are replaced by the offspring of the parent individuals. The selection, evaluation and reproduction phases of the evolutionary strategy is then repeated until the maximum number of trials is reached.

The initial population is composed of 30 individuals with 126 genes randomly created with a uniform distribution over a given search range.

The search range for each parameter type (e.g. speed, delay and servo position) is determined from experience in manually developing gaits. The search range is defined as the magnitude of the Euclidean distance between a certain gene in the manually developed individual and the corresponding gene in a randomly created individual. The search ranges are set to suitable values in order to produce a sufficient amount of individuals in the population that are capable of good performance in the evaluation.

A tournament selection is used to select individuals for parents and the individuals to be replaced by their offspring. Four different individuals are randomly picked from the population and then evaluated one at a time. The two individuals who get the higher fitness are considered as parents and their offspring, produced by recombination and mutation, replaces the two individuals with the lower fitness in the population. The number of generations a certain individual can be selected to be in the tournament is unrestricted.

For reproduction both mutation and recombination is used. Recombination takes the two individuals considered as parents, p_1 and p_2, and creates two child individuals, c_{1i} and c_{2i}. Each gene of the child c_{ki} then gets the value

$$c_{ki} = p_{ki} + \alpha_{ki}(p_{1i} - p_{2i})$$

where c_{ki} is the ith gene of the kth child individual, p_{ki} is the ith gene of the kth parent individual, p_{1i} and p_{2i} are the ith gene of the two parents p_1 and p_2. The α_{ki} is a number randomly chosen to be either + 1 or -1.

In each of the child individuals produced, 20 % of the genes are mutated by a small amount. The genes in these two individuals are selected by random to undergo mutation and it is possible for a gene to be mutated several times. The gene to be mutated gets a value according to the equation

$$c_{ki,mutate} = c_{ki} + \delta_{ki} m_{ki}$$

where $c_{ki,mutate}$ is the mutated ith gene of the kth child individual, c_{ki} is the gene to be mutated. The δ_{ki} denotes a number randomly chosen to be either –1 or +1. The m_{ki} are a random number with uniform distribution that determines how much each gene should be mutated and it is set proportional to each parameter type‘s search range. That is, for the delay parameter, m_{ki} values are set to maximum 6% of its search range and for the speed and servo position parameters, m_{ki} values are, in a similar way, set to 33% maximum respectively.

4.2 Experimental method

The aim in short term for our experiments is to optimize a set of integer values, used as control parameters for a biped robot gait. They should move the robot faster, straighter and in a more robust manner than the previously manually developed set of parameter values did.

The robot is placed on top of a table with a surface of relatively low friction during the evolution. A target wall of 50cm height and white color is placed at one end of the table and to mark the center of that end there is a vertical black stripe on the wall. Right above the robot (65cm above the table surface) there is a horizontal beam, used as a carrier for the power supply cables.

Each individual evaluates under as equal conditions as possible. The robot‘s starting position is at a distance of about 40cm from the wall and facing it. The experimenter centers the robot according to the black line by using its onboard camera. Once centered, the robot measures its distance with the PSD infrared range sensor and starts to locomote towards the wall. After a fixed number of gait cycles it stops. Again it measures its distance from the wall and pans its head (camera) to search for the black line on the wall. Using these measurements and the time required for the locomotion trial, it calculate a fitness value for this actual individual. The robot is then manually reset to its starting position by the experimenter for the next individual to be evaluated.

The primary task for the onboard camera is to provide a precise tool for determination of direction. Initially, the camera is set to continuos image mode, whereas the frames are put to the EyeBot‘s LCD screen and thus made visible to the experimenter. The robot is considered as centered when the image of the black line appears near the center of the LCD display. A single snapshot is then analyzed by the software's image processing routines [16] to precisely determine the robot's position relative the black stripe. This measure is then stored for later use.

After an individual has performed a trial, the camera is again used to determine how straight the

robot moved during the trial. As the robot body remains fixed the camera pans from left to right. When the black line appears on the LCD display, the camera stops moving and the difference between this value and the earlier obtained measure is considered as the angular deviation θ from the desired (straight) path of locomotion.

While the robot uses its onboard camera for determination of direction, distances are measured using a near-infrared PSD range sensor located at the robot's chest.

To determine an individuals fitness score both its average velocity during the trial and its ability to move in a straightforward path is taken into account for. The fitness score function is defined as

$$score = v(d_0, d_f, t) \times s(\theta, d_f)$$

where $v(d_0, d_f, t)$ is the average velocity of the robot during the trial and $s(\theta, d_f)$ is the straightness function. The d_0 and the d_f denote the initial and the final distances to the target wall respectively and t is the time passed during the trial. The straightness function is dependent of both the angular deviation θ and the robot's final distance to the target wall and it is thus defined as

$$s(\theta, d_f) = \frac{d_f(f(\theta)-1)+150-10f(\theta)}{140}$$

Here, f (θ) is a normalization function to convert θ into a 0 – 1 measure. The values 150 and 10 are used as constants for the straightness function because they are raw values corresponding to the maximum and minimum measurable distances for the PSD sensor. The straightness function accounts for the robot's final distance from the black target strip - with the robot at a fixed orientation θ being larger when the robot stops closer to the target wall. Finally, the average velocity function is defined as

$$v(d_0, d_f, t) = \frac{d_0 - d_f}{t}$$

In the case when an individual does not maintain the robot's balance during a complete trial (e.g. the robot falls) it receives a zero fitness score.

4.3 Results

For this evolutionary strategies experiment we used an initial population of 30 individuals and ran for nine generations. The best-evolved individual received a fitness score of 0.1707. The manually developed individual was also tested and received a fitness score, averaged over three trials, of 0.1051. The best-evolved individual outperformed the manually developed individual both in its ability to maintain the robot in a straight course and in robustness, i.e. with a less tendency to fall over. The qualities of different individuals were also tested in other ways than direct fitness measuring. To evaluate one generation, consisting of four individuals, took approximately 30 minutes in this experiment.

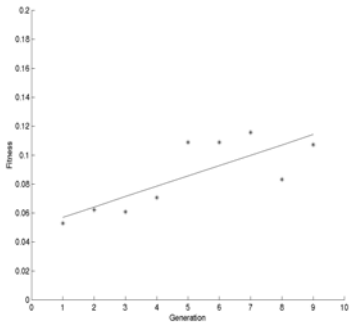

Figure 6. Plot of the average fitness scores

The above figure shows the average fitness scores for each generation as dots and the line is produced by statistical analysis, i.e. linear regression, of the dots. Since the slope of the line is positive, we observe a tendency towards better and better fitness values.

4.4 Discussion

Evolving efficient gaits with real physical hardware is a challenging task. That is because the mechanical structure of the robot is non-rigid. When moving a limb (e.g. a leg), the trajectory, thus the limb's final position, is affected by from which position the movement started. How much the torso leans also affects the resulting position of

the robot. The most vulnerable parts of the robot were proved to be the knee servos. Both these servos were replaced tree times. The torso and both the ankle actuators were exchanged once as well as the two hip servos.

5. Conclusions

The work presented in this paper constitutes of two main parts, the construction of a series of humanoid walking robots and a genetic programming experiment performed on the humanoids.

By manually developing locomotion module parameters, the robot was made capable of autonomous static walking in a first stage. In the next stage we performed a genetic programming experiment on the robot in order to improve the manually developed gait. For this, we used a steady state evolutionary strategy that was run on the robot's onboard computer. This algorithm evolved an individual that outperformed the previously manually developed set of parameter values in a sense that it moved the robot in a straighter path and in a more robust way.

References

[1] P. Nordin and M. G. Nordahl (1999). An evolutionary architecture for a humanoid robot. Proceedings of the Fourth International Symposium on Artificial Life and Robotics (AROB 4th 99). Oita, Japan.

[2] P. Dittrich, A. Burgel and W. Banzhaf (1998). Learning to move a robot with random morphology. Phil Husbands and Jean Arcady Meyer, editors, First European Workshop on Evolutionary Robotics (pp. 165—178). Berlin: Springer-Verlag.

[3] S. Nolfi and D. Floreano (2000). Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. Massachusetts: The MIT Press.

[4] W. Banzhaf, P. Nordin, R. E. Keller and F. D. Francone (1998). Genetic Programming~ An Introduction: On the Automatic Evolution of Computer Programs and Its Applications. San Francisco: Morgan Kaufmann Publishers, Inc. Heidelberg: dpunkt verlag.

[5] G. B. Parker and J. W. Mills (1999). Adaptive hexapod gait control using anytime learning with fitness biasing. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '99) (pp. 519-524). San Francisco: Morgan Kaufmann Publishers, Inc.

[6] N. Jakobi (1998). Running across the reality gap: octopod locomotion evolved in a minimal simulation. Phil Husbands and Jean Arcady Meyer, editors, First European Workshop on Evolutionary Robotics (pp. 39—58). Berlin: Springer-Verlag.

[7] T. Gomi and K. Ide (1998). Emergence of gait of a legged robot by collaboration through evolution. P. K. Simpson, editor, IEEE World Congress on Computational Intelligence. New York: IEEE Press.

[8] G. S. Hornby, S. Takamura, O. Hanagata, M. Fujita and J. Pollack (2000). Evolution of controllers from a high-level simulator to a high dof robot. J. Miller, editor, Evolvable Systems: from biology to hardware; proceedings of the third international conference (ICES 2000) (Lecture Notes in Computer Science; Vol. 1801 pp. 80-89). Berlin: Springer-Verlag.

[9] M. A. Lewis, A. H. Fagg and A. Solidum (1992). Genetic programming approach to the construction of a neural network for control of a walking robot. Proceedings of the IEEE International Conference on Robotics and Automation. New York: IEEE Press.

[10] G. S. Hornby, M. Fujita, S. Takamura, T. Yamamoto and O.Hanagata (1999). Autonomous evolution of gaits with the Sony quadruped robot. Proceedings of the Genetic and Evolutionary Computation Conference. San Francisco: Morgan Kaufmann Publishers, Inc.

[11] M. S. Gazzaniga (1999). The New Cognitive Neurosciences. MIT Press, ISBN: 0262071959

[12] J. P. Nordin (1997), Evolutionary Program Induction of Binary Machine Code and its Application. Krehl Verlag, Muenster, Germany

[13] G. S. Hornby, S. Takamura, J. Yokono, O.Hanagata, T. Yamamoto and M. Fujita (2000). Evolving robust gaits with AIBO. IEEE International Conference on Robotics and Automation. pp. 3040-3045.

[14] J. L. Jones, A. M. Flynn and B. A. Sieger (1999). Mobile Robots: Inspiration to Implementation. Massachusetts: AK Peters.

[15] J. Ziegler, K. Wolff, P. Nordin and W. Banzhaf (2001). Constructing a small humanoid walking robot as a platform for the genetic evolution of walking. Proceedings of the 1st International Conference on Autonomous Minirobots for Research and Edutainment, AMiRE 2001. Paderborn, Germany: The Heinz Nixdorf Institute..

[16] P. J. McKerrow (1991). Introduction to Robotics. Wollongong: Addison-Wesley.

Paper II

Learning biped locomotion from first principles on a simulated humanoid robot using linear genetic programming

In

Proceedings of the Genetic and Evolutionary Computation Conference, volume 2723 of LNCS, pages 495–506, Chicago, IL, USA, July 12-16, 2003.

Learning Biped Locomotion from First Principles on a Simulated Humanoid Robot Using Linear Genetic Programming

Krister Wolff and Peter Nordin

Dept. of Physical Resource Theory, Complex Systems Group,
Chalmers University of Technology,
S-412 96 Göteborg, Sweden
{wolff, nordin}@fy.chalmers.se
http://www.frt.fy.chalmers.se/cs/index.html

Abstract. We describe the first instance of an approach for control programming of humanoid robots, based on evolution as the main adaptation mechanism. In an attempt to overcome some of the difficulties with evolution on real hardware, we use a physically realistic simulation of the robot. The essential idea in this concept is to evolve control programs from first principles on a simulated robot, transfer the resulting programs to the real robot and continue to evolve on the robot. The Genetic Programming system is implemented as a Virtual Register Machine, with 12 internal work registers and 12 external registers for I/O operations. The individual representation scheme is a linear genome, and the selection method is a steady state tournament algorithm. Evolution created controller programs that made the simulated robot produce forward locomotion behavior. An application of this system with two phases of evolution could be for robots working in hazardous environments, or in applications with remote presence robots.

1 Introduction

Dealing with humanoid robots requires supply of expertise in many different areas, such as vision systems, sensor fusion, planning and navigation, mechanical and electrical hardware design, and software design only to mention a few. The objective of this paper, however, is focused on the synthesizing of biped gait.
The traditional way of robotics locomotion control is based on derivation of an internal geometric model of the locomotion mechanism, and requires intensive calculations by the controlling computer, to be performed in real time. Robots, designed in such a way that a model can be derived and used for controlling, shows large affinity with complex, highly specialized industrial robots, and thus they are as expensive as conventional industrial robots. Our belief is that for humanoids to become an everyday product in our homes and society, affordable for everyone, there is needed to develop low cost, relatively simple robots. Such robots can hardly be controlled the traditional way; hence this is not our primary design principle.

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2723, pp. 495–506, 2003.

A basic condition for humanoids to successfully operate in human living environment is that they must be able to deal with unpredictable situations and gather knowledge and information, and adapt to their actual circumstances. For these reasons, among others, we propose an alternative way for control programming of humanoid robots. Our approach is based on evolution as the main adaptation mechanism, utilizing computing techniques from the field of Evolutionary Algorithms.

The first attempt in using a real, physical robot to evolve gait patterns was made at the University of Southern California. Neural networks were evolved as controllers to produce a tripod gait for a hexapod robot with two degrees of freedom for each leg [6]. Researchers at Sony Corporation have worked with evolving locomotion controllers for dynamic gait of their quadruped robot dog AIBO. These results show that evolutionary algorithms can be used on complex, physical robots to evolve non-trivial behaviors on these robots [3] and [4].

However, evolving efficient gaits with real physical hardware is a challenge, and evolving biped gait from first principles is an even more challenging task. It is extremely stressing for the hardware and it is very time consuming [17]. To overcome the difficulties with evolving on real hardware, we introduce a method based on simulation of the actual humanoid robot.

Karl Sims was one of the first to evolve locomotion in a simulated physics environment [13] and [14]. Parker use Cyclic Genetic Algorithms to evolve gait actuation lists for a simulated six legged robot [11], and Jakobi et al has developed a methodology for evolution of robot controllers in simulator, and shown it to be successful when transferred to a real, physical octopod robot [7] and [9]. This method, however, has not been validated on a biped robot. Recently, a research group in Germany reported an experiment relevant to our ideas, where they evolved robot controllers in a physics simulator, and successfully executed them onboard a real biped robot. They were not able to fully realize biped locomotion behavior, but their results were definitely promising [18].

2 Background and Motivation

In this section we summarize an on-line learning experiment performed with a humanoid robot. However this experiment was fairly successful in evolving locomotion controller parameters that optimized the robot's gait, it pointed out some difficulties with on-line learning. We summarize the experiment here in order to exemplify the difficulties of evolving gaits on-line, and let it serve as an illustrative motivation for the work presented in the remainder of this paper.

2.1 Robot Platform

The robot used in the experiments is a simplified, scaled model of a full-size humanoid with body dimensions that mirrors the dimensions of a human. It was originally developed as an alternative, low-cost humanoid robot platform,

intended for research [17]. It is a fully autonomous robot with onboard power supply and computer, and it has 14 degrees of freedom.

The robot has the 32-bit micro-controller EyeBot MK3 [2] onboard, carrying it as a backpack. All signal processing, including control, vision, and evolutionary algorithm, is carried out on the EyeBot controller itself. In its present status, the robot is capable of static walking.

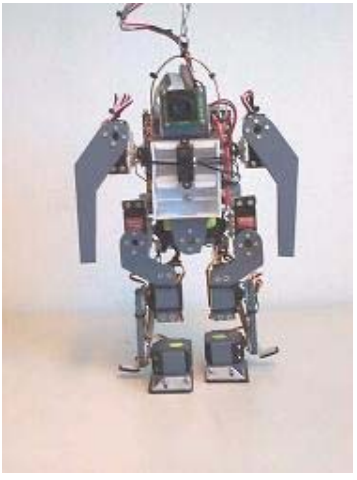

Fig. 1. Image of the real humanoid robot 'elvina'.

2.2 Gait Control Method

The gait control method for this robot involves repetition of a sequence of integrated steps. Considering fully realistic bipedal walk, two different situations arise in sequence: the statically stable double-support phase in which the robot is supported on both feet simultaneously, and statically unstable single-support phase when only one foot of the robot is in contact with the ground, the other foot being transferred from the back to front position. When this sequence of transitions has been repeated twice, one can consider a single gait cycle to be completed. That is, the locomotion mechanism's posture and limb's positions are the same after the completion as it was before it started to move, and hence it's internal state is the same.

If we now study only static walk, i.e. the projection of the center of mass of the robot on the ground always lie within the support polygon formed by feet on the ground, there is obviously a number of statically stable postures in between the internal state of the robot and it's final state, during completion of a single gait cycle. By interpolation between numbers of such, statically stable, consecutive states it is possible to make the robot to complete a single gait cycle. Then, by continually looping, biped gait is produced.

2.3 Evolutionary Gait Optimization Experiment

This experiment was performed in order to optimize a by hand developed set of state vectors, defining a static robot gait. The evolutionary algorithm used was a tournament selection, steady state evolutionary strategy [1] and [16] running on the robot's onboard computer. Individuals were evaluated and fitness scores automatically determined using the robots onboard digital camera and proximity sensor.
A population of 30 individuals, stemming from a manually developed individual was created with a uniform distribution over a given search range. The best-evolved individual and the manually developed individual were independently tested, and their performances were compared to each other's. The former one received a fitness score, averaged over three trials, of 0.1707, and the latter one, tested under equal conditions, got a fitness of 0.1051. Within this context, a higher fitness value means a better individual, and thus the best-evolved individual outperformed the manually developed individual both in its ability to maintain the robot in a straight course and in robustness, i.e. with a lesser tendency to fall over [17].

2.4 Observations

To run such an evolutionary experiment as described above span over several days, and requires manual supervision all this time. Between each generation of four individuals evaluated, the experiment was paused for about 15 minutes in order to spare the hardware and especially the actuators. The main reason for this is that the actuators accumulate heat when they are running continuously under heavy stress. They then run the risk of getting overheated and gradually destroyed. One way to handle this problem was, as mentioned above, to run the robot intermittent so that the servos maintain an approximately constant temperature.
Evolving efficient gaits with real physical hardware is a challenging task. During the experiments, the torso and both the ankle actuators were exchanged once as well as the two hip servos. The most vulnerable parts of the robot were proved to be the knee servos. Both these servos were replaced tree times.
Obviously there are a number of difficulties related with evolving biped walking behavior on a real, physical robot. In an attempt to overcome some of the problems, we want to use a physically realistic simulation of the robot. The central idea in this concept is to evolve control programs from first principles on a simulated robot, transfer the resulting programs to the real robot and continue to evolve efficient gait on the real robot. Of course, there will arise other problems applying this method, as simulation systems always imply some simplifications of the real world.

3 Evolution of Control Programs

Our primary goal is to utilize Genetic Programming [5] and [8] for evolving locomotion control programs from first principles for our simulated biped robot,

i.e. with no a priori knowledge for the robot on how to walk, information of morphology etc. The evolved programs take the robot's *current* internal state parameter values as input vector and return a vector predicting it's *next* internal state parameter values, in order to produce robust biped gait.

3.1 Dynamic Physics Simulation

The Open Dynamics Engine (ODE) is a free library for simulating articulated rigid body dynamics, developed by Russell Smith [15]. An articulated structure is created when rigid bodies of various shapes are connected together with joints of various kinds.
The robot model is qualitatively consistent with the real robot in the aspect of geometry, mass distribution, and morphology. See [17] for details of the robot. It consists of 12 actuated joints and 13 body elements. It is constructed with its mass concentrated to the main body elements, which in the real robot correspond to the servo actuators, batteries and computer. The plastic body parts, interconnecting the servos to each other, are not rendered in the simulation, since their mass is very low compared to the total mass.

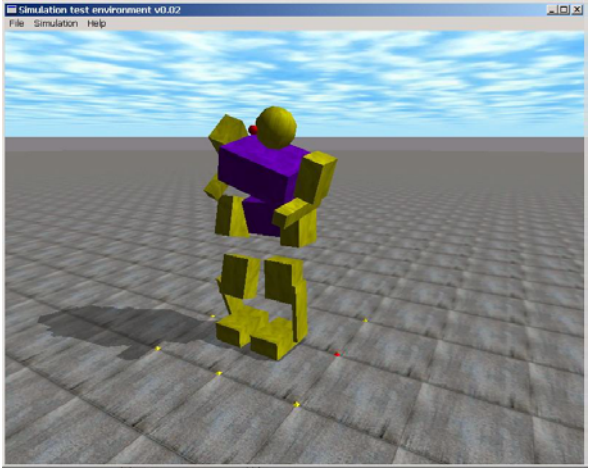

Fig. 2. Snapshot of the simulated humanoid robot. The body elements are directly connected to each other, although this is not visualized here.

3.2 Virtual Register Machine

The Genetic Programming representation used for this problem of robot control program induction is an instance of a Virtual Register Machine, VRM(k, l) [10].

It has k I/O registers and l internal work registers. In the current implementation of our system, l equals k. The function set consists in the present of arithmetic functions ADD, SUB, MUL, DIV, where DIV is protected division, and SINE. We now define a register state vector $\mathbf{Reg} \equiv [Reg_1, ..., Reg_k]$ of k integers, each of the elements corresponding to one of the actuated joints of the simulated robot. All program input/output is communicated through the states of the I/O registers. That is, program inputs are supplied in the initial state Reg, and output is taken from the final register state Reg'. Further, the I/O register state vector is initially copied into the internal work registers. We can do this in a straight forward manner, since we have imposed that the number of I/O registers, k, equals the number of work registers, l. The Virtual Register Machine is allowed writing only to the internal work registers when looping the program instructions. The I/O registers are write-protected in this phase, and their final state is updated after the end of the program execution cycle, before they are passed to the robot and then updating it's internal state.

3.3 Linear Genome Representation

Each individual is composed of simple instructions between input and output parameters. Each instruction consists of four elements, encoded as integers, and the whole individual is a linear list of such instructions:

```
 8, 22, 3, 12,
19, 11, 2, 16,
15, 12, 3, 12,
 8,  3, 4, 19,
12, 12, 4, 21,
 1,  6, 5, 12,
20,  3, 1, 19,
 9, 12, 2, 21,
23,  5, 3, 19,
16,  9, 3, 14,
13, 21, 5, 19,
 6, 13, 5, 14,
16, 22, 3, 16,
16,  3, 4, 18,
 8, 19, 2, 13,
20,  5, 3, 20,
13,  6, 1, 14,
```

The encoding scheme is as follows; the first and second elements of an instruction refers to the registers to be used as arguments, the third element corresponds to the operator, i.e. ADD=1, SUB=2, MUL=3, DIV=4, and SINE=5, and the last element is a register reference for where to put the result of the operation. The meaning of the first line (instruction) here is: multiply register 8 with register 22 and put the result in register 12. The operators take two arguments,

except when the operator is SINE, which of course only take one argument. In this case, the SINE operator is applied to the first element in the instruction, and the second element is simply discarded. A mutation on that element will thus have no effect on that individual's genotype. The register references 1-11 are assigned to I/O-registers, and register references 12-23 are assigned for the internal work registers. Parsing the individual above, and print out the first three instructions in 'C-style' looks like this:

```
Reg12 = Reg8 * Reg22;
Reg16 = Reg19 - Reg11;
Reg12 = Reg15 * Reg12;
```

3.4 Evolutionary Algorithm

At the beginning of the evolutionary process, the population is filled with randomly created individuals. The length, or number of instructions, of an individual is chosen randomly with Gaussian distribution, with expectation value 20. The maximum length is restricted to 256 instructions. The genes are created with a uniform distribution over their respective search range; 1-23 for the two first genes of an instruction, 12-23 for the last gene, and 1-5 for the third gene, which corresponds to the function set.
Our GP-system is a steady state tournament selection algorithm, with the following execution cycle:

```
1. Select four members of the population for tournament.
2. For all members in tournament do:
   a. Create an instance of the simulated robot.
   b. Record the position in 3d-space of all the robot's limbs.
   c. Execute the individual for 2500 simulation time steps.
   d. Record the final position of all the robot's limbs.
   e. Compute the fitness value (see below).
   f. Destroy the simulated robot.
3. Perform tournament selection.
4. Apply genetic operators on the winners to produce two children.
5. Replace the two losers in the population with the offspring.
6. Go to step 1.
```

The individuals are evaluated (evaluation cycle starting with point 2a. above) under identical conditions, since the simulation is entirely deterministic. They all start from the same standing upright pose, with the same orientation. The execution time for individuals are 2500 simulation time steps (corresponding to approx. 20 seconds of real time simulation), and if an individual cause the robot to fall before this time is completed, the evaluation is terminated. In the beginning of an experiment, a great majority of individuals are terminated before the intended time. Looping an individual once does *not* correspond to a single simulation time step, but to moving the robot's limbs between two consecutive internal states ('states' being referred to as in the subsection *Gait Control Method*).

Table 1. Koza style tableau, showing parameter settings for the evolution of locomotion control programs for the simulated humanoid robot.

Parameter	Value
Objective	Approximate a function that produce robust biped gait
Terminal Set	24 integer registers,
Function Set	ADD, SUB, MUL, DIV, SINE
Raw Fitness	According to eq. (2), scalar value
Standardized Fitness	Same as Raw Fitness
Population Size	800
Initialization Method	Random
Simulation Time	2500 simulation time steps
Crossover Probability	100%
Mutation Probability	80%
Initial Program Length	Gaussian distribution, expectation value 20.
Maximum Program Length	256 instructions
Maximum Tournament Number	None
Selection Scheme	Tournament, size 4
Termination Criteria	None (determined by the experimenter)

Fitness Calculation. As in all GP-applications, finding a proper fitness function that guides the artificial evolution in the desired direction is of great importance. The primary goal for the experiment was to produce a "human-like", bipedal gait without the robot falling. To accomplish this task, the individual controlling the robot should; (i) locomote the robot as straight forward as possible, and (ii) keep the robot in an upright pose during the movement. Hence, the proper measurements to feed the fitness function with are related to the height maintained by the robot, and the covered distance during simulation. Explicitly formulated in mathematical terms, the proper fitness function was found to be:

$$f = W\left[1.0 - \frac{h_{start}}{h_{stop}}\right] + (d_{left} + d_{right}) \tag{1}$$

where h_{start} is the height of the robot at the starting position, h_{stop} is the height when evaluation terminates (either the simulation is fully completed, or it is terminated before the intended time, caused by the robot falling). The height measure is applied to the position of the robot's head, however one could take the height of any body part. The second term is a measure of the distance covered by the robot during evaluation, applied to its feet. The robot is always starting with its feet in origo (in xy-plane). The first term will give a positive contribution to fitness if $h_{stop} > h_{start}$, negative contribution in the case when $h_{stop} < h_{start}$, and zero contribution if $h_{stop} = h_{start}$. Thus we have a fitness function rewarding forward locomotion and keeping the upright pose, and punishing backward movements and falling. The W in the first term is a weight,

scaling the mutual relation of rewarding and punishing. After some tweaking, it was found to work best when set to a value in the order of 10.

Genetic Operators. We use only two-point string crossover, with 100% probability for crossover, divided mutually on the rate 4:1 on homologous and non-homologous crossover.
When an individual is chosen for mutation, the mutation operator works by randomly selecting one single instruction from the individual, and make a change in the selected instruction. It makes that change either by changing any of the register references to another randomly chosen register reference from the register set, or the operator in the instruction may be changed. The probability for an individual to undergo mutation is 80%.

4 Results

When observing the experiments in run-time, it is compelling how quickly the simulated robot learns. In the first couple of hundred tournaments, a great majority of the individuals cause the robot to fall almost immediately in the beginning of the evaluation cycle, and the greater part of them tip over backwards. Maybe one out of ten individuals fall to the fore, which is a good starting point of taking a step ahead. Rather soon, however, one can observe the opposite situation, one out of ten individuals' overturn backwards and the rest fall ahead. This was not the desired goal for the evolution, but we regard this as being the first refined behavior that emerged.
The next observable stage of development in the evolution is when a large fraction of individuals is keeping the robot at a standstill, almost motionless, on its feet. In the beginning of our experiments, we faced some problems with evolution converged to this state. By increasing the population size and making some adjustments to the fitness function (mainly by decreasing the weight w, giving lesser punishment for tipping over), we could guide the evolution towards the desired goal. The mix of individuals showing this behavior, and individuals with a more 'energetic' behavior guarantee sufficient diversity of the population for evolution to proceed.
The final results of these experiment was indeed consistent with our initial objectives. That is, evolution created controller programs that made the simulated robot produce forward locomotion behavior. Some of the resulting programs made the robot walking forward in a spiral manner, with small movements, and others produced gait patterns with more lively movements. When tested, some of the individuals managed to keep the robot on its feet for the whole evaluation time (2500 simulation time steps), but when executed for a longer time, the robot usually ended up overturned. Nevertheless, a division of evolved programs could accomplish the task during the test run, without ever tipping over the robot.
Figures 3 and 4 displays some statistics from a representative run. In these experiments we did more than thirty independent runs, ranging from a few thousand

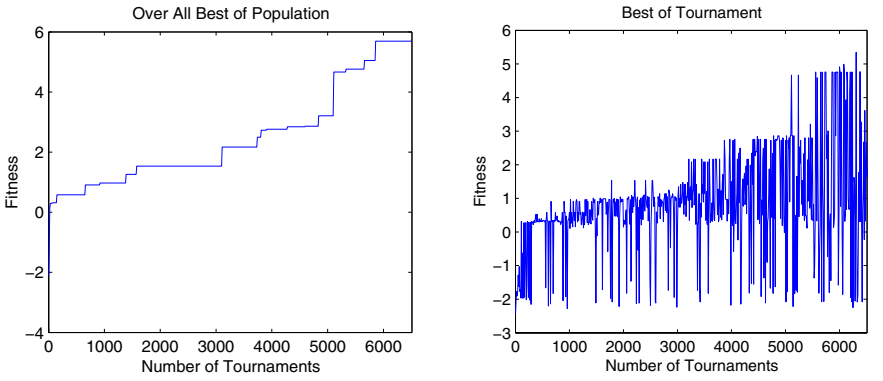

Fig. 3. a,Fitness value of over all best individual in the population (left) and **b**,fitness value of the best individual in every tournament (right).

tournaments, up to more than 80000 tournaments. The way fitness was defined (eq. 2), a fitness value < 0 correspond to the robot falling backward, and a small positive value (typically ranging from ~ 0.3 to ~ 0.6) correspond to the robot immediately falling ahead, while a value around 1.5 indicate a standstill. In figure 3a, one can observe how the best individual performed those behaviors; falling backward in the first few hundred tournaments, falling ahead in the first thousand tournaments, and standing still up to the 3000 tournaments. Fitness values in the range of ~ 1.5 to ~ 2.5 indicate some good locomotion, but usually ended up with the robot overturned, and fitness > 2.5 was successful locomotion behavior.

As depicted in figure 3a, the currently best individuals in the population showed progress from the beginning of the evolution and continued to develop over time. The program length typically decrease below the initialization length in the beginning of a run, but after a short while it starts to increase above that threshold, and finally it stabilize around some value. See figure 4. In all experiments we used the same initialization program length, with gaussian distribution and expectation value 20. It was observed that the program length, averaged over the whole population, did never go below the value 13, and never above 50, and it usually stabilized somewhere around 30.

5 Summary and Conclusions

We describe the first instance of an approach for control programming of humanoid robots. It is based on evolution as the main adaptation mechanism, utilizing computing techniques from the field of Evolutionary Algorithms. The central idea in this concept is to evolve control programs from first principles on a simulated robot, transfer the resulting programs to the real robot and continue to evolve efficient gait on the real robot. As the key motivation for using

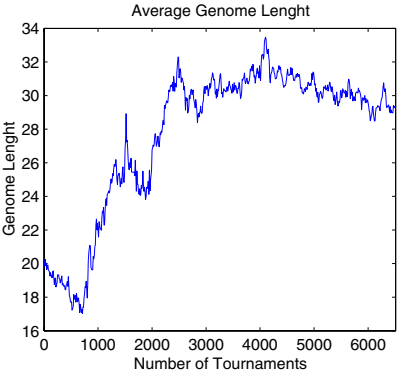

Fig. 4. Average genome length of all individuals in the population, length being defined as the number of instructions in an individual.

simulators, we briefly describe an on-line learning experiment performed with a biped humanoid robot.

The Evolutionary Algorithm is an instance of Genetic Programming, implemented as a Virtual Register Machine with 12 internal work registers and 12 external registers for I/O operations. The individual representation scheme is a linear genome, encoded as an array of integers. The selection method is a steady state tournament algorithm, with size four. The final results of these experiment was consistent with our initial objectives. That is, evolution created controller programs that made the simulated robot produce forward locomotion behavior. Current versions of the simulation system and the robot, however, do not allow the evolved programs to be directly downloaded to the robot. Further investigations and improvements are needed. To begin with, we must implement a subsystem of the simulated robot's control system and program interpreter on the real robots micro controller. Further, the real robot has an active feedback system, consisting of a color camera and a distance sensor, which will be implemented on the simulated robot as well. The development of the robot platform is an ongoing process, hence other sensors will be implemented on the robot. Then, the simulated robot should of course reflect all aspects, morphological and perceptual, of the real robot.
With this system of two phases of evolution, it will be possible to have a flexible adaptation mechanism that can react to hardware failures in the robot, e.g. if an actuator or sensor break down. By extracting information about malfunctioning parts and do off-line evolution with a modified model of the robot, it will become possible to react to the changes in the robot morphology. Another approach in this spirit, called Punctuated Anytime Learning, has been proposed by Parker [12]. For robots working in hazardous environments, or in applications with remote presence robots, this feature would be very useful.

References

1. Banzhaf, W., Nordin, P., Keller, R.E., and Francone F. D.: Genetic Programming An Introduction: On the Automatic Evolution of Computer Programs and Its Applications. San Francisco: Morgan Kaufmann Publishers, Inc. Heidelberg: dpunkt verlag. (1998)
2. Bräunl, T. 2002: EyeBot Online Documentation. Last visited: 01/21/2003. http://www.ee.uwa.edu.au/∼braunl/eyebot/
3. Hornby, G.S., Fujita, M. Takamura, S., Yamamoto, T., and Hanagata, O.: Autonomous evolution of gaits with the Sony quadruped robot. Proceedings of the Genetic and Evolutionary Computation Conference. San Francisco: Morgan Kaufmann Publishers, Inc. (1999)
4. Hornby, G.S., Takamura, S., Yokono, J., Hanagata, O., Yamamoto, T., and Fujita, M.: Evolving robust gaits with AIBO. IEEE International Conference on Robotics and Automation, New York: IEEE Press, pages 3040–3045. (2000)
5. Koza, J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA, USA: MIT Press. (1992)
6. Lewis, M. A., Fagg, A. H., and Solidum, A.: Genetic programming approach to the construction of a neural network for control of a walking robot. Proceedings of the IEEE International Conference on Robotics and Automation. New York: IEEE Press. (1992)
7. Lund, H., and Miglino, O.: From Simulated to Real Robots. Proceedings of IEEE 3rd International Conference on Evolutionary Computation. New York: IEEE Press. (1996)
8. Langdon, W. B., and Poli, R.: Foundations of Genetic Programming. New York: Springer-Verlag. ISBN 3-540-42451-2, 274 pages. (2002)
9. Miglino, O., Lund, H., and Nolfi S.: Evolving Mobile Robots in Simulated and Real Environments. Technical Report, Institute of Psychology, C.N.R., Rome. (1995)
10. Nordin, P.: Evolutionary Program Induction of Binary Machine Code and its Applications. Ph.D. Thesis, der Universität Dortmund am Fachbereich Informatik, Germany. (1997)
11. Parker, G. and Rawlins, G.: Cyclic Genetic Algorithms for the Locomotion of Hexapod Robots. Proceedings of the World Automation Congress, Volume 3, Robotic and Manufacturing Systems. (1996)
12. Parker, G.: Punctuated Anytime Learning for Hexapod Gait Generation. Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems. (2002)
13. Sims, K.: Evolving Virtual Creatures. Proceedings of Siggraph, pp.15–22. (1994)
14. Sims, K.: Evolving 3D Morphology and Behavior by Competition. Proceedings of Artificial Life IV, Brooks and Maes, editors, MIT Press, pp.28–39. (1994)
15. Smith, R.: Open Dynamics Engine v0.030 User Guide. Last Visited: 03/27/2003. http://opende.sourceforge.net/ode-0.03-userguide.html
16. Schwefel, H. P.: Evolution and Optimum Seeking. New York, USA: Wiley. (1995)
17. Wolff, K., and Nordin, P.: Evolution of Efficient Gait with an Autonomous Biped Robot using Visual Feedback. Proceedings of the Mechatronics Conference. University of Twente, Enschede, the Netherlands. (2002)
18. Ziegler, J., Barnholt, J., Busch, J., and Banzhaf W.: Automatic Evolution of Control Programs for a Small Humanoid Walking Robot. 5th International Conference on Climbing and Walking Robots. (2002)

Paper III

Evolution of biped locomotion using linear genetic programming

Submitted to

Autonomous Robots.

Evolution of Biped Locomotion using Linear Genetic Programming

Krister Wolff

Department of Applied Mechanics, Chalmers University of Technology
41296 Göteborg, Sweden

Abstract

The development of anthropomorphic bipedal locomotion is addressed by means of artificial evolution using linear genetic programming. The proposed method is investigated through full rigid-body dynamics simulation of a bipedal robot with 26 degrees of freedom. Stable bipedal gait with a velocity of 0.054 m/s is realized. Locomotion controllers are evolved from first principles, i.e. the evolved controller does neither have any a priori knowledge on how to walk, nor does it have any information about the kinematics structure of the robot. Instead, locomotion control is achieved based on intensive use of sensor information. Also, the energy consumption of the robot is monitored during simulation, in order to yield a pressure on evolution to favor energy efficient gaits. In this linear genetic programming approach, randomly generated individuals undergo structural evolution. This is, to our knowledge, the first example of entirely model free evolution of bipedal gaits for a system with high number of degrees of freedom.

1 Introduction and motivation

There are numerous application areas for robots with anthropomorphic shape and motion capabilities. In a world where man is the standard for almost all interactions, such robots have a very large potential acting in environments created for people. They can function in certain areas which are not accessible for wheeled robots, such as stairways or natural terrain, for instance. Furthermore, robots capable of bipedal locomotion have the ability to interact with the environment using the whole body, and climb over large obstacles in its path.

From a control theory point of view however, bipedal walking robots are more difficult to deal with than wheeled robots. Wheeled robots are designed to maintain their wheels in contact with the ground at all times. Thus, stability is usually not an issue, except when the robot is on a steep slope. Legged robots, on the contrary, lift their feet off the ground to walk. Thus, the motion of walking dynamically changes the stability of the robot.

In bipedal walking a complete cycle is divided into two phases; the *single support phase* and the *double support phase*, taking place in sequence. During the single support phase one foot is in contact with the ground and the other foot is in swing motion, being transferred from back to front position. The so called *support polygon* is formed by the robot's stance foot in the single-support phase, and by the robot's feet in the double support phase. Further, bipedal locomotion is commonly divided into two main classes; *static gaits*, and *dynamic gaits*. The most fundamental issue of bipedal locomotion, apart from the realization of the relative motions of the mechanism's links, is how to preserve the walking balance in the system [41]. That question has long been the main topic of many scientific studies, and some of the more influential results will be briefly discussed in this section.

In the research literature on topics related to bipedal walking, the terms *stability*, *equilibrium*, and *balance* are often used interchangeably. Throughout this paper, we will use the following notions in order to avoid confusion; the term *stability* refers to a system which could be analytically

treated as stationary (i.e. the static case), whereas for a non-stationary system (the dynamic case), the terms *balance* and *equilibrium* are used.

By definition, in static walking the robot is restricted to move in such a way that, in order to avoid tipping over, the vertical projection of the centre-of-mass (PCOM) of the robot remains within the convex envelope of the support polygon at all times. Further, it is assumed that the robot's motions are so slow that dynamical effects, which could arise due to the amount of torque that is applied to the robot over time, could be neglected. Thus, the system's stability depends solely on the PCOM, as described above [24]. Such a gait is also referred to as a *statically stable gait* [16]. However, the resulting gait is usually too slow for practical use in real bipedal robots [8].

A dynamic gait is any gait that is not statically stable at all times [36]. In dynamic walking, in fact, the above described PCOM condition has very little to do with the equilibrium of the bipedal system; the PCOM is allowed outside the support polygon. Instead, dynamical effects which arise due to the link's velocities and accelerations are heavily involved for maintaining the walking balance [24]. Walking with dynamic balance provide higher locomotion speed and greater efficiency than walking with statically stable gaits, but the control problem of dynamic walking is harder [24].

Within the domain of dynamic walking, the idea of the centre-of-mass projection has got its generalized equivalents, such as the zero-moment point (ZMP) [43, 42, 39, 41, 40], and the foot rotation indicator (FRI) point [16], in order to deal with the walking balance problem.

The ZMP concept was introduced by Vukobratovic *et al.* around 1969 [43, 42], but the term ZMP was formally introduced a couple of years later [44, 39]. During the single-support phase only one foot is in contact with the ground, while the other is in swing motion. In order to then maintain dynamic equilibrium of the bipedal mechanism, the *ground reaction force* (GRF) should act at the appropriate point on the foot sole (of the stance foot) to balance all forces acting on the system during the motion. It should be clear that the boundary condition for dynamic equilibrium states that the torques acting around the horizontal axes (x and y), at the point where GRF is acting, will always be equal to zero. There may exist a torque around the vertical axis, but it is a realistic assumption that it is balanced by frictional forces (given that the friction coefficient is high enough), and it will not cause foot motion. Thus, the ZMP is defined as the contact point between the ground and the foot sole of the supporting leg where the torques around the horizontal axes, generated by all forces acting on the robot, are equal to zero. Any change in the walking dynamics will cause a simultaneous change of the GRF vector, altering its magnitude, direction, and acting point ZMP. During a dynamically balanced gait, the ZMP can move only within the supporting area. In the double-support phase, the positions of both feet are fixed relative to the ground, but also in this situation the ZMP should remain within the convex envelope of the support polygon. This is the fundamental meaning of the ZMP concept [41].

The ZMP concept has been involved in numerous practical applications of anthropomorphic locomotion mechanisms. The first one was the realization of a dynamically balanced bipedal gait in 1984, with the WL-10RD robot at the Waseda University in Tokyo [37].

An extension to the ZMP conception, named the FRI point, has been suggested lately [16]. In the case of a dynamically balanced gait, the FRI point may exit the physical boundary of the support area, which the ZMP never does. However, researchers in the field still disagree about the notions and interpretations of the ZMP and the FRI point. Generally accepted definitions of, and means of treating dynamic equilibrium in bipedal walking still remain elusive [16, 41, 24, 17, 40, 1]. Moreover, Goswami [17] have recently introduced an additional concept, the zero rate of change of angular momentum (ZRAM), in order to deal with balance maintenance in dynamic gaits. However, it is beyond the scope of this paper to further examine which one is the most adequate concept.

Conventionally, there are two main approaches to bipedal gait synthesis; *off-line trajectory generation* and *on-line motion planning* [45, 24]. The first approach, the off-line method, assume that there exists an adequate dynamic model of the robot and its environment, for the generation of periodic walking pattern [26, 21, 20, 18, 48, 37]. In order to keep the ZMP within the stable region, as described above, a desired ZMP reference trajectory is generated. The reference trajectory can for instance be generated

using *spline functions* [2]. Then, a body motion that adheres to the desired ZMP trajectory is derived, using knowledge about the inverse kinematics relationships and the system's dynamics. Only if the ZMP stays within the stable region, the generated gait can be implemented on the actual robot.

The second approach uses limited knowledge about the kinematics and dynamics of the robot and its environment. In on-line controller design, this is referred to as finding the *plant model* and *transfer function* of the system one wishes to control. A plant model describes the static relationship between input and output, and a transfer function includes dynamic effects as well. Given the transfer function, appropriate torques can then be generated by means of e.g. a PID regulator. This type of control scheme relies much on the feedback. Control methods for bipedal walking based on the dynamical equations of motion [15, 13] have been proposed. In order to decrease the demand for computational resources in real-time implementations, simplified models, such as the inverted pendulum model [23, 30], and a static motion strategy [49], have been proposed. With on-line methods higher walking robustness is achieved at the cost of an increase in demand for computational resources, compared with off-line methods.

Control policies based on classical control theory, like the ones outlined above, have been successfully utilized for bipedal locomotion in various implementations. However, the success of these methods relies on the calculation of reference trajectories for the robot to follow. That is, trajectories of joint angle, joint torque, or the centre-of-mass of the robot are calculated so as to satisfy constraints, such as the ZMP criterion [18, 2, 48, 13, 30, 12]. When the bipedal agent, either it is a real physical robot or a simulated equivalent, is acting in a well-known environment the abovementioned control methods should work well. However, the drawback of using such a method when acting in a realistic, dynamically changing environment is that reference trajectories can rarely be specified there. A bipedal robot will encounter unexpected situations in the real world, which cannot all be accounted for on beforehand. Therefore, alternative biologically inspired computational methods have been considered when generating gaits and other behaviors for bipedal robots. Such methods do not, in general, require any reference trajectory.

Some researchers have employed a connectionist approach, i.e. artificial neural network (ANN) based strategies, for control of bipedal walking. Katic and Vukobratovic have reviewed such methods in a recent paper [24]. Others, including the authors of this paper, advocate the use of evolutionary algorithms (EAs) for bipedal gait synthesis, e.g. [2, 8, 31, 7].

Although many works have been published on natural bipedal gait generation by means of using EAs, and in particular genetic algorithms (GAs), most of them concerned on a simplified simulation model of the biped, i.e. the 5-link model developed by Furusho and Masubuchi [14]. Such a simplified model might be useful for illustrating a gait generation method, but it has a severe limitation; only motion in the sagittal plane is considered. Of course, the impact of the proposed technique would have a greater impact if demonstrated in a 3-dimensional rigid-body simulation instead. There exist, however, some examples in the research literature of synthesizing locomotion for full rigid-body simulated bipeds, by incorporating EAs.

Pettersson [31] have reported on the development of a method for generating walking behaviors for bipedal robots. An adaptation of evolutionary programming (EP) to the case of finite state machines (FSMs) is used to operate both on the structure and on the parameters of the robotic brain. The method has been demonstrated on a simplified five link robot, constrained to move in the sagittal plane. Two test cases were used; energy optimization and robust balancing. In the case of energy optimization, a 134% improvement in walking length was obtained. In the case of robustness, FSMs evolved that could withstand some perturbations, which the initial FSMs could not. The authors have also recently reported ongoing research with evolution of RNNs, used as CPGs, for balancing behavior of a simulated bipedal robot [32].

An example of using a GA to optimize the operational parameters of central pattern generators (CPGs), which were used to generate a walking pattern, was reported by Mojon [27]. Here, the model for the CPGs was based on the equations of the harmonic oscillator. A very realistic bipedal model was used. It was a model of a real physical humanoid robot, namely the QRIO[1] robot, developed by the

[1] http://www.sony.net/SonyInfo/QRIO/top_nf.html

Sony Corp. Sony's QRIO has 25 DOF, but the simulated model had "only" 21 DOF. The outcome of these simulations was that the simulated robot could walk in a straight line in a speed-equivalent of 1.5 km/h. This should be compared with the real QRIO robot that can walk in a speed of 2.5 km/h, on a flat surface. Also, the evolved gait did not look very realistic at all.

Recently, a research team from Australia used a standard GA with fitness proportionate selection scheme to evolve the parameters of a limited spline control system [4]. They managed to evolve a cyclic walking pattern, which could generate a slow forwards walk.

Reil and Massey [35] and Reil and Husbands [34] have used GAs to optimize fully recurrent neural networks (RNNs), used to control biped walking. They used a real-valued encoded GA to optimize weights, time constants, and biases in fixed architecture RNNs. Regarding the network's architecture; the fixed number of neurons was 10, and node 1 through 6 was considered as motor neurons. Connections between neurons could be removed, in the sense that its corresponding weight was set equal to zero. Their biped model has six DOF, and it consists of a pair of articulated legs connected with a link. Reil et al. reported results capable of biped walking in a straight line on a planer surface, without the use of proprioceptive input. Furthermore, simple sensory input to locate a sound source was integrated to achieve directional walking.

Ok et al. [29] have applied genetic programming (GP) to induce parts of a "nervous system" for a 3d simulated bipedal robot. The neural system represents a rhythm generation mechanism with CPG, consisting of neural oscillators, and a global feedback network. The authors have assumed that parameters and structures within single neural oscillators are known and fixed, and focused on creation of the feedback networks between the neural system and the body dynamics system. They managed to evolve a global feedback network which could generate bipedal gait, but only 4 steps of walking could be achieved with the simulated biped.

Using a genetic programming (GP) approach, Ziegler et al. [51] evolved gait controllers for a full rigid-body simulated bipedal robot. Their GP-system used linear genomes as representation. The best control program that emerged was capable of moving the biped forward by making fast, small steps.

The idea to utilize EAs for generating gait controllers for bipeds is not new, as shown by these examples. However, there exist only a few examples, to our knowledge, where one goes beyond parametric optimization, and optimize also the structure of the gait controller. Then, a more flexible scheme than the GA binary representation is required, e.g. GP [25].

While GAs usually operates on fixed-length binary strings, GP deals with the evolution of syntactically correct computer programs. The ordinary representation scheme in GP is called a *tree representation*. A GP tree consists of *functions* and *terminals* which, assembled into a structure, can be executed as a computer program. A terminal provides a value to the system, while functions process values contained in the system. The tree representation is a very flexible program structure. The set of functions can be very rich; any programming construct in any programming language may be used. In GP, there is much freedom to choose which functions to include. Further, the *size* of the GP-trees are allowed to vary during evolution.

Another sometimes used representation in GP is the *linear* representation, and consequently the variant of GP using that representation is referred to as linear genetic programming (LGP) [6]. LGP evolves sequences of *instructions* of an imperative programming language, e.g. C language or FORTRAN. LGP is, compared with tree-based GP, relatively easy to implement. Unlike tree-based GP, LGP also facilitates the use of *multiple program outputs*. This makes LGP ideally suited for the task of evolving controller programs for bipedal walking.

In this work, LGP is utilized to generate locomotion controllers from first principles for a simulated bipedal robot with 26 degrees of freedom (DOFs). It is noteworthy that in this robotic system there is no model of the bipedal system, neither any a priori knowledge on how to walk, available to the evolved controllers. Randomly generated individuals, or controllers, undergo evolutionary processes on the structural level. This is, to our knowledge, the first example of an entirely model free evolution of bipedal locomotion for a system with a high number of DOFs.

However, it should also be noted that learning such a complex task as bipedal locomotion from

scratch, starting with an empty brain and an already developed body, is far from trivial. For living animals in the nature, including humans, the body and the brain develop together. In this approach, the body of the bipedal does not undergo any evolutionary process. Still, this project is an attempt to generate a *robust* and *anthropomorphic* (i.e. human-like) bipedal gait by means of artificial evolution.

The rest of this paper is organized as follows: in Section 2 the physical simulation environment and the biped are described, together with the LGP system. Then, the simulation setup is described in Section 3. In Section 4 the results of the simulations are described, and in Section 5 the results are discussed. Finally, the paper ends with Section 6 Future work.

2 Method

In this section the methodology for the rigid-body simulations and the evolutionary method used here will be described. First, the physical simulation environment and the biped model are described. Second, the LGP system is explained.

2.1 Physical simulation

For simulating the articulated rigid-body dynamics the Open Dynamics Engine[2] (ODE) library is utilized. In ODE the equations of motion are derived from a Lagrange multiplier velocity-based model, and a first order integrator is used. The actual implementation of ODE puts emphasis on speed of execution, rather than on accuracy. It is therefore mainly used for qualitative engineering tasks involving rigid-body dynamics simulation.

2.1.1 Robot model

The biped model used here has no original equivalent design in the real world, but it could be seen as representing a generic bipedal robot model. It was created from the *body* and *joint* primitives available in the ODE simulation package. The robot model consists of 22 rigid-body parts, and 18 ODE joints. There are 8 *universal* joints, and 10 *hinge* joints used to connect the rigid-body parts into an articulated rigid-body structure (*universal* and *hinge* are the internal names of specific joint types in ODE). The rigid-body primitives used are 9 *capped cylinders* and 13 rectangular *boxes*. The robot's structure is defined using multiple chains, starting from its feet with each link described in terms of the previous links. This composition results in a 26 DOF bipedal model.

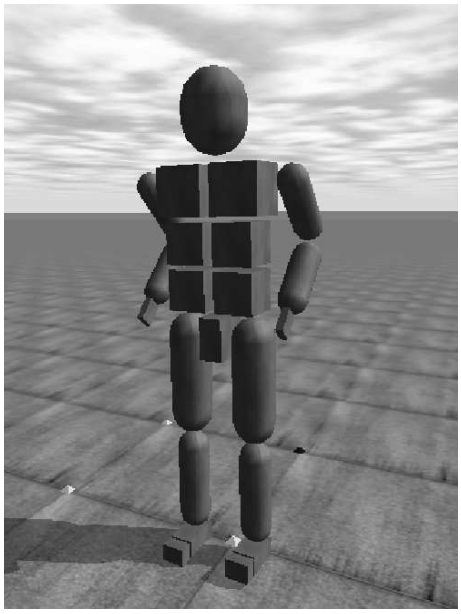

Figure 1: The biped model used.

2.1.2 Sensor feedback

Sensors monitoring the internal state of the robot, such as joint angles are referred to as *proprioceptive sensors*. In this setting, the *current joint angles* of the previous time step of the simulation are used by the evolved controller to compute the next set of motor signals for the robot.

Simulating a biped robot in a realistic environment most likely requires feedback loops between the robot's control system and the robot's body, as well as between the control system and the environment. The set of external sensors constitute the robot's "window" to the environment. Those sensors can measure quantities such as the robots acceleration or inclination relative a fixed coordi-

[2] http://ode.org/

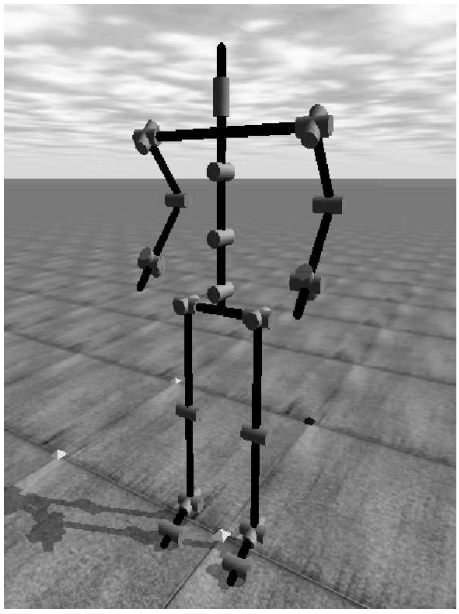

Figure 2: Schematic picture of the biped, showing its kinematics structure. The model consists of 19 links (black) interconnected by 18 joints (yellow), all in all resulting in 26 degrees of freedom.

nate frame, light intensity, external forces applied to the robots body, and sound waves, to mention a few examples.

A rigid body has six DOFs; three translational, and three rotational DOFs. In robotics, accelerometers are utilized to measure linear movement, and gyroscopes are used to measure rotations. To fully keep track of the movement of an object, tree accelerometers and three gyroscopes is sufficient. Mounted together, these form an inertial measurement unit (IMU). In this set of simulations, the robot was equipped with an *artificial balancing sense*, in the form of a virtual IMU in its head, and a three axis accelerometer in each foot. The linear accelerations and angular rates are obtained directly from the rigid body simulation in every time step.

2.1.3 Controller Model

Joints can be powered in ODE. i.e., for each joint in the bipedal model used here, there is a motor associated with it. The ODE package supplies a method, used in this investigation, to control the joints by simply setting a desired speed of the motor and a maximum force or torque that the motor will use to achieve that speed. Thus, in order to make the robot walk appropriate torque values need to be applied to the joints in each time step of the simulation.

In this implementation the speed and torque values have been pre-set, and the output of the evolved controller just sets the rotational direction (+) or (−) for each joint in the simulation. Appropriate values for *velocity* and *torque* were determined empirically. The generated controller takes as input the relative positions of the joints, i.e. the joint angles, additional sensor readings, and constants.

Appropriate motor signals are generated from the raw output of the evolved individual by means of a modified signum function ξ. These signals are then sent to the robot for execution, see Fig. 3 for more details. The modified signum function is defined as follows:

$$\xi(x) = \begin{cases} -1 & \text{if } x < \kappa \\ +1 & \text{if } x > \kappa \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The value of the parameter κ was set to 0.12 in the simulations. Furthermore, updated motor signals are sent to the robot at a constant time interval, covering a fixed number η of simulation time steps. The reason for not letting the motor signals be updated in each simulation time step was to avoid that joints could change their rotational direction in every time step, which would have resulted in rapid oscillations of the joints. Such movements are not desirable.

2.2 Linear genetic programming

The EA variant used here is referred to as linear genetic programming (LGP), and it follows the paradigm of genetic programming of automatic induction of syntactically correct computer programs. LGP evolves sequences of instructions of an imperative programming language, e.g. C language, FORTRAN, or machine code [28]. The instructions of LGP are restricted to operations (and conditional operations) that accept a minimum number of constants or memory variables, called *registers*, and assign the result to a register

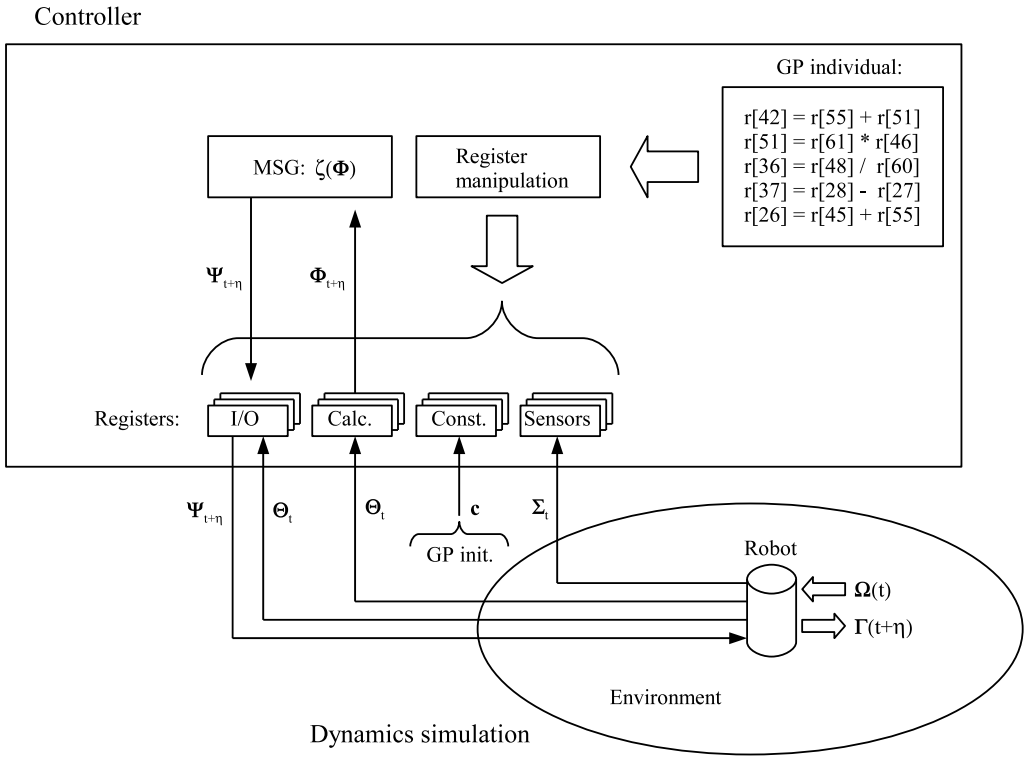

Figure 3: Schematic depiction of the information flow through the robot control system. At the discrete timestep t, the robot receives perceptual input $\mathbf{\Omega}(t)$ through its sensor channels. The sensor signals $\mathbf{\Sigma}_t$, are then fed into the *sensor registers* ($r_{55} - r_{66}$). Simultaneously, the robot's current joint angle positions $\mathbf{\Theta}_t$ are recorded in the *I/O-* and *calculation registers* ($r_0 - r_{25}$ and $r_{26} - r_{51}$ respectively). There is one register of each of these types associated with each degree of freedom. At the beginning of the GP run, the *constant registers* ($r_{52} - r_{54}$) were supplied with the values 0.001, 0.01, and 0.1 respectively. The VRM then executes the GP-individual (program), which manipulates the contents of the calculation registers. The I/O- sensor- and constant registers are *read only* during this stage. When the program execution has terminated (i. e. evaluation steps off the end of the program), motor signal generation (MSG) is initiated; a modified signum function ξ operates on the final contents $\mathbf{\Phi}_{t+\eta}$ of the calculation registers, and supplies the output $\mathbf{\Psi}_{t+\eta}$ to the I/O registers. These motor signals are then sent to the robot, which then in timestep $t + \eta$ execute these motor commands and accomplish some actions $\mathbf{\Gamma}_{t+\eta}$.

again, e.g. $r_0 := r_1 + 1$. In this section, the LGP concept will be explained in more detail.

2.2.1 Evolutionary algorithm basics

In *The Origin of Species* [10], it was argued that all existing organisms are the descendants of a few simple ancestors that arose on Earth in the distant past, and that the driving mechanism behind this evolutionary development was natural selection. Over the past 35 years or so, the principle of natural selection has been successfully utilized on computers to optimize a solution towards a pre-defined goal, and from this evolutionary algorithms (EA) have developed. Several research subfields, such as evolutionary programming (EP) [11], genetic algorithms (GA) [19], evolution strategies (ES) [33], and genetic programming (GP) [25] have emerged. Although they differ from each other in many aspects they all mimic natural evolution in

some ways. Based on very simple models of organic evolution, EAs are applied to various problems such as combinatorial optimization problems or learning tasks.

The main ingredients common for all types of EA are *population of solution candidates*, *variation operators*, *conservation operators*, *quality differentials*, and *selection methods* [3]. The first decision that has to be made, however, when designing an EA in order to solve a problem is how to define the *representation* of a solution.

In GAs solution candidates are represented as fixed-length strings of zeros and ones, and in ES the individuals are represented as vectors of continuous variables. In EP, the representation scheme consists of collections of finite state machines. In GP there are three principal representation structures used, which are called *tree*, *linear*, and *graph*. An assembly of solution candidates, or individuals, are simply called a *population*.

The *variation operator* ensure that new aspects of a problem are considered. In EAs this operator is called *mutation*. It comes with three EA parameters determining its strength, its spread, and its frequency of application. A very strong mutation operator would basically generate a random parameter at a given position within a solution. If applied to all positions within a solution, it would generate a solution completely uncorrelated with its origin, and if applied with maximum frequency within the entire population, it would erase all information generated in the population so far. Variation operators are means to increase the diversity in the population of solution candidates.

The *conservation operators* are used to consolidate what has already been learned by various individuals in the population, i.e. conservation operators are means to decrease the diversity. *Recombination*, or *crossover* of two or more solutions is the primary tool for achieving this goal. Provided the different parameters in a solution representation are sufficiently independent from each other, combinations of useful pieces of information from different individuals would result in better overall solutions. There are different ways to achieve a mixing of solutions. The most frequently used method involves two individuals recombining their information, although multi-recombinant methods are also used. Common recombination methods are one-point, two-point, or n-point crossover for discrete parameter values between two individuals, as well as intermediate recombination for continuous parameter values. A subset of the population can also be copied without change to the next generation, i.e. *reproduction*.

An important property of an EA is to have some *quality differentials*, i.e. a graded *fitness function* that distinguishes a better solution candidate from a good one, or a bad candidate from a really bad one. If there was only a binary fitness function, stating a solution by "1" and no solution by "0", then there would not be enough diversity among individuals to drive the evolutionary search process.

Since the population is finite, not all the individuals generated by the EA can be stored. Both the solution candidates to be included in further evolution and the ones to be replaced from the population are selected on the basis of the quality differentials. Following Darwin's tradition, this procedure is called *selection* [3]. The most commonly used selection method is *fitness-proportional selection*, which stems from the GA community. In GP *tournament selection* is also widely used. A generic evolutionary algorithm is summarized in Algorithm 2.1 below.

Algorithm 2.1 (Generic EA)

1. Randomly *initialize* a population of solution candidates.
2. By using a certain selection method, *select* individuals that are fitter than others. The fitness measure defines the problem that has to be solved by the algorithm.
3. *Generate new individuals* by applying one or more of the genetic operators reproduction, recombination, and mutation.
4. If the *termination criterion* is not met, go to 2.
5. *Stop*. The best solution found is represented by the fittest individual.

□

2.2.2 LGP fundamentals

When dealing with the actual implementation of LGP, the concept of the so called *virtual register*

machine (VRM) [22] is useful. The term is derived from a hypothetical model of computation, called a register machine [38].

A particular instance of a virtual register machine, VRM-$\mathcal{M}_{m,n}$, consists of a finite set of registers $R = \{r_1, ..., r_m\}$, each of which can hold a floating point value, a finite set of instructions $I = \{I_1, ..., I_n\}$. Such a register machine, implemented in a high level imperative language, is referred to as "virtual" because it has to be interpreted by software. The registers constitute the machine's mutable memory, and all program input and output is communicated through the registers. Additionally, registers may also be used to supply the program with constants, which can be synthesized in otherwise unused registers. Therefore, it seems natural to define a *register state* $\mathbf{S_r}$ as a vector of m floating point values

$$\mathbf{S}_r \equiv (r_1, ..., r_m)$$

Program inputs are supplied in the initial state $\mathbf{S}_{r,i}$ and outputs are taken from the final register state $\mathbf{S}_{r,f}$. Beside the required number of *input registers*, additional registers can be provided in order to facilitate calculations. These so called *calculation registers* (sometimes also referred to as *internal work registers*) are normally initialized with a constant value each time before a program is executed on the fitness cases. The VRM-$\mathcal{M}_{m,n}$ is allowed only to write to the calculation registers when executing the list of instructions (program). Thus, the input registers are write protected during this phase.

Finally, one or more registers may be defined as *output register(s)*, either among the calculation registers or among the input registers. The LGP structure facilitates the use of multiple program outputs. By contrast, functional expressions like trees calculate one output only [6].

In addition to the external register state, VRM-$\mathcal{M}_{m,n}$ maintains an internal state: the *program counter*, PC. The program counter is an integer that selects which instruction to fetch and execute. Branch instructions modify the PC to point to the branch's target; all other instructions always increment the PC to point to the next instruction. By definition, in LGP a *program* $\mathbf{P}$ is a vector of n instructions

$$\mathbf{P} \equiv (I_1, ..., I_n)$$

Table 1: Instruction Set

Instruction type	*General notation*	*Input range*
Arithmetic operations	$r_i := r_j + r_k$ $r_i := r_j - r_k$ $r_i := r_j \times r_k$ $r_i := r_j / r_k$	$r_i, r_j, r_k \in \mathbf{R}$
Trigonometric functions	$r_i := \sin r_j$	$r_i, r_j \in \mathbf{R}$
Conditional branches	$if(r_j > r_k)$ $if(r_j \leq r_k)$	$r_j, r_k \in \mathbf{R}$

The program counter PC naturally corresponds to an index i, of $\mathbf{P}$. A program terminates when $PC = n$, i.e. when evaluation steps off the end of the program.

The choice of *instruction set* should be based on the same principles as in tree-based GP. The ability of GP to find a solution strongly depends on the expressiveness of the instruction set. On the other hand, the dimension of the search space, i.e. all possible programs that can be built from these instructions, increases exponentially with the number of instructions and registers. Therefore, the set of possible instructions is limited to those of Table 1. The presence of the periodic `sine` operator in the operator set is strongly motivated by the fact that locomotion in biological creatures is cyclic in nature. To assure semantic correctness, a slightly different division operator (`div`) than the standard division operator is defined. It works exactly as the standard division operator, except for zero denominator input. In that case, the *protected division operator* returns a large constant value, here set to 10^8.

In LGP conditional branching is usually interpreted in the following way: if the condition in the `if` statement evaluates to `true`, the subsequent instruction is executed. If, on the other hand, the condition in the `if` statement evaluates to `false` that instruction is skipped, and program execution jumps to the next instruction instead, see Example 2.1. When using conditional branching, the control flow of the program may be different for different input situations, which is the motivation for including branching in this investigation.

Example 2.1 (Linear Program Execution)

This example shows a simple *register manipulating program* in C notation. The code from line 2 to line 10 constitutes the linear program, or instruction body, and the other code is just a wrapping for managing the GP program. The execution of the linear genome starts at the *topmost* instruction, i.e. `r[42]=r[55] * r[51]`, and proceeds *down* the list, one instruction at a time. Suppose that the first `if` statement (line 3) evaluates to `true`, and that the second `if` statement (line 7) evaluates to `false`. Then, the instructions on lines 2, 3, 4, 5, 6, 7, 9, 10 will be executed, but *not* the instruction on line 8.

```
1:  void GP(double* r){
2:     r[42] = r[55] * r[51];
3:         if(r[18] > r[32]);    \\true
4:     r[51] = r[61] * r[46];
5:     r[36] = r[48] - r[60];
6:     r[37] = r[28] * r[27];
7          if(r[15] > r[27]);    \\false
8:     r[42] = r[18] + r[45];
9:     r[37] = r[53] / r[36];
10:    r[26] = r[45] + r[55];
11:     return;
12: }
13: int main(void){
14:     double reg[80];
15:     ...
16:     GP(reg);
17:     ...
18:     return 0;
19: }
```

□

The actual register machine uses instructions for two or three registers. Three-register instructions operate on two arbitrary registers and assign the result in a third register (e.g. `r[i]=r[j]+r[k]`), while two-register instructions operate on only one operand (e.g. `r[i]=sin(r[j])`).

The term 'register' here can be misleading. The GP system *does not* operate on the actual CPU registers of the computer. The registers in this context are simply allocated as arrays of integer values in the RAM. Each instruction consists of four elements, encoded as integers, and the whole individual is a linear list of such instructions, see example 2.2 below.

Before the *initialization* process can begin, the maximum allowed genome length has to be defined. In LGP, the length of an individual is simply the number of basic instructions it contains. A linear genome individual is initialized in the following way:

Algorithm 2.2 (Initialization)

1. Randomly choose a *length* of the genome within the permitted interval.
2. Add a randomly chosen *instruction* to the genome.
3. Fill out the instruction with *register references* to randomly chosen registers from the register set and/or randomly chosen constants from the constant range.
4. Repeat step 2 and 3 until the number of instructions added equals the length chosen in step 1.
5. Repeat step 1 to 4 for every individual in the population.

□

So far, only steady-state *tournament selection* has been used. The following tournament selection scheme was used: two individuals are randomly picked from the population to compete against each other, and the best individual is selected with probability one. This procedure is carried out twice, and the two selected individuals mate and produce two new individuals. The newly created individuals are then inserted into the population, on the two worst individuals' place, which are then erased from the population. The quality of the randomly initialized individuals is usually very low. The initial population is transformed in an evolutionary search process by means of the *genetic operators*, crossover and mutation.

Crossover works by swapping parent individuals linear genome segments. Two crossover points in the first parent's genome are randomly chosen, and then two other crossover points in the other parent is chosen. The instructions in-between the crossover points are swapped, and the resulting individuals are the new offspring. This crossover operator allows the genome lengths of the individuals to vary over time, see figure 4.

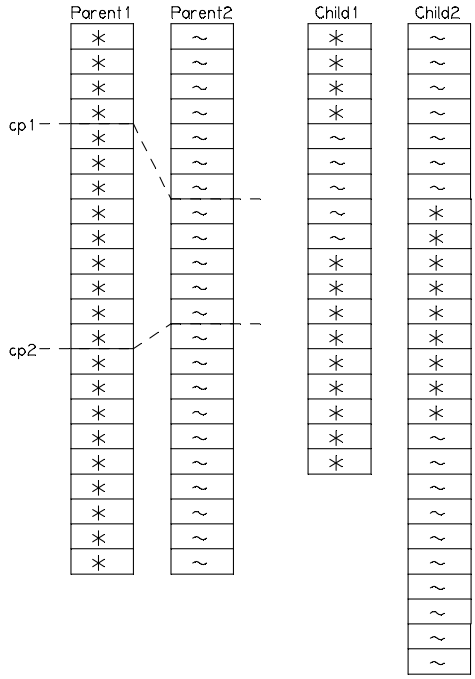

Figure 4: Length-varying two-point linear GP Crossover. Selecting different crossover points in both parents and swapping the intermediate genome segments varies the length of the genomes. The tokens ($*$, $\sim$) represent arbitrary instructions from the set.

In LGP, the mutation operator works on two different levels. The mutation operator first selects *which* instruction to be mutated, and then it makes one or more changes to that instruction. There are normally three types of change that can occur:

- Change any of the register destinations to another randomly chosen register.
- Change the operator in the instruction to another randomly chosen operator.
- Change a constant to another randomly chosen constant.

The atomic units of the mutated instruction (operator, registers and/or constant) must be members of the original sets.

In all GP-applications, finding a proper *fitness function* that guides the artificial evolution in the desired direction is of great importance. The primary goal of this project includes producing a sustainable, robust, anthropomorphic (i.e. human-like) bipedal gait. Defining the characteristics of a sustainable, robust gait is almost self evident; the biped should never fall, no matter how long time a good individual is being executed, and it should handle perturbations from the environment in a robust way. One can easily find proper measures for this, like 'time-of-walking', 'walking-speed', etc. Defining a measure for 'anthropomorphic' in this context is harder. Up to now, we simply let this be judged by the humans watching it.

2.2.3 EA for bipedal walking

Since the simulation is entirely deterministic are all individuals evaluated under identical conditions. They all start from the same upright pose, with the same orientation. The execution time for an individual is maximum N simulation time steps (evaluation time = $N/0.009$ s, where $N = 4000$), and if an individual causes the robot to fall before this time is reached, the evaluation is terminated. In the beginning of a simulation, the great majority of individuals are terminated before the intended evaluation time. The settings of the EA are summarized in table 2. The evolutionary algorithm used in these simulations was a steady-state tournament selection algorithm, utilizing two-point crossover and mutation, and with the following execution cycle:

Algorithm 2.3 (Evolutionary Algorithm)

1. Randomly *initialize* a population of individual solutions.
2. *Evaluate* all individuals according to the *fitness* measure, Eq. 2. The simulation comprises the following steps:
   a. Create an instance of the simulated robot.
   b. Record the initial position in 3d-space of all the robot's limbs.
   c. Execute the individual for N simulation time steps.
   d. Record the final position of all the robot's limbs.
   e. Compute the fitness value.
   f. Destroy the simulated robot.

3. Perform tournament *selection*.

4. Apply genetic operators *crossover* and *mutation* on the winners to produce two children.

5. *Replace* the two losers in the population with the offspring.

6. *Evaluate* the children according to the fitness function (as described above in step 2).

7. If the *termination criterion* is not met, go to step 3.

8. *Stop*. The best individual represents the best solution found.

□

2.2.4 Encoding scheme for walking

In the current implementation a chromosome takes form of an array of integers, as shown in example 2.2, where each row in the chromosome is interpreted as an instruction. An instruction is encoded in the following way: the first and second elements of an instruction refer to the registers to be used as arguments, the third element corresponds to the operator, and the last element is a register reference for where to put the result of the operation. The arithmetic operators are encoded as `add`=1, `sub`=2, `mul`=3, `div`=4, `sine`=5 in the genome. Conditional branching operators are encoded in the third element as 6=`if(r[j] > r[k])`, and 7=`if(r[j[ <= r[k])`. When decoded, the first line (instruction) in example 2.2 is interpreted as `r[42]=r[55] * r[51]`.

Example 2.2 (Linear Program Encoding)

This individual comprises nine instructions. The genome shown in this example is also shown in decoded form in Example 2.1.

```
55, 51, 3, 42,
18, 32, 6, 41,
61, 46, 3, 51,
48, 60, 2, 36,
28, 27, 3, 37,
15, 27, 6, 33,
18, 45, 1, 42,
53, 36, 4, 37,
45, 55, 1, 26,
```

□

The information flow between the control system and the robot is communicated through the registers. Currently, there are 67 registers used. At timestep t sensor signals are fed into the *sensor registers* ($r_{55} - r_{66}$). Simultaneously, the robot's current joint angle positions are recorded in the *I/O-* and *calculation registers* ($r_0 - r_{25}$ and $r_{26} - r_{51}$ respectively). Registers for ephemeral constants ($r_{52} - r_{54}$) are initiated at the beginning of the GP run. An ephemeral constant is initialized with a random value when created, and then it retains that value throughout its lifetime during a run. When the VRM executes the GP-individual (program) the contents of the calculation registers are manipulated. When the program execution has terminated, motor signal generation (MSG) is initiated; a modified signum function (Eq. 1) operates on the final contents of the calculation registers, and supplies the output to the I/O registers. These motor signals are then sent to the robot, which then in timestep $t + \eta$ executes the motor commands. A more detailed description of information flow in the controller is given in figure 3.

3 Simulations

A similar simulation using a simplified version of the LGP-system described in this paper was conducted by Wolff and Nordin [47] (Paper III in [46]). The bipedal model used was also simpler, i.e. it had a smaller number of DOF. The only feedback to the control system was the current joint angles of the biped.

The biped model used in the simulations described in this paper has a more human-like appearance. Further, the control system is enhanced with readings from accelerometers, and from a virtual inertial measurement unit (IMU). The LGP system is augmented with conditional branching as well.

In this section, the organization of three different sets of simulations is described. In the first set of simulations approximately 80 test runs were conducted in order to develop, test and verify the functionality of the evolutionary system with physics simulation. Preliminary results were obtained, which motivated some modifications and improvements of the evolutionary simulation system. These simulations are referred to as the *groundwork simu-*

Table 2: Koza style tableau, showing parameter settings for the evolution of locomotion control programs for the simulated humanoid robot.

Parameter	Value
Objective	Approximate a function that produce robust biped gait
Terminal Set	66 integer registers
Function Set	`add, sub, mul, div, sine, branching`
Raw Fitness	According to Eq. 1, scalar value
Standardized Fitness	Same as Raw Fitness
Population Size	between 8 and 512 individuals
Initialization Method	Random Gaussian distribution
Simulation Time	corresponding to 36 s. of real time
Crossover Probability	Between 0.0 and 1.0%
Mutation Probability	1.0
Initial Program Length	Gaussian distribution, between $\langle 32 \rangle$ and $\langle 512 \rangle$ instructions.
Maximum Program Length	1024 instructions
Maximum tournament number	25000
Selection Scheme	Tournament, size 4
Termination Criteria	Max tournament number

lations (Section 3.1). These simulations constitute the foundation for the setup of the simulations coming next.

As described in Section 3.2, *main simulations setup*, 60 independent simulation runs were performed mainly to investigate how the outcome is affected by specific parameter choices for the LGP system and the robot. Finally, a number of *additional simulations* are appended (Section 3.3). These were carried out in order to investigate some possible future enhancements of the evolutionary method.

3.1 Groundwork simulation setup

Here, the organization of the initial set of simulations is outlined. The fitness measure is described in detail, and its configuration is motivated. An implicit fitness measure, i.e. *energy discharging*, is introduced as well. The four different stopping criterions used are described as well.

3.1.1 Fitness Measure

Several different approaches to fitness measure was investigated, the first one investigated by Wolff and Nordin [47]. The initial assumption mentioned there, that it should be important to include the "height above ground of the robot" to the fitness measure, was dropped. The reason for this was that here, when this measure was included, it was found that the robot was prevented from moving freely enough to really do any advancement in efficient walking. To simply drop that term seemed to be the best choice here. Another problem arose when the gait controllers had reached the level of performance to maintain the robot in an upright pose and balance it. In order to reach higher levels of fitness they just let the robot stand idle until there was a very short moment of evaluation time left, and then take a large leap forward. Then, they get rewarded for good distance coverage over the trial, and the fact that the robot would have fallen

to the ground if evaluation was not terminated at that moment did not affect the evolutionary process. This became visible only when individuals were examined after the evolutionary run. Individuals can be recapitulated for infinitely long time afterwards. However, a proper fitness measure was found to be:

$$F = \sum_{t=0}^{N} (\mathbf{r}_R + \mathbf{r}_L)\hat{\mathbf{y}} \tag{2}$$

where N is the number of time steps in the simulation, $\mathbf{r}_R$ and $\mathbf{r}_L$ are the position vectors to the robot's right and left foot respectively, and $\hat{\mathbf{y}}$ is the unit vector along the y-axis, which is the forwards direction of the robot. One way to cope with the programs keeping the robot idle most of the time and activating it at the end of the evaluation, is simply to give it a (small) reward in every time step. The fitness function above is a sum of the contributions given for the robots actions in each time step. Thus the individual will be given higher fitness if the robot *gradually* approaches end point of its locomotion trajectory, rather than if it stand still until evaluation has almost finished, and then takes a large leap. That is, if the end point is the same in both cases.

3.1.2 Energy Discharging

In the early runs, most individuals hardly walked at all, and those who did, they walked in a very unnatural way. For those individuals, some of the joints showed "oscillating" behavior, i.e. they rapidly switch direction of revolution between each controller time step η. The oscillations were hardly visible for an observer, but nevertheless the robot slowly drifted across the floor. Another peculiar means of locomotion that emerged was robots walking sideways, like a crab fish do. That kind of gait is faster than the oscillating gait described above, but neither it is anthropomorphic. In order to prevent such misbehaviors to emerge, an energy discharging function was added. Human bipedal locomotion is very energy efficient, compared to the walks of humanoid robots. As an example, the state-of-the-art Honda humanoid Asimo uses at least 10 times the energy (scaled) of a typical human [9].

In each evaluation round the energy consumption of the biped was monitored as it moved its joints, and it was only allowed to use a specific amount of energy. The assumption was that this should yield a pressure on evolution to favor specific gaits that are energy efficient, and thus more anthropomorphic gaits ought to emerge.

Energy consumption, E_{jt}, of the j:th joint during timestep t equals the work performed in that joint during the timestep t. The total energy consumption of the biped, E_{tot} then, is the sum of E_{jt} over all joints $j_i \in \{j_0, j_1, ..., j_n\}$ and all timesteps $t_i \in \{t_0, t_1, ..., t_n\}$. Further, the work performed by a (generalized) force in circular motion, moving from an angle φ_a to φ_b, is defined by the following relationship:

$$W_{ba} = \int_a^b |\mathbf{M}| d\varphi \tag{3}$$

where $\mathbf{M}$ is the applied torque. In the simulation, time is discretized and the applied torque is constant during each timestep. Thus, the total energy consumption is given by:

$$E_{tot} = \sum_{j,t} (\varphi_{b,jt} - \varphi_{a,jt}) \, |\mathbf{M}_{ba,jt}| \tag{4}$$

When the total energy consumption E_{tot} equals some predefined value, evaluation of that individual is terminated what so ever. This can be thought of as a motor battery that discharges as the robot moves, and when the battery eventually gets empty, the robot stops.

3.1.3 Termination of Individuals

There are several ways in which the evaluation process of an individual could be terminated. First of all, there is a maximum allowed evaluation time of 4000 timesteps. When this time has passed, termination is definite. Second, if the current individual causes the biped to fall, evaluation will also be interrupted. Third, too high energy consumption (as described in section 3.1.2) could impose the termination of an individual as well. Last, in order to speed up the evolutionary process, another continual monitoring of individuals has also been introduced. A conditional termination criterion is specified according to the following expression:

$$\frac{F(i) + i_c}{t_i} < \frac{F_c + i_c}{i_c} \tag{5}$$

where $F(i)$ equals the fitness contribution at timestep t_i. F_c and i_c are constants, set to 20.0 and 1000 respectively. The interpretation of the above inequality is that the fitness contribution in each timestep should grow at least linear with time. The right hand side of Eq. 5 is a constant, specifying the minimal growth rate accepted. If the expression evaluates to *true* at some point, evaluation of that individual is terminated immediately. This constraint has the effect on evolution that individuals which spend most of their evaluation time standing idle are terminated earlier. Thus, a large portion of the expensive simulation time is freed up. Further, a positive side effect is that such individuals receive lower fitness scores.

3.1.4 The role of body proportions

In human development, basic skills like walking are acquired in an early age, usually before the age of 2.5 years. At that age, however, the body development is far from accomplished. That is, a child's head is extremely large in proportion to the rest of the body, compared to an adult's body. The head of a very young child make up about a quarter of their total height [5].

For that reason, simulation runs with different body proportions of the bipedal has been conducted. That is, evolution with the body proportions of an adult, and the body proportions of a child was examined.

3.2 Main simulation setup

The aim of these simulations was to investigate how the outcome was affected by specific parameter choices of the system. Parameters of the LGP system were investigated, as well as a parameter specific to the actual control problem. LGP specific parameters was *population size* $P(t)$, expectation value of *initial genome length* of the population $\langle L_i \rangle$, *crossover probability* p_c, and *mutation rate* r_{mut}. The problem specific parameter examined was the *initial energy level* E_i of the motor battery. Initially, these parameter values were set to their *default values*, which were determined empirically. The following default parameter values were used; 128, 128, 0.8, 0.2 and 128, respectively.

Table 3: Parameter settings examined, the default values are typeset in *italic*.

Parameter	Values
$P(t)$	8, 32, *128*, 512
$\langle L_i \rangle$	32, 64, *128*, 256
p_c	0.0, 0.2, *0.8*, 1.0
r_{mut}	0.1, *0.2*, 0.4, 0.8
E_i	32, *128*, 256, 512

3.3 Additional simulation setup

In this section, a number of supplementary runs will be described. Those runs were conducted because the results from the main simulation was not entirely convincing, so possible ways of improving the setup need to be investigated.

3.3.1 Robot controlled by dual controllers

When starting to walk the robot's configuration is transformed from the initial standing posture, into a cyclic motion pattern. Once the cyclic phase is entered, a configuration similar to the initial static configuration is not likely to occur again. At least not until the walking stops. It is reasonable to assume that the *transformation phase*, from static posture to dynamic walking, should best be handled by one controller, or brain, and the cyclic walking phase by another brain. Therefore, an approach with dual controllers, executed in series, was investigated as well. Control of the robot is then switched from the first brain (B1) to the second brain (B2) after a fixed amount of execution time. Here, the switching occurs after 3 s.

In this approach the genomes is divided in two parts, B1 and B2. Apart from that, everything else works in the same way as before, regarding the LGP system. However, the crossover operator is restricted to only exchange genetic material between the B1 part of the first individual, with genetic material of the B1 part of the second individual. Correspondingly, only B2 material can be exchanged with B2 material, between individuals.

3.3.2 Counting the number of foot steps

In order to prevent the robots from taking many short, small steps during walking, individuals were punished for the number of steps they used when

walking. A function was introduced that counted the number of walking steps, and then individuals were punished accordingly by reduced fitness rewards. This was done by simply multiplying the fitness score according to Eqation 2, with the following factor:

$$F_s = \frac{1}{\sqrt{s_{l,r}}} \tag{6}$$

where $s_{l,r}$ is the number of foot steps during walking, with the left and right foot respectively. Thus, the total fitness contribution is reduced according to a factor inversely proportional to the square root of the number of footsteps.

3.3.3 Obstacle in front of the robot

In an attempt to force individuals to make the robot to take larger and higher foot steps, an obstacle was placed directly in front of the robot at the evaluation. The height of the obstacle was varied, in different runs, between one fifth, and one half of the height of the robot's feet.

The simulation was based on the following assumption: if the robot's first step is high enough to climb the obstacle in front of it, it is more likely that the robot will continue to take that high foot steps. Further, that one individual will also get higher fitness than other individuals, which cannot cope with the obstacle, and thus give rise to more offspring. Hence, individuals taking high foot steps will be promoted by evolution.

3.3.4 A lifting force applied to the robot

It is very a difficult task to learn to walk, just starting from a standing pose. In an attempt to make it easier for individuals to start walking, an approach was tried were an *upwards* directed force was attached to the robots body during evaluation. This setup resemble to some extent the situation were an infant is learning to walk, supported by his parent or another adult person: when the child starts to lose balance, the supervisor (adult person) can quickly give him support and lift him up again, so that he can continue with the training again. In this simulation, the *spring force* was applied to the robot's body, which prevented the robot from completely falling to the ground. The spring force obeyed Hooke's law, $F = C \cdot \Delta l$, where C is the so called spring constant, and Δl is the vertical displacement from the starting position. Note that there are *no* horizontal components of the spring force, just the vertical one.

4 Results

4.1 Groundwork simulation results

Apart from the testing, verification, and fine-tuning of the LGP system with physics simulation, the major contribution from these simulations is that adding the *energy discharging function* improved the results by a great portion. More anthropomorphic gaits evolved, and both the oscillating behaviors and the crab-like walking behavior ceased to emerge from the evolution.

The case when comparing evolution with different robot morphologies (Section 3.1.4) did not yield any specific result. Similar gaits emerged with both morphologies, approximately equally often.

4.2 Main simulation results

This section presents the outcome of 60 independent simulation runs. Due to limitations of time and computational power, not all of the 1024 possible combinations of alternative parameter settings were examined. Instead the parameters were varied, one at a time, from their default value settings. Thus, we obtain 20 unique parameter combinations. The actual settings for each run are clear from table 4, as well as the resulting fitness values. For statistical significance all runs were repeated three times with identical parameter settings.

The best over-all fitness values were obtained for parameter settings equal to 128, 128, 0.8, 0.2, and 256. As clear from table 4 default parameter values of $P(t)$ and r_{mut} produced the best fitness values, of respective partition. A (default) value of 0.8 for the parameter p_c also gave the best result of its partition, at least when looking at the $\bar{f}_{\max}$ and $\bar{f}_{avg}$ fitness values. On the contrary, when considering only the best fitness value of a single individual of that partition, a p_c value of 0.0 (i.e. only mutation was used) outperformed the others, however with the worst value of $s(\bar{f}_{\max})$. The best fitness values for $\langle L_i \rangle$ and E_i were obtained for values of 32 and 256, respectively.

Table 4: Fitness values of the best individuals, $\max(\bar{f}_{\max})$; averages of the best individual fitness values, $\bar{f}_{\max}$, with standard deviations $s(\bar{f}_{\max})$; averages of the population fitness values, $\bar{f}_{avg}$. All values were found after 25000 tournaments. Each one of the five partitions of the table shows the variations of a single parameter value. Numbers in bold represent the best fitness values of each partition.

$\max(\bar{f}_{\max})$	$\bar{f}_{\max}$	$s(\bar{f}_{\max})$	$\bar{f}_{avg}$	$P(t)$	$\langle L_i \rangle$	p_c	r_{mut}	E_i
5728	4635	1777	3475	8	128	0.8	0.2	128
5839	5015	1144	4701	32	128	0.8	0.2	128
7303	**6760**	480	**6535**	128	128	0.8	0.2	128
6954	6604	486	5032	512	128	0.8	0.2	128
7449	**6900**	586	**6546**	128	32	0.8	0.2	128
5817	5065	848	4864	128	64	0.8	0.2	128
7303	6760	480	6535	128	128	0.8	0.2	128
6176	5232	1194	4007	128	256	0.8	0.2	128
7943	6054	1662	4621	128	128	0.0	0.2	128
6580	5468	973	5303	128	128	0.2	0.2	128
7303	**6760**	480	**6535**	128	128	0.8	0.2	128
7197	6687	578	6419	128	128	1.0	0.2	128
6417	5170	1109	4679	128	128	0.8	0.1	128
7303	**6760**	480	**6535**	128	128	0.8	0.2	128
6797	6292	580	5933	128	128	0.8	0.4	128
6308	6194	148	4517	128	128	0.8	0.8	128
2049	1270	581	1101	128	128	0.8	0.2	32
7303	6760	480	6535	128	128	0.8	0.2	128
8958	**7598**	1375	**7368**	128	128	0.8	0.2	256
7076	6352	660	6020	128	128	0.8	0.2	512

Summarizing the results of table 4; best fitness values of each partition of the table, according to $\bar{f}_{avg}$, were obtained for the following five parameter values: 128, 32, 0.8, 0.2, and 256. The best individual found had a fitness value of 8958. During its evaluation time, which corresponds to 36 s of real time simulation, it covered a distance of 1.93 m.

A second series of simulation runs were performed with the parameter value settings equal to 128, 128, 0.8, 0.2, and 256. Then, the best individual obtained had a fitness value $\max(\bar{f}_{\max})$ of 9044, and the averages of the best individual fitness values $\bar{f}_{\max}$ were equal to 7483. The deviation of fitness values compared with the best runs of the first series was within the margin for error.

The graphs of figure 6 show covered distances, $d(t)$, for the best individual found and two other individuals, for comparison. As clear from the figure, the best individual (i.e. the one labeled "8958") traveled the longest distance during the evaluation time, and thus received highest fitness score. At the end of the evaluation period, however, it fell backwards. That occurence is indicated by the graph $d(t)$ dropping off at the end. The other two individuals both remained on their feet for the whole evaluation period of 4000 timesteps, but they received lower fitness values. Successful individuals in these simulations all exhibit cyclic locomotion behavior, exemplified by the graphs in figure 7.

4.3 Additional simulation results

Only a few preliminary test runs have been performed with these simulation setups. The setup with a lifting force applied to the robot (Section 3.3.4) did not yield any improvement. No walking behavior at all emerged.

The case with an object placed in front of the robot (Section 3.3.3), no individual so far have

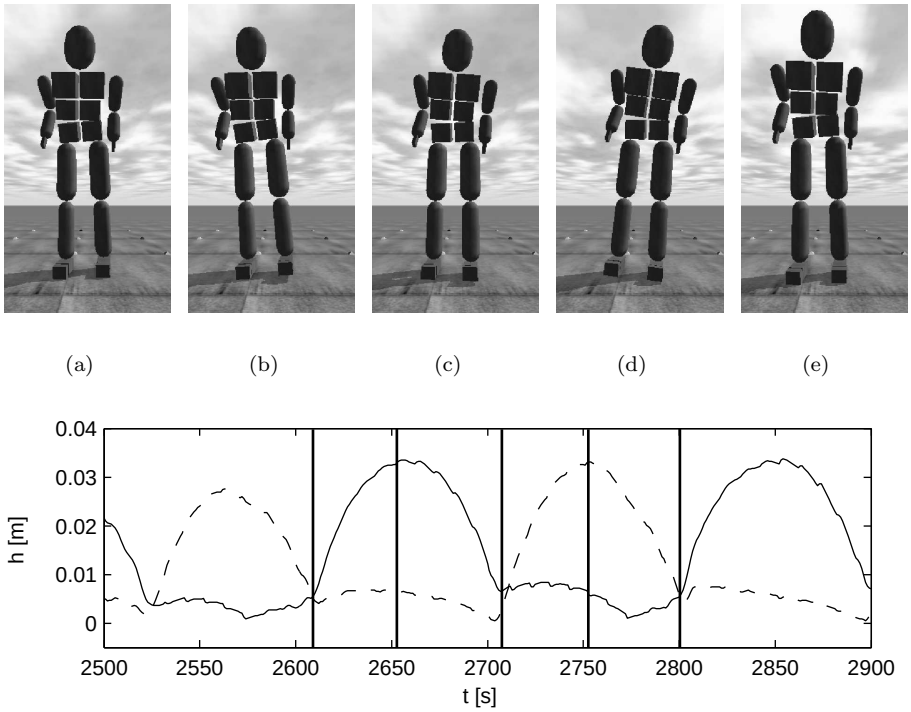

Figure 5: *Top panel*: Walking scene of the bipedal robot. Starting from the left, double support phase is depicted (a), followed by the single support phase with left foot in swing motion (b). Then again a double support phase (c), followed by single support phase with the right foot in swing motion this time (d). Finally, the gait cycle is completed with a double support phase (e). *Bottom panel*: Plot of the depicted gait cycle. The graphs show the height above ground, $h(t)$, of the CoM point of the left foot (solid line) and the right foot (dashed line), respectively. Horizontal positions of the vertical lines indicate corresponding double- and single support phases (a) throgh (e).

managed to climb the obstacle. Thus, no walking behavior emerged.

When counting the number of foot steps of the robot (Section 3.3.2), the best individual found so far managed to walk a few steps, and then it stood idle until it got terminated, accumulating some fitness meanwhile. Most runs in the main simulation did better off.

However, in the simulation with two brains serially controlling the robot, individuals emerged that has almost as high fitness values as the best individuals from the main simulation runs. But, considering the way they walk, it looks even better than the best individuals from previous runs. That is, the gaits look more anthropomorphic.

5 Discussion and Conclusions

The primary goal of the simulations reported in this paper was to evolve robust, anthropomorphic bipedal locomotion. Evolution generated controller programs that made the simulated robot stride across its environment by means of bipedal locomotion. When executed afterwards, a couple of the best individuals were actually capable of producing locomotion behavior several times longer than the evaluation time of 36 s. For instance, the individual labeled '5592' in this paper was executed for a time period of more than 20 minutes, and the robot was still walking forward. However, no one individual exceeded a walking speed of 5.37 cm/s. Clearly, this is a very moderate walking speed, considering the

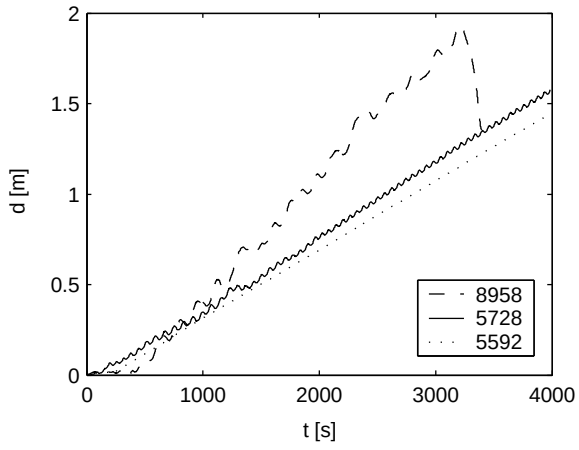

Figure 6: Graphs of covered distances, $d(t)$, during 4000 evaluation timesteps, for three different individuals. These individuals received fitness scores of 8958, 5728 and 5591, respectively. The individual with fitness value 8958 was actually the best individual found in all the runs.

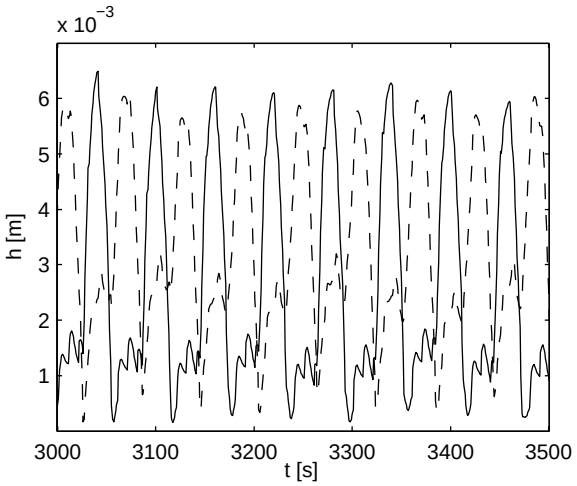

Figure 7: Time series of resulting body dynamics, of the individual of fitness 5728. The graphs show the height above ground, $h(t)$, of each foot, during 500 timesteps (equal to 4.5 s) of simulation. The fact that both curves never indicate an altitude of zero (which is supposed to happen when a foot touches the ground) is a consequence of the fact that both feet were constantly tilted some unspecified angle away from the horizontal plane, which the graphs were not compensated for. However, since this individual performs a cyclic walking pattern each local maximum indicates a single footstep. The figure contains a sequence of 9 steps of the right foot (dashed line), and 8 steps of the left foot (solid line).

fact that the dimensions of the bipedal robot corresponds to the size of a (small) human being. Even the best individuals lifted their feet only a small distance above the ground when walking, and also used a very short step length. Given these characteristics, low walking speed will of course be the result. These individuals simply did not activate the knee joints of the biped very much. When walking, these individuals activated the hip joints primarily. This kind of locomotion behavior is somewhat dissimilar to human gait. However, apart from these defects, good individuals generally made the robot walk in an upright, forward directed manner. Further, some of the successful individuals clearly utilized the inherent dynamics of the physical system when walking.

Evolution is a theory of gradual, hereditary change. When observing the evolutionary process of any one of the successful simulation runs described here, that statements is clearly supported. For example, the individual previously referred to as '5728' in this paper shows an atypical way of walking with its left foot slanted "toe up", i.e. only the heel is in contact with the ground. Individual '5728' emerged as the best one of the run in tournament number 18839, but the above mentioned peculiarity was observed in evolution as early as in tournament 180, for an individual with fitness equal to 444. It could very well be possible that different "tracks" of anomalies, e.g. like the one described above, could co-exist during evolution, but that was not investigated here.

Figure 8 show fitness progress during a representative evolutionary run. Typically, there are several distinct increments in the fitness curve of the best individual. Some of these fitness leaps correspond to what could be thought of as transitions in evolution. In the initial randomly generated population, almost all individuals fall over after a few seconds of evaluation, thus they receive poor fitness scores. Rather soon however, individuals learn to stand up without falling over. At this stage, they

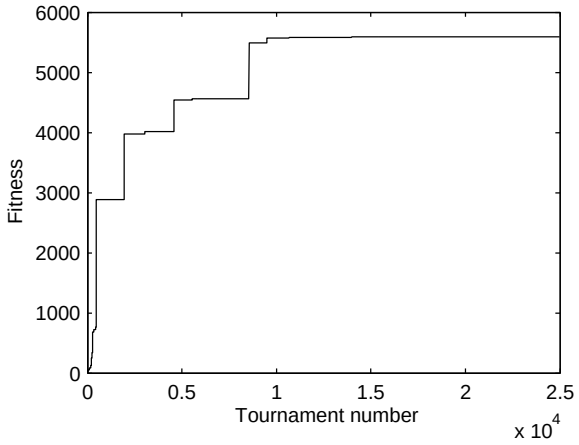

Figure 8: Fitness progress during an evolutionary run, exemplified by the run which generated individual of fitness '5728'.

receive slightly higher fitness, but they get terminated according to the fourth termination criterion (Eq. 5). The next transition occurs when the best individuals get more movable, mainly by activating the robot's hip joints. Gradually, the magnitude of these motions increase, which results in a few small forward steps. Hence, fitness scores reach the next level, despite most of the individuals fall over after a while. The evolutionary stage following from that include individuals that can continue to walk forward without falling at all, but they all get terminated, again because of the fourth criterion. The last great transition arises when a majority of individuals begin to walk just slightly faster, but receive much better fitness scores. They manage to adapt their walking pace to just above the threshold defined by the fourth criterion (Eq. 5), and as a consequence they walk almost the full evaluation time, terminated only because of the energy discharging criterion. After that, evolution favors those individuals that can cut down the energy cost. During that phase only small fitness increments were observed.

6 Future work

In order to fully accomplish the goals of this project, there are several alternative extensions or modifications left to investigate.

In this study a population of individuals, or programs, is evaluated for the task of bipedal locomotion. When evaluating individuals in the simulation, all individuals have to start the robot from the same upright standing position (similar to the 'attention' stand). The activity of bipedal walking is mainly cyclic to its nature. It might therefore seem unnatural to commit charge of control to one single closed-loop program in such a position, when that posture is not present as any of the postures in the walking cycle. Accordingly, we have carried out preliminary simulations using two individuals controlling the robot. One program is controlling the robot for the first 3 s of evaluation time, and then control is taken over by a second program. Thus, we have one *start-up program*, and one *cyclic program* in the same individual. The results so far, which are based on a few simulation runs only, look promising. However, the evaluation time greatly increases since there are two genomes, instead of one, in each individual which should evolve into successful individuals. Currently, the point at which control is taken over by the second individual is specified at before hand. A natural extension would thus be to let this moment be decided upon during execution of the individual instead. A suitable *behavior selection* method to do this would be the utility functions (UF) method for instance [32].

Even the best individuals generated in the simulations described in this paper are impaired by serious shortcomings, as mentioned in the previous sections. First of all, the foot steps are too short, and second, the knee joints are used very little. Before any really anthropomorphic gaits can be expected to emerge, such weaknesses must be eliminated.

Evolving robot controllers for such a complex task as bipedal locomotion requires realistic simulation of the bipedal robot model. Such simulations are computationally very costly. During an evolutionary run, most of the computation time (maybe as much as 99%) is spent on evaluating individuals by means of full rigid-body simulations. Only a small fraction of the time is allotted to EA related tasks (mutations, crossover etc.). An approach to speed up the evolutionary search process has been proposed by Ziegler et al. [50]. In their method a second GP system is involved, which evolves meta-models of the first GP system's fitness function. Hence, time consuming evaluations can

be replaced by faster classifications of individuals, or fitness value estimations. In their simulations (evolving gaits for a 12 DOF, four-legged robot) they could save as much as 50% of the evaluations with this method, compared to "standard" fitness evaluations.

References

[1] M. Abdallah and A. Goswami. A biomechanically motivated two–phase strategy for biped upright balance control. In *Proceedings of the International Conference on Robotics and Automation, ICRA'05*. IEEE, 2004.

[2] T. Arakawa and T. Fukuda. Natural motion generation of biped locomotion robot using hierarchical trajectory generation method consisting of GA, EP layers. In *Proceedings of the International Conference on Robotics and Automation, ICRA'97*, pages 211–216. IEEE, 1997.

[3] W. Banzhaf, P. Nordin, R. Keller, , and F. Francone. *Genetic Programming – An Introduction: On the Automatic Evolution of Computer Programs and Its Applications*. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1998.

[4] A. Boeing, S. Hanham, and T. Bräunl. Evolving autonomous biped control from simulation to reality. In *Proceedings of the Second International Conference on Autonomous Robots and Agents, ICARA'04*, Palmerston North, New Zealand, 2004. IEEE, Massey University.

[5] B. A. Bogin. *Patterns of Human Growth*. Cambridge University Press, 2 edition, 1999.

[6] M. Brameier. *On Linear Genetic Programming*. PhD thesis, Dortmund University, 2003.

[7] G. Capi, Y. Nasu, L. Barolli, K. Mitobe, and K. Takeda. Minimum energy gait synthesis for walking biped robots based on genetic algorithms. In *Proceedings of the International Conference on Production Engineering, Design and Control PEDAC'01*, volume 2, pages 903–912, 2001.

[8] M. Y. Cheng and C. S. Lin. Genetic algorithm for control design of biped locomotion. *Robotic Systems*, 14(5):365–373, 1997.

[9] S. H. Collins, A. L. Ruina, R. Tedrake, and M. Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307:1082–1085, 2005.

[10] C. Darwin. *On the Origin of Species by means of Natural Selection or the Preservation of Favored Races in the Struggle for Life*. Murray, London, 1859.

[11] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. Wiley, New York, NJ, 1966.

[12] Y. Fujimoto and A. Kawamura. Simulation of an autonomous biped walking robot including environmental force interaction. *IEEE Robotics & Automation Magazine*, pages 33–42, jun 1998.

[13] Y. Fujimoto, S. Obata, and A. Kawamura. Robust biped walking with active interaction control between foot and ground. In *Proceedings of the International Conference on Robotics and Automation, ICRA'98*, volume 3, pages 2030–2035. IEEE, 1998.

[14] J. Furusho and M. Masubuchi. A theoretically motivated reduced order model for the control of dynamic biped locomotion. *Journal of Dynamic Systems, Measurement and Control*, 109:155–163, 1987.

[15] J. Furusho and A. Sano. Sensor-based control of a nine-link biped. *International Journal of Robotics Research*, 9(2), 1990.

[16] A. Goswami. Postural stability of biped robots and the foot-rotation indicator (FRI) point. *The International Journal of Robotics Research*, 18:523–533, 1999.

[17] A. Goswami and V. Kallem. Rate of change of angular momentum and balance maintenance of biped robots. In *Proceedings of the International Conference on Robotics and Automation, ICRA'04*, volume 4, pages 3785–3790. IEEE, 2004.

[18] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of Honda humanoid robot. In *Proceedings of the International Conference on Robotics and Automation, ICRA'98*, volume 2, pages 1321–1326. IEEE, 1998.

[19] J. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, MI, USA, 1975.

[20] Q. Huang and Y. Nakamura. Sensory reflex control for humanoid walking. *IEEE Transactions on Robotics and Automation*, 21(5):977–984, 2005.

[21] Q. Huang, Y. Nakamura, and T. Inamura. Humanoids walk with feedforward dynamic pattern and feedback sensory reflection. In *Proceedings of the International Conference on Robotics and Automation, ICRA'01*, pages 4220–4225. IEEE, 2001.

[22] L. Huelsbergen. Toward simulated evolution of machine–language iteration. In *Proceedings of the '96 Conference on Genetic Programming*, pages 315–320, Stanford University, CA, USA, 1996.

[23] S. Kajita and K. Tani. Adaptive gait control of a biped robot based on realtime sensing of the ground profile. In *Proceedings of the International Conference on Robotics and Automation. ICRA'96*, pages 570–577. IEEE, 1996.

[24] D. Katic and M. Vukobratovic. Survey of intelligent control techniques for humanoid robots. *Intelligent and Robotic Systems*, 37(2):117–141, 2003.

[25] J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* The MIT Press, Cambridge, MA, USA, 1992.

[26] Q. Li, A. Takanishi, and I. Kato. Learning control of compensative trunk motion for biped walking robot based on zmp stability criterion. In *Proceedings of the '92 International Conference on Intelligent Systems*, pages 597–603. IEEE, 1992.

[27] S. Mojon. Using nonlinear oscillators to control the locomotion of a simulated biped robot. Master's thesis, Computer and Communication Sciences, BIRG, EPFL, Lausanne, Switzerland, 2004.

[28] P. Nordin. *Evolutionary Program Induction of Binary Machine Code and its Applications.* PhD thesis, Dortmund University, Dortmund, Germany, 1997.

[29] S. Ok, K. Miyashita, and K. Hase. Evolving bipedal locomotion with genetic programming —a preliminary report—. In *Proceedings of the Congress on Evolutionary Computation, CEC'01*, pages 1025–1032, Seoul, South Korea, 2001. IEEE.

[30] J.H. Park and H.C Cho. An on-line trajectory modifier for the base link of biped robots to enhance locomotion stability. In *Proceedings of the International Conference on Robotics and Automation, ICRA'00*, pages 3353–3358. IEEE, 2000.

[31] J. Pettersson, H. Sandholt, and M. Wahde. A flexible evolutionary method for the generation and implementation of behaviors for humanoid robots. In *Proceedings of the 2nd International Conference on Humanoid Robots, Humanoids'01*, pages 279–286, Waseda University, Tokyo, Japan, 22-24 November 2001. IEEE-RAS, Humanoid Robotics Institute.

[32] J. Pettersson and M. Wahde. Application of the utility function method for behavioral organization in a locomotion task. *IEEE Transactions on Evolutionary Computation*, 9(5):506–521, oct 2005.

[33] I. Rechenberg. Cybernetic solution path of an experimental problem. *Royal Aircraft Establishment Translation*, (1122), August 1965.

[34] T. Reil and P. Husbands. Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Transactions in Evolutionary Computation*, 6(2):159–168, 2002.

[35] T. Reil and C. Massey. Biologically inspired control of physically simulated bipeds. *Theory in Biosciences*, 120(3–4):327–339, 2001.

[36] C. Ridderström. *Legged locomotion: Balance, control and tools - from equation to action.* PhD thesis, The Royal Institute of Technology, Stockholm, Sweden, 2003.

[37] A. Takanishi, M. Ishida, Y. Yamazaki, and I. Kato. The realization of dynamic walking by the biped walking robot WL-10RD. In *Proceedings of the International Conference on Advanced Robotics, ICAR'85*, pages 459–466, 1985.

[38] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc London Math Soc*, 42:230–265, 1936.

[39] M. Vukobratovic. How to control artificial anthropomorphic systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(5):497–507, 1973.

[40] M. Vukobratovic and B. Borovac. Zero-moment point – thirty five years of its life. *International Journal of Humanoid Robots*, 1(1):157–173, 2004.

[41] M. Vukobratovic, B. Borovac, and D. Surdilovic. Zero-moment point - proper interpretation and new applications. In *Proceedings of the 2nd International Conference on Humanoid Robots, Humanoids'01*, pages 279–286, Waseda University, Tokyo, Japan, 22-24 November 2001. IEEE-RAS, Humanoid Robotics Institute.

[42] M. Vukobratovic, A. A. Frank, and D. Juricic. On the stability of biped locomotion. *IEEE Transactions on Bio-Medical Engineering*, 17(1):25–36, 1970.

[43] M. Vukobratovic and D. Juricic. Contribution to the synthesis of biped gait. *IEEE Transactions on Bio-Medical Engineering*, 16(1), 1969.

[44] M. Vukobratovic and J. Stepanenko. On the stability of anthropomorphic systems. *Mathematical Biosciences*, 15:1–37, 1972.

[45] M. Wahde and J. Pettersson. A brief review of bipedal robotics research. In J. van Amerongen, J. Jonker, P. Regtien, and S. Stramigioli, editors, *Proceedings of the 8th Mechatronics Forum International Conference, Mechatronics'02*, pages 480–488, Enschede, the Netherlands, 24-26 June 2002. IEE, Drebbel Institute for Mechatronics, Twente University.

[46] K. Wolff. *Evolutionary Humanoids for Embodied Artificial Intelligence.* Licentiate thesis, Chalmers University of Technology, Göteborg, Sweden, December 2003.

[47] K. Wolff and P. Nordin. Learning biped locomotion from first principles on a simulated humanoid robot using linear genetic programming. In E. Cantú-Paz, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'03)*, volume 2723 of *LNCS*, pages 495–506, Chicago, 12-16 July 2003. AAAI, Springer Verlag.

[48] J. Yamaguchi, E. Soga, S. Inoue, and A. Takanishi. Development of a bipedal humanoid robot-control method of whole body cooperative dynamic biped walking. In *Proceedings of the International Conference on Robotics and Automation, ICRA'99*, pages 368–374. IEEE, 1999.

[49] Y.F. Zheng and J. Shen. Gait synthesis for the SD–2 biped robot to climb sloping surface. *IEEE Transactions on Robotics and Automation*, 6(1):86–96, 1990.

[50] J. Ziegler and W. Banzhaf. You can judge a book by its cover – evolution of gait controllers based on program code analysis. In *Proceedings of the 6th International Conference on Climbing and Walking Robots, CLAWAR'03*, pages 205–212. Professional Engineering Publishing, 2003.

[51] J. Ziegler, J. Barnholt, J. Busch, and W. Banzhaf. Automatic evolution of control programs for a small humanoid walking robot. In *Proceedings of the 5th International Conference on Climbing and Walking Robots, CLAWAR'02*. Professional Engineering Publishing, 2002.

Paper IV

Behavioral selection using the utility function method: A case study involving a simple guard robot

In

Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment, pages 261–266, Fukui, Japan, September 20-22, 2005.

Behavioral Selection Using the Utility Function Method: A Case Study Involving a Simple Guard Robot

Mattias Wahde[1], Jimmy Pettersson[1], Hans Sandholt[1], and Krister Wolff[1,2]

[1] Department of Applied Mechanics, Chalmers University of Technology, 412 96 Göteborg, Sweden
{mattias.wahde, hans.sandholt, jimmy.pettersson}@chalmers.se
[2] Department of Microtechnology and Nanoscience, Chalmers University of Technology, 412 96 Göteborg, Sweden
krister.wolff@mc2.chalmers.se

Summary. In this paper, the performance of the utility function method for behavioral organization is investigated in the framework of a simple guard robot. In order to achieve the best possible results, it was found that high-order polynomials should be used for the utility functions, even though the use of such polynomials, involving many terms, increases the running time needed for the evolutionary algorithm to find good solutions.

1 Introduction

In behavior-based robotics (BBR) [1], the artificial brain of a robot is built in a bottom-up fashion, starting from simple low-level behaviors. An obstacle facing the behavior-based approach is the problem of behavioral selection, i.e. the problem of activating appropriate behaviors at all times. In simple robots, with small behavioral repertoires, the selection of behaviors (for activation) can be generated manually, which indeed is what is done in most methods for behavioral selection [4, 5, 6].

However, in robots with larger behavioral repertoires, specifying behavioral selection by hand is a daunting task, not least because of the difficulty in comparing, at all times and in all situations, the relative merits of several behaviors. Such comparison requries a common currency which, in economic theory and game theory, goes under the name *utility*, a concept that has also been introduced in ethology and, more recently, in robotics [3, 4].

In order to overcome the difficulties associated with behavioral selection, a method known as the *utility function* (UF) *method* has been developed [4]. In this method, behavioral selection is based on the value of utility functions

that are *evolved* rather than hand-coded, thus minimizing the bias introduced by the user of the method.

In this paper, the UF method will be illustrated by means of an example, namely a simple simulated guard robot. In addition, the performance for various utility function specifications will be studied.

2 The Utility Function Method

Due to space limitations, only a brief description of the UF method will be given here. A more complete discussion of the method is available in [4]. In the UF method, each behavior $B_j, j = 1, \ldots, N$, where N is the number of behaviors, is associated with a utility function, whose variables are (a subset of) the state variables of the robot. The state variables are of three kinds: External variables, denoted s_i (e.g. the readings of IR sensors on the robot), internal physical variables, denoted p_i (e.g the readings of a battery sensor), and internal abstract variables, denoted x_i. The latter correspond to the readings of internal variables known as hormones in the UF method. In general, each utility function is given by a polynomial ansatz. For example, the ansatz for a second-degree polynomial utility function of two variables s_1 and x_1 is given by

$$U(s_1, x_1) = a_{00} + a_{10}s_1 + a_{01}x_1 + a_{20}s_1^2 + a_{11}s_1x_1 + a_{02}x_1^2. \tag{1}$$

The UF method is an arbitration method, i.e. a method in which one and only one behavior is active at any given time. Behavioral selection is simple in the UF method: at all times, the behavior whose utility function takes the highest value is activated. The problem, of course, is to specify the utility functions so as to generate purposeful and reliable behavioral selection. In the UF method, this is done using an evolutionary algorithm (EA). As in any EA, a fitness function must be specified. In the UF method, the fitness is often associated with the execution of a given task behavior, the other behaviors being considered as auxiliary behaviors, i.e. behaviors which are needed (such as battery charging), but which do not increase the fitness of the robot. Once the fitness function has been specified, the task of the EA is thus to set the coefficients of the N polynomial utility functions.

An interesting question in this regard concerns the number of such coefficients, which, in turn, determines the complexity of the problem that the EA must solve. In general, it can be shown that a polynomial function of n variables and of degree p contains

$$\binom{n+p}{p} = \binom{n+p}{n} \tag{2}$$

distinct terms. (For example, in Eq. (1), $n = 2$ and $p = 2$, so that the number of terms, according to Eq. (2), equals $\binom{4}{2} = 6$).

3 Case Study: A Simple Guard Robot

As a case study, consider a simple simulated guard robot whose task it is to patrol the arena shown in the left panel of Fig. 1. The arena contains numerous obstacles in the form of pillars, as well as three battery charging stations, located at corners in the arena. The simulated robot is a differentially steered, two-wheeled robot, with two DC motors. The robot will be equipped

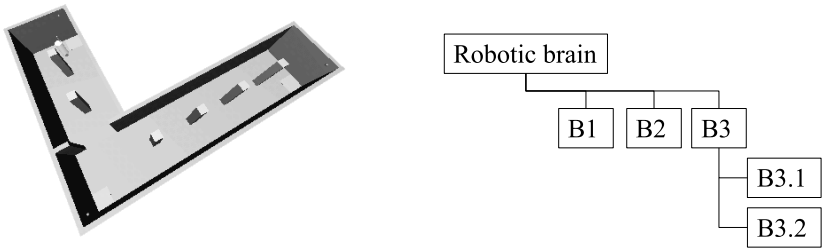

Fig. 1. Left panel: The arena patrolled by the guard robot. Right panel: Behavioral hierarchy of the robotic brain.

with five behaviors, namely *straight-line navigation* (B_1), *obstacle avoidance* (B_2), *energy maintenance* (B_3), *corner seeking* ($B_{3.1}$), and *battery charging* ($B_{3.2}$). In the UF method, as implemented in the UFLibrary software library currently under development at Chalmers University of Technology, behaviors are organized in hierarchies, as indicated in the right panel of Fig. 1. Utility functions are compared on a level-by-level basis. Thus, for each time step, it is determined which of the three functions U_1, U_2, and U_3 takes the highest value. If it happens to be U_3, the comparison of $U_{3.1}$ and $U_{3.2}$ will determine which of these two behaviors is active.

In B_1, the robot simply moves in a straight line, by setting its motor outputs to equal values. In B_2 the robot turns until no obstacle is visible in front of it, and then stops. If $B_{3.1}$ is active, the robot will rotate in an attempt to find a charging station (each of which is associated with an IR beacon detectable by a sensor on the robot). In $B_{3.2}$ the robot remains at a standstill, charging the batteries *if* it happens to be at a charging station.

Since the task of the robot is to cover as much of the arena as possible, a suitable fitness measure is simply the time spent in the navigation behavior B_1. In order to avoid rapid swapping between behaviors, a slightly modified fitness function was used, however, in which the robot only obtains a fitness increase if it spends at least one full second executing B_1.

Due to space limitations, the full ansatz for each utility function will not be given here. Suffice it to say that the number of variables in $U_1, U_2, U_3, U_{3.1}$, and $U_{3.2}$ were 4, 3, 3, 2, and 1, respectively. Thus, for degree p, the number of polynomial coefficients that must be determined by the EA equals

$$n_c = \binom{4+p}{4} + 2\binom{3+p}{3} + \binom{2+p}{2} + \binom{1+p}{1}. \quad (3)$$

4 Simulation program

A simulation program was written in Delphi object-oriented Pascal [2] using the UFLibrary software package. The UFLibrary provides a general implementation of the UF method, handling all issues involving the evolution of polynomial utility functions for behavioral selection. The task of the user is to provide (1) the constituent behaviors for the behavioral repertoire, (2) the polynomial degree p of the utility functions, (3) a fitness function, (4) the arena, in the form of a text file readable by the UFLibrary, (5) the definition of the robot body and the general structure of its brain (as shown in the right panel of Fig. 1), also in the form of a text file in a given format. The specification of the robotic brain also involves specifying the state variables included in each utility function polynomial.

5 Results

For simplicity, the simulations were performed without any sources of noise. Each evaluated individual was allowed a maximum simulation time of $100s$. However, the evaluation of a simulated robot was terminated directly in case of collisions with obstacles or if the on-board battery became fully discharged. In each run, 10,000 individuals were normally evaluated, even though some extended runs were carried out as well (see below).

Four different polynomial degrees were investigated, namely $p = 1, 2, 3$, and 4, for which the number of polynomial coefficients (n_c) equals 18, 44, 89, and 160, respectively. In order to allow a fair comparison between the performance of the EA for different values of p, mutation rates (p_m) were set to $1/n_c$ or $3/n_c$. The population sizes n_p were set to 20 or 100 individuals. In the UFLibrary, the crossover procedure swaps entire polynomials between chromosomes. Here, the crossover probability p_c was set to 0.2 or 0.8. Furthermore, tournament selection was used, with the tournament size n_t set to 10% of the population size. Thus, a total of $4 \times 2 \times 2 \times 2 = 32$ parameter combinations were investigated. Furthermore, because of the stochasticity of EAs, several (N_R) runs had to be performed for each setting, in order to form a reliable average. Since a typical run lasted for a few hours, N_R was limited to 5-7, resulting in a total of around 200 runs.

Due to the richer dynamical structure accessible in runs with large p, it could perhaps be expected that these runs would outperform those with lower p values. However, no such simple trend was found: The averages (over all runs with a given value of p) of the best fitness values found after 10,000 evaluated individuals, were found to be 10.36, 9.33, 10.31, and 9.43, for $p = 1, 2, 3$, and

Table 1. Averages of the best fitness values found after 10,000 individuals, for runs with different values of the polynomial degree p and the mutation rate p_{m}.

	Polynomial degree			
p_{m}	1	2	3	4
$1/n_c$	8.973	6.396	7.255	5.291
$3/n_c$	11.76	11.73	13.35	13.59

4, respectively. As can be seen in Table 1 there is, on the other hand, a strong trend in favor of the larger of the two mutation rates. A similar, albeit weaker, trend (not shown) was also found in favor of large population sizes, whereas no discernible trend was encountered for the crossover probability. Note also that, within the categories shown in Table 1, the spread between different runs was quite large, with the majority of runs reaching rather low fitness values.

6 Discussion and conclusion

While the average results obtained from the runs performed here show only very little variation with the polynomial degree p, this does not necessarily imply that the choice of p does not matter. Two possible interpretations of the results are that (1) perhaps larger values of p do indeed make it possible to achieve better results but that *finding* such solutions becomes progressively more difficult as p is increased (due to the large increase in the size of the search space), or (2) maybe $p = 1$ or 2 is sufficient for the problem at hand and that, in the runs with larger p, the coefficients in front of the third and fourth-order terms are simply eliminated by the EA. However, an inspection of those coefficients showed the latter *not* to be the case.

Table 2. Averages of the three best fitness values found for each polynomial degree p in the extended runs.

Polynomial degree			
1	2	3	4
23.24	37.23	47.91	49.33

Evidence in favor of the first interpretation can be found, on the other hand: As shown in Table 1, the increase in performance as the mutation rate is raised is stronger for large p values than for small ones, indicating that the larger p values require a more thorough inspection of the search space before the best solutions can be found. In order to test this tentative conclusion further, several extended runs were performed, the results of which are summarized in Table 2. Note that the extended runs differed somewhat in length: The number of evaluated individuals was on the order of 50,000

to 100,000. For this reason, the table shows an average of the three best results obtained for each p. In Table 2, the results clearly show an increase in performance as p is raised. Thus, it may be concluded that the choice of p strongly influences the quality of the results that can be achieved, but that the benefits of larger p values only become evident after the evaluation of a large number of individuals.

Finally, note that the chosen fitness measure (time spent in behavior B_1) does not lead to an incentive for the robot to explore the whole area. However, such solutions were indeed found by the EA in some runs. One example is shown in Fig. 2, with a fitness value equal to 44.57 found after the evaluation of 41,400 individuals.

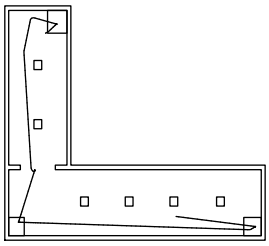

Fig. 2. The motion of one of the best robots found by the EA. The squares in the upper right, lower left, and lower right corners represent the charging stations.

Acknowledgments

The authors wish to thank the Carl Trygger foundation for financial support for this project.

References

1. Arkin, R.C., Behavior-based robotics, MIT Press, 1998
2. `http://www.borland.com/delphi`
3. McFarland, D. and Bösser, T. Intelligent behavior in animals and robots, MIT Press, 1993
4. Wahde, M., A method for behavioural organization for autonomous robots based on evolutionary optimization of utility functions, Journal of Systems and Control Engineering, 217, pp. 249–258, 2003
5. Maes, P., How to do the right thing, Journal of Connection Science, 1, No.3, pp. 291–323, 1989
6. Blumberg, B.M., Action selection in Hamsterdam: Lessons from ethology, In: From Animals to animats 3, Proc. of the 3rd Int. conf. on simulation of adaptive behavior (SAB94), MIT Press, 1994

Paper V

Behavioral selection in domestic assistance robots: A comparison of different methods for optimization of utility functions

In

Proceedings of the 2006 IEEE International Conference on Systems, Man, and Cybernetics, pages 4904–4909, Taipei, Taiwan, October 8–11, 2006.

Behavioral selection in domestic assistance robots: A comparison of different methods for optimization of utility functions

Jimmy Pettersson, David Sandberg, Krister Wolff, and Mattias Wahde

Abstract— **In this paper, the performance of several evolutionary algorithms (EAs), involving different operators, is investigated in connection with the utility function (UF) method, a method for generating behavioral organization (selection) systems in autonomous robots.**

The standard UF method, which uses an ordinary genetic algorithm (GA) with fixed-length chromosomes is compared with modified evolutionary methods in which the chromosomes are allowed to vary in size.

The results show that, contrary to expectations, the standard UF method performs at least as well as the modified methods, despite the fact that the latter have larger flexibility in exploring the space of possible utility functions. A tentative explanation of the results is given, by means of a simple, analytically tractable, behavioral organization problem.

I. INTRODUCTION AND MOTIVATION

As the percentage of elderly people increases, there will be an increase in the demand for domestic robots. These robots are expected to perform tasks such as notifying the owner about periodic events (e.g. intake of medicine), alerting in case of danger, providing transportation assistance, serving as an interface to nurses or doctors, vacuuming floors, and partly filling the need for social stimulation. Already today, there are numerous robots available, designed for domestic use, with vacuum cleaning and entertainment being the most common applications.

For domestic robots to become widely used, they need to become more intelligent and to be able to handle reliably a multitude of tasks. In behavior-based robotics (BBR) [1], the aim is to develop truly intelligent robots, capable of performing their duties in unstructured environments, through the combination of several low-level behaviors. In BBR, a robotic brain commonly consists of a repertoire of behaviors, and a system (the behavioral organizer) responsible for activating the correct behavior at any given time. Thus, a major issue in BBR is to determine *when* (i.e. under what circumstances) a behavior from the repertoire should be selected for activation. In order to solve this problem, several methods for behavioral organization (or behavior selection) have been proposed [2], [3].

However, most such methods require the user to specify numerous parameters by hand. One of the aims of the recently introduced utility function (UF) method [3] is to free the user from this burden. In the UF method, the relative merit (utility) of each behavior is obtained from a utility function (normally a polynomial). The utility functions determining the behavior selection in a given robotic brain are optimized by means of an evolutionary algorithm (EA).

With the UF method, selection among behaviors is simple, as soon as the utility functions have been obtained: At all times, the behavior associated with the highest utility value, as obtained from the utility function, is selected for activation.

In the UF method, each utility function depends on several state variables (see below), and the normal procedure is to use *complete* polynomials to represent the utility functions, i.e. polynomials containing all possible combinations of variables, up to a maximum degree d. For example, a complete utility polynomial $U_{\rm c}(s_1, s_2)$ of two variables and degree 2 would take the following form

$$\begin{aligned} U_{\rm c}(s_1, s_2) &= a_{00} + a_{10}s_1 + a_{01}s_2 + a_{20}s_1^2 + \\ &+ a_{11}s_1s_2 + a_{02}s_2^2, \end{aligned} \tag{1}$$

where the a_{ij} are constants to be determined by the EA. However, it is evident that the number of terms in such a polynomial representation grows rapidly with the number of variables n and the polynomial degree d, thus making the search space very large. In view of the long evaluation times (several seconds up to several minutes per individual) in many behavior selection problems, the need for efficient optimization methods (for the utility functions, in the case of the UF method) is evident.

An obvious alternative to using complete polynomials would be to use *non-complete* polynomials, i.e. polynomials lacking certain terms. A non-complete version ($U_{\rm nc}$) of $U_{\rm c}$ could, for example, take the form

$$U_{\rm nc}(s_1, s_2) = a_{00} + a_{20}s_1^2 + a_{11}s_1s_2, \tag{2}$$

or any other form involving a subset of the 6 terms in (1). Provided that the EA would be able (through crossover and mutations) to modify the exact nature of the terms in the non-complete polynomials, one might expect that such a representation would lead to faster optimization of the utility functions.

In this paper, the performance of different EAs will be studied for the case of a simple domestic exploration robot, which can serve as a metaphor for e.g. a vacuum cleaning robot. Specifically, the performance of a standard genetic algorithm (GA), which uses complete polynomials, is compared to the performance of several modified GAs, which use incomplete (or at least variable-structure) polynomials.

This work was supported by the Carl Trygger foundation.

The authors are affiliated with the Department of Applied Mechanics, Chalmers University of Technology, 412 96 Göteborg, Sweden. Corresponding author's e-mail: mattias.wahde@chalmers.se

II. THE UTILITY FUNCTION METHOD

In the UF method, each behavior in the behavioral repertoire $\{B_1, B_2, \ldots, B_N\}$ is associated with a utility function

$$U_i(\mathbf{s}, \mathbf{p}, \mathbf{x}), \quad i = 1, 2, \ldots, N, \tag{3}$$

where $\mathbf{s}$ is a set of external variables (such as external sensor readings), $\mathbf{p}$ is a set of internal physical variables (such as wheel encoder signals), and $\mathbf{x}$ is a set of internal abstract variables (corresponding to hormones in a biological organism).

The standard UF (hereafter: SUF) method sets up a population in which the chromosomes specify the coefficients of the utility functions for the individual in question. In the initial population, the coefficients are set to random values within a given range. Each individual is then evaluated, by running a simulation in a given arena while monitoring the performance (fitness) of the individual. The exact nature of the fitness measure will, of course, vary from problem to problem.

Being an arbitration method, i.e. one in which only a single behavior is allowed to be active at any given instant, the UF method selects, during the operation of a robot, the behavior whose utility function has the highest current value. The utility functions are optimized using an EA in which the fitness value only increases during the active period of specific behaviors, the *task* behaviors. Behaviors that do not modify the fitness value are referred to as *auxiliary* behaviors.

Once all individuals have been evaluated, new individuals are formed through the procedures of fitness-proportional selection, crossover, and mutation. The new generation thus formed is then evaluated in the same way as the first, etc. For a more detailed discussion of the UF method, see [3].

As mentioned above, in the SUF method, each utility function is represented by a complete polynomial, including all terms up to a pre-specified degree d. If the number of arguments, formed by the union of the sets $\mathbf{s}$, $\mathbf{p}$, and $\mathbf{x}$, is equal to n, it can be shown that the total number of terms in the polynomial is equal to

$$N_t = \binom{n+d}{d} = \binom{n+d}{n}. \tag{4}$$

An example with $n = 2$ and $d = 2$ is shown in (1). Denoting the full variable set $\{\mathbf{s}, \mathbf{p}, \mathbf{x}\}$ by $\mathbf{z}$, the utility functions can thus be written as sums of terms of the form

$$c z_1^{k_1} z_2^{k_1} \cdots z_n^{k_n}, \tag{5}$$

where c is a constant, and the k_i are non-negative integers. A utility function polynomial is said to consist of *unique terms* if any given exponent combination $(k_1, k_2, \ldots, k_n)$, is represented *at most* once in the polynomial. As indicated in connection with (1) above, a polynomial is said to be *complete* if all exponent combinations occur *exactly* once. Thus, in the case of the SUF method, the structure of the utility functions remains intact throughout the optimization process. However, using the methods defined in Sect. IV-B, structural modifications are introduced that enable a variable length of the polynomials, and thus the formation of non-complete polynomials.

III. SIMULATIONS

Simulations were made using software based on the UFLibrary software package [4]. Implemented in object-oriented Pascal, the UFLibrary provides a rapid way of generating behavior selection systems and provides a general framework for the UF method. In addition, the UFLibrary also contains methods for visualization, utilizing the OpenGL interface, methods for solving and integrating the dynamics associated with differentially steered robots, various sensor models etc.

A. Simulation setup

The arena used in the experiments is a small apartment having three rooms, as shown in Fig. 1. Specific areas of the apartment, for instance the bathroom, are considered to be off-limits, and have therefore been omitted from the model.

Cylindrical in shape, the simulated robot is differentially steered, powered by two DC motors, each modeled with a simple DC circuit. For sensing, the robot is equipped with five IR sensors placed symmetrically on the front of the robot in the directions $-60°$, $-30°$, $0°$, $30°$, and $60°$, respectively. Each sensor has an opening angle of 0.5 radians, a range of 0.5 m, and uses a model based on a ray-tracing technique as suggested in [5] for calculating the (diffuse) sensor reading.

In addition to the measurements provided by the IR sensors, the robot is capable of measuring the amount of energy available in its onboard battery. As the robot moves, the battery discharges as

$$\frac{\mathrm{d}E}{\mathrm{d}t} = -k_r - k_m|v| \,, \tag{6}$$

where k_r and k_m are positive constants and v is the speed (with sign) of the robot. The constants in (6) were set so that the battery would last approximately 25 s. Charging of the battery occurs with a constant rate (provided that the robot is currently executing the battery charging behavior, see below) as $\mathrm{d}E/\mathrm{d}t = k_c$, with k_c set so that an empty battery becomes fully charged after 10 s of continuous charging. In all runs, the initial energy level was set to half the battery's capacity.

In order for the robot to be able to explore the arena, three different behaviors were included in the behavioral repertoire, namely *straight-line navigation* (B1), *obstacle avoidance* (B2), and *battery charging* (B3). Each behavior is responsible for setting the command signals to the two DC motors in a certain way. If B1 is active, the motor command signals are set to positive values (equal in magnitude), causing the robot to travel forward in a straight line. In B2, the motor signals are set to be equal in magnitude but with opposite signs, making the robot turn on the spot, after an initial transient in case B1 was active before the activation of B2. The turning is aborted, and the motor signals are thus set to zero, when the sector in front of the robot is free from objects. When B3 is selected for activation, the motor signals are set to zero and the robot stops.

For simplicity, battery charging was modelled in a very simple way: In order to charge the battery, the robot simply has to activate B3, thus avoiding the need of additional behaviors for locating and approaching charging stations. Such behaviors would require additional sensors for identifying a charging station as well as methods for localization.

Since the task of the robot is to explore the environment, fitness is associated with the time spent in B1 (the task behavior). Every time the robot uses B1 it receives a fitness increase (upon leaving B1, or upon termination of the simulation) equal to $\max(0, t_1 - 1)$, where t_1 is the time spent in B1 since its latest activation. Thus, fitness is only increased if B1 is active for a period longer than one second. By incorporating this threshold in the fitness function, behavior dithering (or behavior mixing) is effectively eliminated. A situation involving behavior dithering could, for example, occur between behaviors B1 and B3. Due to the fact that battery charging occurs whenever B3 is active, without regard to the current position of the robot, the behavior selection could be optimized in such a way that B1 and B3 would be effectively mixed by toggling between B1 and B3 every time step. Due to the dynamical properties of the robot, this would enable the robot to travel forward at low speed while, at the same time, charging its onboard battery. This situation is clearly unrealistic and was eliminated by introducing the one-second threshold in the fitness function.

For the three behaviors B1, B2, and B3, the utility functions were specified as

$$\begin{aligned} U_1 &= U_1(s_1, s_2, s_3, s_4, s_5, p_1) \\ U_2 &= U_2(s_1, s_2, s_3, s_4, s_5, p_1, x_1) \\ U_3 &= U_3(p_1, x_2) \end{aligned} \tag{7}$$

where s_i are the readings of the five IR sensors, p_1 is the amount of energy stored in the onboard battery, x_1 is a hormone variable corresponding to *fear*, and x_2 is a hormone variable corresponding to *inverse satiation*. The total number of polynomial terms in a chromosome representing utility functions for the SUF method can easily be calculated, using (4), as

$$N_t = \binom{9}{3} + \binom{10}{3} + \binom{5}{3} = 214\ . \tag{8}$$

Collisions with objects in the environment, as well as battery depletion, cause the simulation to be aborted. Thus, in order for the robot to receive high fitness values, the auxiliary behaviors B2 and B3 must be activated every now and then for collision avoidance and battery charging, respectively.

Since the focus of this paper is to compare different evolutionary approaches to the generation of utility-function based behavioral organization systems, rather than the application *per se*, several simplifications have been made: As indicated above, the behaviors B1-B3 are very simple, and so is the arena. Furthermore, the exploration carried out by the robot does not include any measure of the covered area and neither does it depend on any map. In addition, the rates of battery charging and discharging have been set to very high values, in order for the robot to be forced to activate all three behaviors even during a rather short simulation. Finally, noise was omitted from the analysis, and all simulations were performed using the same initial setup (i.e. placement and orientation of the robot in the arena).

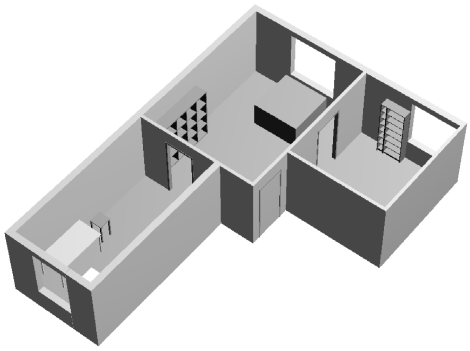

Fig. 1. Arena used in the simulations (view from above).

IV. METHODS

A. Standard UF method

The UFLibrary implements the SUF method, with a GA in which each gene in the chromosome encodes a complete polynomial, representing the utility function associated with a behavior. By default, a single-point crossover operator is used that swaps genes between two selected parents, as exemplified in the top panel in Fig. 2. Hence, the crossover operator effectively swaps entire polynomials, keeping the structure of each polynomial intact. The mutation operator used in this method modifies the coefficients of the polynomial terms, but not the exponents of the variables, again keeping the structure of the polynomials intact.

B. Modified methods

Several modified methods, involving evolutionary operators different from those used in the SUF, were tried, namely

- M_1 Initialization[1] by generating $N = N_t$ (see (4)) random polynomial terms up to a given maximum degree d. In M_1 the initial polynomials are complete, i.e. there exists one, and only one, term (and thus one coefficient) for each exponent combination $(k_1, \ldots, k_n)$, see (5).
- M_2 Initialization by generating polynomials with $N \in [1, N_t]$ random terms up to a given maximum degree d. Thus, in this method, the initial population will contain utility functions with fewer terms than in M_1. Furthermore, in M_2, uniqueness is not enforced even in the initial population. Thus, the polynomials *may* contain more than one term for a given exponent combination $(k_1, \ldots, k_n)$ (again, see (5)).
- M_3 As M_1, but with the additional operation of simplifying the polynomials resulting from the crossover operation to ensure that the polynomials consist of unique terms as defined in Sect. II. Thus, if, for example, an offspring chromosome contains the two terms $c_1 z_1^{k_1} \cdots z_2^{k_n}$

[1] i.e. generation of the initial population.

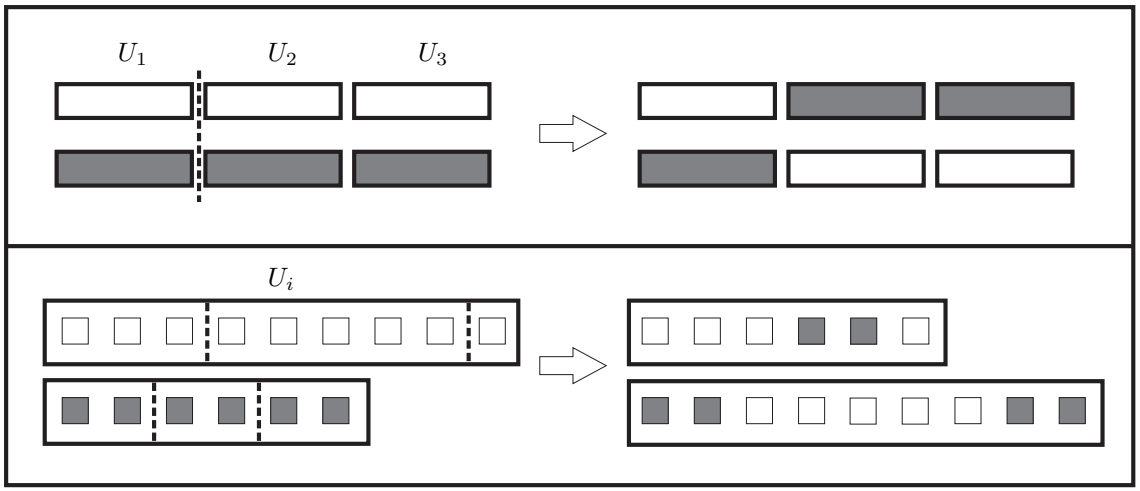

Fig. 2. Crossover operators used in the standard UF (SUF) method (top panel) and in the modified methods (bottom panel). Genetic material associated with the two parent chromosomes is indicated with the colors white (first parent) and gray (second parent), and dashed lines indicate the randomly chosen crossover points. In the top panel, the crossover operator effectively swaps entire polynomials between the two parents as the genes (three in this case) encode the polynomials associated with each behavior. In the modified methods, the crossover operator swaps segments consisting of polynomial terms, which are illustrated as small boxes in in the bottom panel. Note that, for the modified methods, even if the parent chromosomes contain unique polynomial terms, the offspring may contain non-unique terms as a result of crossover.

and $c_2 z_1^{k_1} \cdots z_2^{k_n}$ (with the same exponent combination $(k_1, \ldots, k_n)$), they are fused into *one* term with coefficient $c = c_1 + c_2$.

M_4 As M_2, but with the same simplification procedure as in M_3.

M_5 As M_1, but with an additional mutation operator that randomly selects one existing term and generates a new coefficient *and* new exponent values (with a maximum degree d). Hence, c, k_1, k_2,..., k_n in (5), for the chosen polynomial term, are assigned new random values.

M_6 As M_2, but with the same additional operator as in M_5.

In the methods M_1–M_6, a two-point crossover operator was used, as illustrated in the bottom part of Fig. 2. Contrary to the crossover operator used in the SUF method, in which entire polynomials are swapped, the two-point crossover operates on the genetic material contained *in* the polynomials, i.e. on the level of polynomial terms. For each polynomial in the chromosome, two segments of random lengths are swapped between the two participating individuals. This operator effectively swaps sequences of polynomial terms, changing the structure of the involved polynomials.

Mutation was used in the same way as in the SUF method, i.e. randomly changing the coefficients (c) of the polynomial terms, leaving the combination of exponents unchanged, except in M_5 and M_6, where exponents could be modified as well (see above).

In both the SUF method and in the modified methods (M_1–M_6), generational replacement and elitism were used in the implementation of the algorithms. Thus, when forming the new generation through selection, crossover, and mutation, the best individual in the last generation was transferred unchanged into the new generation.

V. RESULTS

In all runs, a population size of 100 individuals was used and the EAs were normally run for 50 generations (5 000 evaluated individuals). The maximum simulation time was set to 100 s, exceeding the time it takes for a fully charged battery to deplete. Selection was performed using tournament selection with a tournament size of 5 and a probability of selecting the best individual set to 0.7. The probability of crossover was set to 0.5.

In the runs involving the SUF method, a fixed polynomial degree of 3 was used, as suggested by the results in [6]. For the runs made with the modified methods, the same degree was used as an upper bound of the generated polynomial terms.

In the SUF method, a mutation rate of $3/N_t$ was chosen, based on the results reported in [6]. Note that N_t is the number of polynomial terms in a complete polynomial, as defined in (4).

In the methods M_1–M_6, a variable mutation rate of $1/n_c$ was used[2], where n_c denotes the (variable) total number of terms in the utility functions of a given individual. An exception was the exponent mutations in M_5 and M_6, which occurred with the rate $1/(5n_c)$.

Due to the stochastic nature of EAs, several runs should be made in order to evaluate the performance of any given method. Here, 10 runs were made for each method, and the average $\bar{f}$ of the best fitness f^{best} obtained in each run was formed. Thus, for any given method,

$$\bar{f} = \frac{1}{10} \sum_{i=1}^{10} f_i^{\text{best}}. \tag{9}$$

[2]Some test runs were carried out using a mutation rate of $3/n_c$ (for M_1–M_6), but these runs did not give better results.

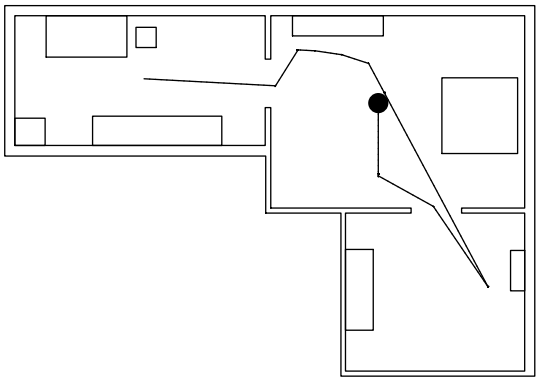

Fig. 3. Typical path taken by one of the best robots. The filled circle marks the initial position of the robot. In the particular case illustrated here, the behavior selection system was optimized using method M_1 and was obtained after the evaluation of 450 individuals.

As seen from the results in Table I, the value of $\bar{f}$ in methods M_1 and M_5 is similar to that of the SUF method whereas M_2, M_4, and M_6 perform significantly worse (on average). M_3 achieved intermediate values of $\bar{f}$.

In terms of the maximum fitness attained in any of the 10 runs ($f_{\max} = \max_i f_i^{\text{best}}$), all methods performed approximately equally well, i.e. all methods were capable of finding at least one good solution. If the EAs were allowed to run for longer periods of time (more than 5 000 evaluated individuals), all methods eventually managed to escape any local optima. The path travelled by one of the best robotic brains obtained using the M_1 method is shown in Fig. 3.

TABLE I

RESULTS FROM RUNS USING THE STANDARD UF (SUF) METHOD AND THE SIX METHODS DEFINED IN SECT. IV-B (M_1–M_6). A TOTAL OF 10 RUNS WERE MADE FOR EACH METHOD. IN EACH RUN, A TOTAL OF 5 000 INDIVIDUALS WERE EVALUATED. $f_{\max}$ IS THE MAXIMUM FITNESS ACHIEVED, $\bar{f}$ IS THE AVERAGE OF THE MAXIMUM FITNESS OVER THE 10 RUNS, AND $C_{95\%}$ IS THE CORRESPONDING 95% CONFIDENCE INTERVAL.

Method	$f_{\max}$	$\bar{f}$	$C_{95\%}$
SUF	59.07	55.75	1.92
M_1	58.52	54.61	2.00
M_2	54.20	40.87	8.53
M_3	55.92	49.63	5.01
M_4	58.36	40.96	10.00
M_5	58.15	55.17	1.36
M_6	54.87	40.86	6.87

VI. DISCUSSION

Looking at the averages $\bar{f}$ of the best results obtained in the 10 runs for each method, the most striking observation is the fact that the methods M_2, M_4, and M_6 perform significantly worse than the SUF method. By contrast, M_1, M_3, and M_5 which, like the SUF method, all start from complete polynomials, are comparable in performance to the SUF method (with the possible exception of M_3, which performs slightly worse).

This was not anticipated since M_2, M_4, and M_6 all start from utility functions with fewer polynomial terms than the SUF method and thus, effectively, are able to carry out their search in a lower-dimensional space, and with flexibility as to which terms should be included in the utility function polynomials. Having obtained these results, one might be tempted to conclude that all polynomial terms are needed in order to find a good solution or, in other words, that only complete polynomials, as defined in Sect. I, will lead to proper behavior selection. However, this is *not* the case since it is *possible* (albeit slower) to find good solutions even if the polynomial degree d is lowered to 2, say.

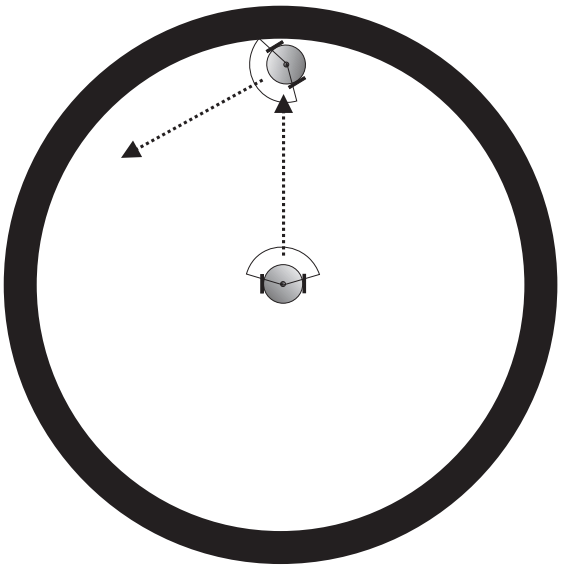

Fig. 4. A robot in an arena with a circularly symmetric obstacle. The robot is differentially steered and equipped with a single wide-range proximity sensor.

How, then, can the difference in performance be accounted for? Consider the simple toy problem in which a robot is placed in a circularly symmetric arena as shown in Fig. 4. The robot is assumed to be equipped with the two behaviors *straight-line navigation* (B_1) in which both motors receive equal signals, and *turning* (B_2), in which the motor signals are equal in magnitude but have opposite sign, making the robot turn without moving its center-of-mass. It is further assumed that the robot moves slowly, that it can stop almost instantaneously, and that it has an unlimited energy supply. The proximity sensor is assumed to give a signal $s = 1$ if an obstacle is detected and $s = 0$ otherwise. In order for the robot to achieve collision-free navigation, maximizing the distance travelled in a given time, it should, of course, keep B_1 active unless the sensor detects an obstacle, in which case the robot should turn until the obstacle is no longer detected, at which point the execution of B_1 should be resumed (see Fig. 4). Now, in order to solve this simple problem using the UF method, one can without restriction set one of the utility functions (U_1, say) to 0. For U_2, an ansatz of the form

$$U_2 = a_0 + a_1 s + a_2 s^2 + \ldots + a_d s^d, \qquad (10)$$

can be used. For this simple problem, $d = 1$ would be sufficient. However, assuming that the level of complexity

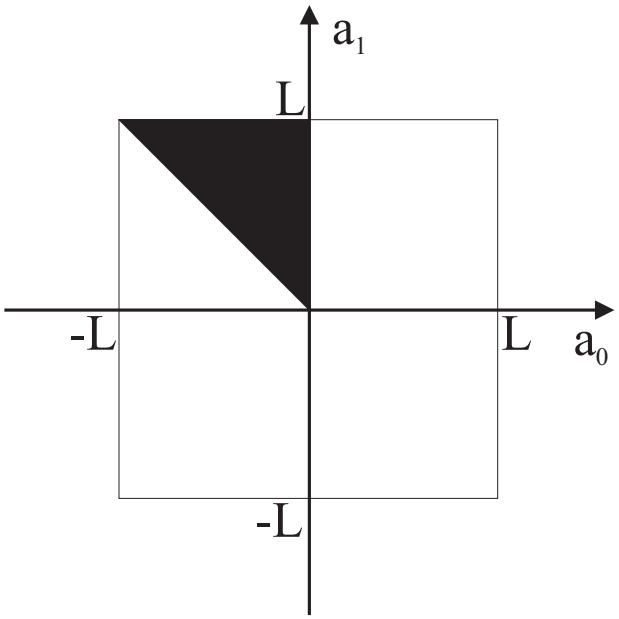

Fig. 5. The integral I_2. The region C corresponds to the shaded part of the space.

of the problem is unknown (as is normally the case, in more realistic applications), one must select a suitable value of d, and then search the space of possible solutions. The question is, then: What fraction of the search space will give correct solutions (as described above), as a function of the dimensionality ($D = d + 1$) of the search space? In this problem, correct solutions occur if $U_1 > U_2$ for $s = 0$ and $U_2 > U_1$ for $s = 1$. Hence, the conditions

$$a_0 < 0 \tag{11}$$

and

$$\sum_{i=0}^{d} a_i > 0 \tag{12}$$

must be satisfied in order for a given solution (i.e. a set of polynomial coefficients) to be correct. Assuming that all parameters a_i lie in the interval $[-L, L]$, the fraction of the search space meeting these criteria can be formulated as an integral, according to

$$I_D = (2L)^{-D} \int_{\{a_i\} \in C} \mathrm{d}A \ , \tag{13}$$

where $\mathrm{d}A$ denotes $\mathrm{d}a_0 \mathrm{d}a_1 \cdots \mathrm{d}a_d$, and the region C is defined by the criteria given in (11) and (12). As an example, the integral I_2 is illustrated in Fig. 5. As is evident from the figure, $I_2 = 1/8 = 0.125$. It is straightforward to compute the integral for any value of D, and the resulting values are shown in Table II. As is evident from the table, the fraction I_D of the search space giving correct solutions *increases* as the number of dimensions is raised. Put differently, the results in Table II exemplify the fact that an increase in the size of the search space does not always make it more difficult to find the correct solution. Instead it is the *fraction* of the search space containing correct solutions that matters.

TABLE II

THE INTEGRAL I_D SHOWN FOR VARIOUS VALUES OF THE DIMENSION (D) OF THE SEARCH SPACE. NUMERICAL VALUES WERE CALCULATED USING A MONTE CARLO METHOD WITH A TOTAL OF AROUND 10^9 SAMPLED POINTS FOR EACH VALUE OF I_D.

D	I_D	D	I_D	D	I_D
2	0.1250	6	0.1783	10	0.1948
3	0.1458	7	0.1837	11	0.1974
4	0.1615	8	0.1881	100	0.2327
5	0.1711	9	0.1918	214	0.2382

Finally, for the simulations, it should be noted that the maximum attained fitness values $f_{\max}$ shown in Table I, are quite similar. This is due to the fact that, during the 100 s simulations used here, there is a limit to the amount of fitness that can be attained. An estimate of the maximum attainable fitness (the details of which will not be given here) leads to a value of around 64, rather close to the maximum fitness values reached by all methods. Had the problem been more complex, involving, say, five or 10 behaviors, it is likely that the difference in performance between, on the one hand the SUF method, M_1, M_3, and M_5 and, on the other hand, M_2, M_4, and M_6 would be evident also in the values of $f_{\max}$.

VII. CONCLUSION

The main conclusion of this work is that, at least for certain problems with fairly low level of complexity, the performance of the standard UF method is equal to, or better than, the performance of the modified methods, despite the more rigid structure of the utility function polynomials in the SUF method. A possible explanation is the fact that, in some problems, an effective increase in the size of the search space does *not* necessarily make the problem of finding good solutions harder. This has been illustrated by means of an analytically tractable toy problem, for which it was shown that the probability of finding a correct solution *increases* with the size of the search space.

VIII. ACKNOWLEDGMENTS

The authors gratefully acknowledge financial support from the Carl Trygger foundation for this project.

REFERENCES

[1] R. Arkin, *Behavior-based robotics*. MIT Press, 1998.
[2] P. Pirjanian, "Behavior coordination mechanisms – state-of-the-art," Institute of Robotics and Intelligent Systems, USC, Los Angeles, Tech. Rep., 1999.
[3] M. Wahde, "A method for behavioural organization for autonomous robots based on evolutionary optimization of utility functions," *Journal of Systems and Control Engineering*, no. 217, pp. 249–258, 2003.
[4] http://www.me.chalmers.se/~mwahde/robotics/UFMethod/UFLibrary/Demo.html.
[5] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," in *Lecture notes in computer science*, vol. 929. Springer-Verlag GmbH, 1995, pp. 704–720.
[6] M. Wahde, J. Pettersson, H. Sandholt, and K. Wolff, "Behavioral selection using the utility function method: A case study involving a simple guard robot," in *Proc. of the* 3rd *Int. Symp. on Autonomous Minirobots for Research and Edutainment (AMiRE 2005)*, 2005, pp. 261–266.

Paper VI

Structural evolution of central pattern generators for bipedal walking in 3D simulation

In

Proceedings of the 2006 IEEE International Conference on Systems, Man, and Cybernetics, pages 227–234, Taipei, Taiwan, October 8–11, 2006.

Structural Evolution of Central Pattern Generators for Bipedal Walking in 3D Simulation

Krister Wolff, Jimmy Pettersson, Almir Heralić, and Mattias Wahde

Abstract— **Anthropomorphic walking for a simulated bipedal robot has been realized by means of artificial evolution of central pattern generator (CPG) networks. The approach has been investigated through full rigid-body dynamics simulations in 3D of a bipedal robot with 14 degrees of freedom. The half-center CPG model has been used as an oscillator unit, with interconnection paths between oscillators undergoing structural modifications using a genetic algorithm. In addition, the connection weights in a feedback network of predefined structure were evolved. Furthermore, a supporting structure was added to the robot in order to guide the evolutionary process towards natural, human-like gaits. Subsequently, this structure was removed, and the ability of the best evolved controller to generate a bipedal gait without the help of the supporting structure was verified. Stable, natural gait patterns were obtained, with a maximum walking speed of around 0.9 m/s.**

I. INTRODUCTION AND MOTIVATION

The great interest in humanoid robots during the last decade is motivated by the many advantages of bipedal robots over wheeled robots. First of all, humanoid robots (and bipedal robots in general) are able to move in areas that are inaccessible to wheeled robots, such as staircases and rugged outdoor terrain. In addition, their human-like shape allows such robots to function in constructed environments, such as homes or industries which, naturally, are adapted to people. Furthermore, recent studies [1], [2], [3] have claimed that people are more comfortable interacting with a robot with an approximately human shape, rather than a tin can-like wheeled robot.

However, an obvious problem confronting humanoid robotics is the generation of stable gaits. Whereas wheeled robots normally are statically balanced and remain upright regardless of the torques applied to the wheels, a humanoid robot must be actively balanced, particularly if it is to execute a human-like, dynamic gait. Several methods for generating bipedal gaits have been proposed in the literature. An important example is the ZMP method [4], where control torques are generated in order to keep the zero-moment point within the convex hull of the support area defined by the feet.

However, the success of gait generation methods based on classical control theory, such as the ZMP method, relies on the calculation of reference trajectories for the robot to follow. That is, trajectories of joint angles, joint torques, or the centre-of-mass of the robot are calculated so as to satisfy the ZMP constraint [5], [6]. When the robot is acting in a well-known constructed environment, the ZMP method should work well. When acting in a dynamically changing real world environment, however, the robot will encounter unexpected situations which cannot all be accounted for beforehand. Hence, reference trajectories can rarely be specified under such circumstances. To address this problem, there has recently been a movement in the robotics community towards alternative, biologically inspired control methods. Such methods do not, in general, require any reference trajectory. Typically, robotics researchers employ bio-inspired control strategies based on artificial neural networks (ANNs) [7], [8] or central pattern generators (CPGs) [9]. Often some kind of evolutionary algorithm (EA) is utilized for the design of the controller [10], [11], [12], [13], and [14].

Clearly, walking is a rhythmic phenomenon, and many biological organisms are indeed equipped with CPGs, i.e. neural circuits capable of producing oscillatory output given tonic (non-oscillating) activation [15]. CPGs have been studied in several simple animals, such as the lamprey [16] for which mathematical models have been developed as well [17], [18]. CPGs have also been studied in more complex animals, such as cats and primates ([19], [20], [21]), and there are also observations that support the notion of CPGs in humans. For example, treadmill training of patients with spinal cord lesions is assumed to rely on the adequate activation of a CPG [21].

Developing artificial counterparts to biological CPGs, with the aim of generating robust gaits for bipedal robots, is an active field of research. In seminal works by Taga *et al.*, [9], [22], a gait controller based on the half-center CPG model (see below) has been investigated. It was demonstrated in a 2D simulation of a five-link biped that the controller made the robot robust against physical perturbations [9]. Furthermore, obstacle avoidance through regulation of the step length was realized [22].

Shan *et al.* [11] generated bipedal walking in a 2D simulation using CPGs. A multi-objective genetic algorithm was used to optimize the synaptic weights in a network composed of nine CPG units. Reil and Husbands [23] used genetic algorithms (GAs) to optimize fully connected recurrent neural networks (RNNs), which were used as CPGs to generate bipedal walking in 3D simulation. They used a GA, with a real-valued encoding scheme, to optimize weights, time constants, and biases in fixed architecture RNNs. Their biped model had six degrees-of-freedom (DOFs), and consisted of a pair of articulated legs connected with a link. The resulting CPGs were capable of generating bipedal, straight-line walking on a planar surface. Furthermore, simple sensory input to

The authors are affiliated with the Department of Applied Mechanics, Chalmers University of Technology, 412 96 Göteborg, Sweden. Corresponding author's e-mail: krister.wolff@chalmers.se

locate a sound source was integrated to achieve directional walking.

CPGs have desirable properties, such as intrinsic aptness for the formation of periodic output patterns and adaptation to the environment through entrainment, for the generation of gaits and other types of repetitive and stereotypic motions.

Manually tuning the parameters of the CPGs and defining the feedback and interconnection paths in an optimal way is a daunting task. In many cases reported in the literature, e.g. [22], [24], [25], [26], and [27], the design of CPG networks has commonly been carried out in an intuitive manner; a time-consuming and difficult process which may lead to sub-optimal performance. Even in cases where GAs have been applied, as in several of the references mentioned above, the approach has generally been restricted to parametric optimization in a network of fixed architecture.

In this paper, the problem of generating both the structure, i.e. the network feedback and interconnection paths, and the parameters of a CPG network controlling a fully three-dimensional, simulated bipedal robot with 14 DOFs will be considered, using a GA as the optimization method. The half-center CPG model, as originally proposed by Matsuoka [28], will be adopted as the oscillator unit. A challenging problem, which is seldom mentioned (the papers by Paul and Bongard [29] are an exception), is the fact that, while biological organisms have developed their walking patterns (and, indeed, other behaviors as well), over long periods of simultaneous evolutionary optimization of both body and brain, in robotics one attempts instead to provide an already fixed body structure with a brain capable of generating a bipedal gait. This poses many problems for a GA-based approach. For example, if only the distance covered is used as the fitness measure, a common result is to find individuals that simply throw themselves forward, rather than walking; Walking would certainly yield a higher fitness value, yet this solution may be very hard to find, given the readily accessible local optimum found by those individuals throwing their body forward. Thus, the evolutionary process grinds to a halt almost immediately. Of course, this type of solution can be avoided simply by adding constraints on body posture as part of the fitness measure. However, such constraints must often be added in an *ad hoc* manner, and they often lead to results (such as non-natural gaits) that are undesirable. Rather than changing the fitness measure, one may attempt to change the body of the robot. Evolving an upright, bipedal gait from, say, an initial population of crawling individuals would perhaps be infeasible. However, another option, which will be considered in this paper, is to add a supporting structure to the robot, helping it to balance as it starts to walk. Some different strategies for subsequently removing this support, while maintaining a dynamically stable gait, will then be investigated.

II. CENTRAL PATTERN GENERATORS

A. Models from biology

From biological studies, three main types of neural circuits for generating rhythmic motor output have been proposed,

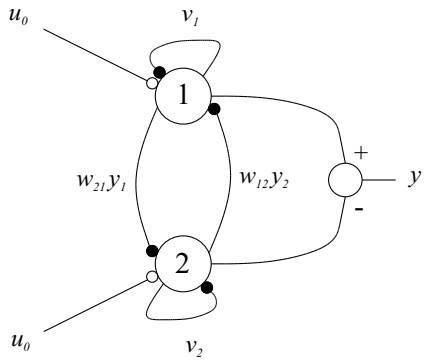

Fig. 1. A half-center model (Matsuoka) oscillator unit. Excitatory connections are indicated by open circles, and inhibitory connections are indicated by filled disks.

namely the *closed-loop model*, the *pacemaker model*, and the *half-center model*. The two former models are described in [30]. The half-center model, which will be considered in this paper, was proposed to account for the alternating activation of flexor and extensor muscles of the limbs of a cat during walking. The basis of this model is the classical experiments reported by Brown from 1911 [31] and 1912 [19]. Each pool of motor neurons for flexor or extensor muscles is activated by a corresponding half-center, or pool, of interneurons. Another set of neurons provides for a steady excitatory drive to these interneurons. Between each pool of interneurons are inhibitory connections which ensure that, when one pool is active, the other is suppressed. Matsuoka [28] analyzed the mutually inhibiting neurons and found the conditions under which the neurons generated oscillations.

B. Mathematical formulation of the CPG model

Commonly, a CPG is computationally modeled as a network of identical systems of differential equations, which are characterized by the presence of attractors[1] in the phase space [32]. Usually, a periodic gait of a legged robot is a limit cycle attractor, since the robot periodically returns to (almost) the same configuration in phase space.

Each node in the network is referred to as a neuron, or cell. The half-center model mentioned above is commonly adopted as the biological foundation for a rhythm generator, see e.g. [9], [11], [22], [24]. The neurons in the half-center model are described by the following equations [9]:

$$\tau_u \dot{u}_i = -u_i - \beta v_i + \sum_{j=1}^{n} w_{ij} y_j + u_0, \quad (1)$$

$$\tau_v \dot{v}_i = -v_i + y_i, \quad (2)$$

$$y_i = \max(0, u_i), \quad (3)$$

where u_i is the inner state of neuron i, v_i is an auxiliary variable measuring the degree of self-inhibition (modulated by the parameter β) of neuron i, τ_u and τ_v are time constants, u_0 is an external tonic (non-oscillating) input, w_{ij} are the weights connecting neuron j to neuron i, and, finally, y_i

[1] Attractors are bounded subsets of the phase space, to which regions of initial conditions converge as time evolves.

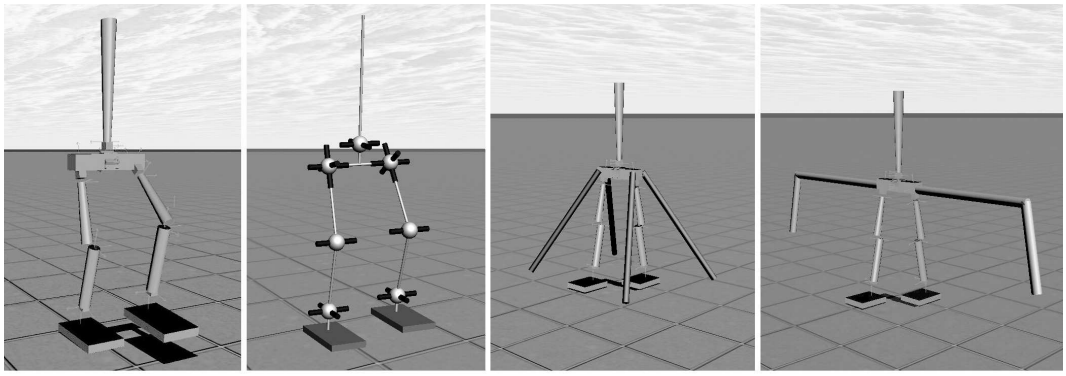

Fig. 2. The leftmost panel shows the simulated robot, and the second panel from the left shows its kinematics structure with 14 DOF. The two right panels show the robot with its supporting structure, the left one having four contact points, while the right one has two contact points.

is the output of neuron i. Two such neurons arranged in a network of mutual inhibition (a half-center model), as shown in Fig. 1, form an oscillator, in which the amplitude of the oscillation is proportional to the tonic input u_0. In addition, if an external oscillatory input is applied, the oscillator will lock to the frequency of the input. If the input is removed, the oscillator smoothly returns to the original frequency.

III. METHOD

In this section the physical simulation environment, the CPG network structure, the feedback paths, and the evolutionary algorithm will be described.

A. Dynamical simulation

A fully three-dimensional bipedal robot with 14 degrees of freedom, shown in the leftmost panel of Fig. 2, was used in the simulation experiments. The simulated robot weighs 7 kg and is 0.75 m tall. The distance between the ground and the hips is 0.45 m. As shown in the second panel from the left in Fig. 2, the waist has 2 DOFs, each hip joint has 3 DOFs, the knee joints have 1 DOF each, and the ankle joints have 2 DOFs each. The CPG network generates torques, which are applied to their respective joints in order to control the robot. To guide the evolution towards a natural biped gait, the robot has been fitted with two different mass-less posture-support structures, as depicted in the two right panels of Fig. 2.

The simulations were carried out using the EvoDyn simulation library [33], which was developed at Chalmers University of Technology. Implemented in object-oriented Pascal, EvoDyn is capable of simulating tree-structured rigid-body systems and runs on both Windows and Linux platforms. Its dynamics engine is based on a recursively formulated algorithm that scales linearly with the number of rigid bodies in the system [34]. For numerical integration of the state derivatives of the simulated system, a fourth order Runge-Kutta method is used. Visualization is achieved using the OpenGL library.

B. CPG network

In the CPG network, which is responsible for the generation of motions, each joint is assigned a specific half-center oscillator consisting of two neurons; a flexor neuron and an extensor neuron. The overall structure of the CPG network is depicted in Fig. 3.

In order to reduce the size of the search space for the GA, symmetry constraints were added, motivated by the fact that, modulo a phase difference, the movements of the left and right parts of the robot are symmetrical. Hence, the structure of the CPGs on the right side of the robot mirrors that of the left side. For example, the connection weight between the left hip and the left knee is equal in value to the weight connecting the right hip to the right knee. In the network the hip CPG on a given side responsible for rotation in the sagittal plane, can be connected to all the other ipsilateral[2] joint CPGs, the corresponding contralateral hip CPG, and the waist CPGs as well. Note, however, that this hip joint CPG can only receive connections from the corresponding contralateral hip joint CPG. Thus, the total number of connections to be determined sums up to 32, see also Fig. 3 for the details of inter-CPG connectivity.

For reasons that will be discussed in Sect. IV, the internal parameters of the individual two-neuron CPGs were set to fixed values, generating a frequency approximating that of a normal walking pattern. The CPG parameters were set to the following values for all CPGs, except for the knee joint CPGs and the waist joint (rotation in the sagittal plane) CPG: $\tau_u = 0.025, \tau_v = 0.3, \beta = 2.5, u_0 = 1.0, w_{12} = w_{21} = -2.0$. In analogy with human gait, the knee joint CPGs and the waist joint CPG oscillate with double frequency, compared to the other CPGs. Thus, for these joints' CPGs the $\tau_{u,v}$ values were set to half of the value for the other CPGs.

C. Genetic algorithm

A GA has been used for optimizing the structure of the CPG network controlling the movements of the robot. As mentioned above, the number of evolvable connections equals 32. In the GA, two chromosomes were used for the CPG network: one binary-valued chromosome determining the presence or absence of each of the 32 connections,

[2]The term *ipsilateral* refers to the same side of the body, and is thus the opposite of *contralateral*.

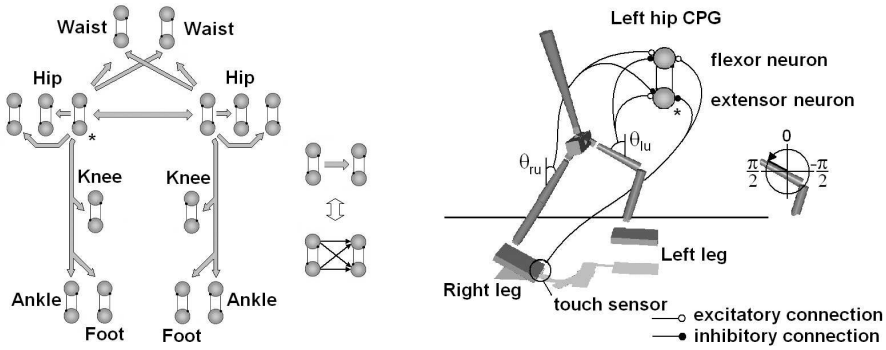

Fig. 3. Left panel: The structure of the CPG network used in the simulations. The connections are represented, in a slightly simplified way, by the arrows in the figure. Note that an arrow indicates the *possibility* of full connection, as shown in the rightmost part of the panel. Right panel: The robot depicted with a single hip joint CPG with feedback paths, and a possible choice of connection types. In the situation shown in the figure, the flexor neuron is responsible for rotating the hip joint in the counterclockwise direction.

and one real-valued chromosome determining the parameter values for those connections which are actually used in a given individual. Along with the CPG network structure, the feedback network can also be evolved using a third (real-valued) chromosome, which includes 20 parameters determining the sign and the strength of the different feedback paths (see below).

The fitness measure was normally taken as the distance walked by the robot in the initial forward direction, decreased by the sideways deviation. Some attempts were made to use a multi-objective GA (MOGA), with three populations evolving simultaneously towards different fitness measures, namely (1) the distance walked, (2) the number of times an entire foot touched the ground, and (3) the sum of the shortest distance (of the two legs) in the vertical direction between the hips and the knees over all time steps. Criterion (2) was included in order to suppress running behavior where the feet hardly touched the ground, which affected the frontal plane balance. The last criterion should promote an upright posture. However, the MOGA did not lead to any significant improvement. Hence, a standard GA was eventually chosen for the simulation experiments.

For selection, a tournament scheme of size 8 was adopted. The individuals were randomly picked from the population to compete against each other, based on the fitness values. The individual with highest fitness value was then selected with a probability equal to 0.75. After selection, the mutation operator was applied, randomly changing a gene's value with the probability $10/N$, with N being the total number of genes of the individual.

D. Feedback

In order to guide the evolutionary process towards an upright and stable bipedal gait, feedback was introduced measuring the waist angle, thigh angle, and lower leg angle, all relative to the vertical axis. Also, a touch sensor in each foot was introduced in the simulation. This sensor is used both to produce a feedback signal and to enable, or prohibit, feedback to a certain joint CPG during a specific phase, e.g. the stance phase. The feedback was incorporated into the CPGs by adding an extra term to (1), which then becomes

$$\tau_u \dot{u}_i = -u_i - \beta v_i + \sum_{j=1}^{n} w_{ij} y_j + u_0 + f \qquad (4)$$

where f is the feedback. In this setup, the feedback structure is decided upon beforehand. However, the actual type of the connection (inhibitory or excitatory) and the strength of the feedback are determined by the GA. An example of the feedback paths connected to the hip joint (and a possible choice of connection type) is shown in the right panel of Fig. 3. In detail, the feedback paths are given by the following equations:

$$\begin{aligned}
\text{waist}_1 &= c_1 w_{1_{f,e}} \theta_w + p_r[w_{1_{f,e}}(c_2\theta_{r,u} + \theta_{r,l})] \\
&\quad + p_l[w_{1_{f,e}}(c_2\theta_{l,u} + \theta_{l,r})] && (5)\\
\text{waist}_2 &= w_{2_{f,e}}\theta_{l,u} - w_{2_{f,e}}\theta_{r,u} && (6)\\
\text{hip}_{1,l} &= w_{3_{f,e}}\theta_{l,u} - w_{3_{f,e}}\theta_{r,u} + c_3 w_{3_{f,e}} e_r && (7)\\
\text{hip}_{2,l} &= e_l[w_{4_{f,e}}\theta_{l,u}] && (8)\\
\text{hip}_{3,l} &= w_{5_{f,e}}\theta_{\text{hip}_{3,l}} && (9)\\
\text{knee}_l &= e_r[w_{6_{f,e}}\theta_{r,l}] && (10)\\
\text{ankle}_l &= e_l[w_{7_{f,e}}\theta_{l,u}] && (11)\\
\text{foot}_l &= e_l[w_{8_{f,e}}\theta_{l,l}] + e_r[c_4 w_{8_{f,e}}\theta_{r,l}] && (12)\\
\text{hip}_{1,r} &= w_{3_{f,e}}\theta_{r,u} - w_{3_{f,e}}\theta_{l,u} + c_3 w_{3_{f,e}} e_l && (13)\\
\text{hip}_{2,r} &= e_r[w_{4_{f,e}}\theta_{r,u}] && (14)\\
\text{hip}_{3,r} &= w_{5_{f,e}}\theta_{\text{hip}_{3,r}} && (15)\\
\text{knee}_r &= e_l[w_{6_{f,e}}\theta_{l,l}] && (16)\\
\text{ankle}_r &= e_r[w_{7_{f,e}}\theta_{r,u}] && (17)\\
\text{foot}_r &= e_r[w_{8_{f,e}}\theta_{r,l}] + e_l[c_4 w_{8_{f,e}}\theta_{l,l}] && (18)
\end{aligned}$$

where waist_1 is the joint rotating the torso in the sagittal plane, and waist_2 denotes the joint responsible for frontal plane rotation. Likewise, hip_1 rotates the leg in the sagittal plane, while hip_2 rotates the leg in the frontal plane. The hip_3 joint is responsible for rotation around the vertical axis. The strength and the sign of the feedback paths are determined by the 16 weights $w_{i_{f,e}}$, along with the four additional constants c_i.

Since each joint CPG consists of two units, a flexor neuron and an extensor neuron, two different connection weights are used, w_{i_f} and w_{i_e}, respectively, as indicated in the equations. Apart from this, the feedback paths for the two CPG neurons are identical. Hence, the total number of parameters to be determined sums up to 20.

Furthermore, θ_w is the torso angle in the sagittal plane, $\theta_{l,u}$ is the left upper leg (thigh) angle, and $\theta_{l,l}$ is the left lower leg angle. Correspondingly, the angles for the right leg are denoted $\theta_{r,u}$ and $\theta_{r,l}$. The angle of the hip_3 joint in the local frame is denoted $\theta_{\text{hip}_{3,i}}$, where *i* is either r (right) or l (left). Finally, e_i and p_i stand for *enable* and *prohibit* respectively. If the corresponding foot is on the ground, e_i is equal to one, and zero otherwise. If the corresponding foot is *not* in contact with the ground, p_i equals one, and zero otherwise.

IV. SIMULATIONS

In this section, simulation experiments of three different methods, all involving posture-support structures, will be discussed.

In order to guide the evolution towards human-like gait, and at the same time avoid the problems related to complex fitness functions (see Sect. I), the simplest fitness measure, i.e. the distance covered, has been used here, in combination with a mass-less posture-support structure, as shown in the right panels of Fig. 2. Given a supporting structure, the robot is forced to an upright position, and only individuals capable of producing repetitive leg motion will gain high fitness. When the support is used, such individuals will appear early in the evolution. However, a drawback with this method is that when the repetitive leg motion is discovered, individuals will start to exploit the support mechanism in many different ways. One common result is that individuals tend to take unnaturally large steps. While this gives high fitness when the support is used, it will certainly have a negative effect on the frontal plane balance once the support is removed. Thus, in an attempt to avoid this motion pattern, a choice was made not to evolve the internal parameters of the individual two-neuron CPGs, shown in Fig. 1, simply because the evolution would strive towards lower frequencies. Also, in order to prevent crawling behaviors, each individual run was aborted if the hips of a robot collided with the ground.

Information concerning the simulations is given in Table I. In the following subsections the simulation experiments will be described in more detail.

A. Method 1: Four-point support

The first experiments were made using a posture support with four contact points, as shown in the right panel of Fig. 2. The four-point support was attached to the robot in such a way that there was a predetermined distance d between the contact points of the support and the ground. Different values of d were examined, as shown in the 1^{st}, 2^{nd}, and 3^{rd} rows of Table I. Feedback was not used here, and the hip_2, hip_3 and ankle joints were locked. However, no successful gait patterns were obtained in this way. The individuals simply exploited the support too much, leading to unnatural gait patterns of different kinds, also briefly described in the table. For example, the $0.02, 2$ support configuration (3^{rd} row) led to an individual performing a running gait, which one might expect to be useful, but that individual over-exploited the support to such an extent that it could not maintain its balance at all without the support. In Fig. 4, some of the resulting motions are depicted.

In order to meet the intended goal, i.e. evolving a human-like gait for the robot, two modified strategies were also tried. In the first strategy the support was gradually removed, in the sense that d increased during evolution, as better fitness values were obtained. The assumption here was that this should eventually lead to an individual that was completely independent from the support. However, this approach did not improve the outcome, compared with the previous results.

In the second strategy, the support was not gradually removed during evolution, but instead individuals were punished for using it. The fitness measure was simply decreased by a factor, properly normalized, measuring the number of ground contacts with the support. However, this approach did not yield any improvements either.

In the case of four-point support no useful results were obtained; the individuals simply exploited the support too much, resulting in unnatural gait patterns. For example, when using the first modified strategy, gradually removing the support, a slow unstable gait pattern, resembling a drunkard's walk, emerged. When using the second modified strategy, with punishment for support usage, the result was an individual using a hop gait for locomotion.

B. Method 2: Two-point support in 2D, then 3D

Since no acceptable results were found with the four-point support, the support structure was changed to one having only a single contact point on each side of the robot, as seen in the rightmost panel of Fig. 2. Rather than evolving 3D balance at once, as in the previous case, the idea now was to divide the problem into two phases; first evolving gait in 2D, and second, to generalize it to a full 3D gait.

In this procedure, a CPG network capable of producing a stable upright gait in the sagittal plane should first be evolved using the two-point support, with the hip_2, hip_3 and ankle joints locked at this stage. Second, when a stable individual has been obtained, it should be cloned creating a new population consisting of copies of this individual. At this stage, the support should be removed and the GA should find a way to balance the robot in the frontal plane as well. Before the evolution starts, the hip_2 and ankle joints should be unlocked and the corresponding genes, including the genes encoding the waist joint parameters, should be randomly initiated for each individual in the population. Since the remaining genes (which are identical for all individuals) ensure sagittal plane balance they should not be changed in the second step.

1) Phase 1: Evolving balance in the sagittal plane: During this first phase the hip_2, hip_3 and ankle joints were locked. Balance in the sagittal plane was evolved using a

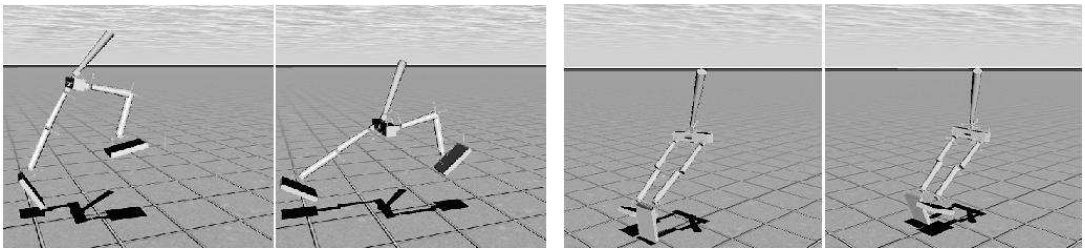

Fig. 4. The left panel shows the simulated robot taking unnaturally large steps. The right panel shows the robot exploiting the four-point support. Note that the supporting structure is not shown in the pictures.

two-point support, with contact points placed 2 m from the robot and 0.25 m above the ground. This configuration was chosen since it ensures low sideways leaning angle and at the same time allows the robot to bend its knees without the support touching the ground. Furthermore, if a robot's hips collided with the ground, the evaluation of that particular individual was terminated.

In order to balance in the sagittal plane the evolutionary procedure started misusing the torso as a third leg, achieving speeds up to 0.3 m/s. This problem was solved by simply removing the contact point in the torso which is used to detect the collision with the ground. Once the torso could not be used for support, evolution found the large step motion, as described earlier, ensuring balance in the sagittal plane with a speed of approximately 0.45 m/s, see the 4^{th} and 5^{th} rows of Table I.

In order to reduce the step length, hand-tuned feedback paths were introduced measuring torso, upper leg, and lower leg angles, as described in Sect. III. Adding feedback, a significant reduction in evolution time was observed. The same fitness value as before could now be reached approximately 5 times faster. However, the individuals were still taking unnaturally large steps.

Success in obtaining an upright human-like gait, with normal step size, was achieved using the following rule: during the evaluation of each individual, if the robot's hip fell below a certain value, the support was removed until the end of that run. If the step length is large and the support is removed in this way, the robot will most likely be unable to maintain the frontal plane balance. Thus, it will fall to the ground ending the run. Forced by this rule, evolution was able to find individuals moving at a speed of 1.13 m/s, see Table I, 6^{th} row. However, even after 400 generations, these individuals could not walk more than 10 to 15 meters before falling to the ground. In order to improve the performance, the GA was allowed to evolve the feedback paths as well. As a result, a stable individual, i.e. one that did not fall even after the end of the nominal evaluation time, was obtained within 50 generations, walking at a speed of 0.4 m/s, see Table I, 7^{th} row. To ensure stability, the whole foot sole was on the ground during almost the entire stance phase, resulting in a perfect condition for full 3D balance.

2) Phase 2: Evolving balance in full 3D: Once a satisfactory stable individual had been obtained using the support, evolution in the full 3D environment could begin. In the initial population at this stage, all individuals were mutated copies of the best individual from the previous step, as described above. The GA should now only consider the hip_2, ankle, and waist joints, as well as their feedback paths. The fitness measure was still taken as the distance covered in the initial forward direction, decreased by the sideways distance.

TABLE I

PARAMETERS AND RESULTS OF THE SIMULATION RUNS

In the column labeled *support*, the numbers i, j denote the initial placement of the contact points in a given run, where i is the height above the ground [m], and j is the horizontal distance from the hip [m]. Evaluation time is denoted by t [s], and the f column indicates whether or not feedback was used. F [m] is the obtained fitness, v is the average locomotion speed [m/s] of the robot during the evaluation period, and the last column gives a short description of the resulting gait. Note: † denotes phase 1, and ‡ denotes phase 2.

Support	t	f	F	v	Gait
4-point, 0.3, 2	7	No	3.85	0.55	hop gait
4-point, 0.1, 1	7	No	4.60	0.66	large steps
4-point, 0.02, 2	7	No	6.55	0.93	running
2-point, 0.25, 2 †	7	No	2.10	0.30	tripod gait
2-point, 0.25, 2 †	7	No	3.15	0.45	large steps
2-point, 0.25, 2 †	7	Yes	7.91	1.13	running
2-point, 0.25, 2 †	20	Yes	7.21	0.38	slow, stable
no support ‡	40	Yes	18.26	0.46	slow, stable
1 sec. 0.25, 2 †	40	Yes	19.54	0.56	slow, stable
1 sec. 0.25, 2 ‡	40	Yes	23.09	0.58	slow, stable
1 sec. 0.25, 2	40	Yes	35.56	0.90	fast walk

Within 150 generations, the best individual was able to maintain balanced walking for up to 60 seconds. After an additional 100 generations, the best individual was generally able to maintain balance indefinitely, see Table I, 8^{th} row. Since the robot was unaware of its direction of motion and because of the fact that the hip_3 joints were locked, the smallest perturbation would set it out of course, resulting in a lower fitness value.

Given the best individual from phase 2, it was possible to continue evolving the parameters for the hip_3 joint. The feedback for the hip_3 joint was defined as described in Sect. III. Evolution found a solution (not shown in the table) striving towards keeping the feet facing forward. A similar gait as before emerged.

C. Method 3: 2D one second support, then 3D

The reason for introducing the two-point posture-support structure in Method 2 was mainly that it is much harder

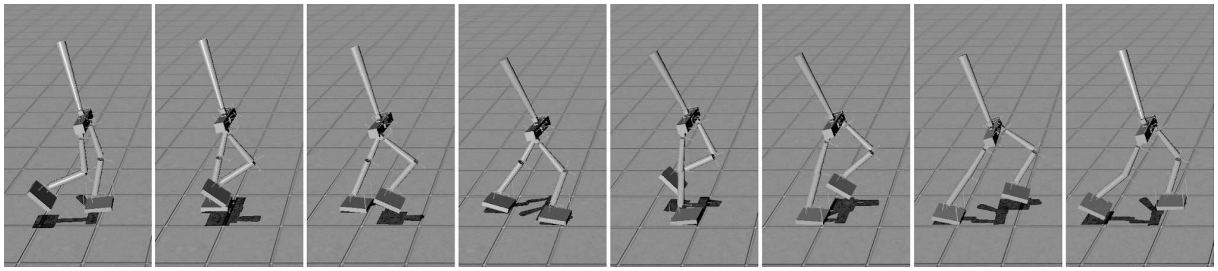

Fig. 5. The best evolved stable gait pattern in the full 3D environment. The details of the corresponding individual are shown on the 10^{th} row of Table I.

to maintain balance in the frontal plane than in the sagittal plane. By using the supporting structure, the problem was separated into two stages of evolution; first, generating a stable gait in 2D, and second, generalizing the 2D gait to three dimensions. The assumption here was that this way of splitting up the problem should make it easier for evolution to find a good solution to the overall problem of generating a robust 3D gait. However, since the hip_2 and ankle joints were locked during the 2D stage the balance in the frontal plane is then only ensured by the torso. As a consequence, evolution most often creates individuals that solve the problem in an unnatural way, i.e. individuals that exploited the supporting structure too much. Such individuals are usually not suitable for further evolution in 3D. Another drawback of Method 2 is that there is no obvious way of deciding at what point to interrupt the 2D evolution stage, and thus to enter the 3D phase: This simply has to be judged by the experimenter in an *ad hoc* manner. Hence, there is no guarantee that the individuals evolved in 2D will perform well in the 3D stage.

Therefore, a third method was investigated as well. The same two-point supporting structure as described in the 2D case in the previous method, was used. The difference here, as compared to the other method, is that under the first stage in 2D the individuals were evaluated in a procedure where the supporting structure was present only during the first second of the evaluation, and was then completely removed. This arrangement is motivated by the fact that it is during the start sequence, before entering the gait cycle, that the individuals are most vulnerable, in terms of frontal plane balance. Here, the hip_2, hip_3 and ankle joints were still locked. However, after some generations most individuals in the population should be able to walk without the supporting structure present, except during the initial second. This has been confirmed to work well in simulation experiments. The goal in this stage was to obtain a large portion of individuals able to walk in cautious manner, which is essential for individuals to be able to generalize to 3D. In the next stage, evaluations were performed in full 3D. That is, the evaluation procedure was performed in the same way as described above, but all joints except the hip_3 joints were unlocked. The 9^{th} row of Table I shows the results from evolution in 2D (joints locked and 1 second support), and the 10^{th} row of the table shows the results from phase 2 (evolution based on the best individual from the previous run, with hip_2 and ankle joints unlocked). A visualization of the gait for the best individual, corresponing to the 10^{th} row in the table, is shown in Fig. 5.

Moreover, the results from a run with hip_2 and ankle joints unlocked, performed in a single step, i.e. without the two-phase procedure described above, is shown in the last row. The gait obtained in this last run was fast, but appeared to be rather unstable.

V. DISCUSSION AND CONCLUSIONS

The outcome of the examinations and experiments described in this paper indeed fulfilled the intended goal, i.e. to generate robust bipedal gaits for a simulated robot by means of structural evolution of CPG networks: Two of the three methods introduced in this paper solved the problem of generating gaits for the simulated bipedal robot in 2D and 3D environments. However, Method 2 was affected by some drawbacks, compared to the third method. Firstly, since the support structure is present the whole time during phase 1 (evolution in 2D), evolution might very well find solutions that receive high fitness scores in 2D, but are less successful in generalizing to 3D. Examples of such solutions are individuals that walk with too long steps, giving high fitness in 2D because of their ability to cover large walking distances in short time. However, this kind of walking behavior seriously affects the frontal plane balance in 3D. Thus, evolution in 2D has to be aborted at an appropriate time, before this kind of behavior emerges which, in turn, requires that the evolutionary process is monitored more or less continuously in order to determine when it should be interrupted.

Secondly, while it is possible (at least in principle) to monitor the progress and stop the evolution when sufficient locomotion speed is achieved, it is not always the case that evolution chooses a path, i.e. relatively small steps with a high speed, that is suitable for further evolution in 3D. Often the large step motion behavior emerges before any stable, small-step gait pattern is obtained.

In the case of the third method the two problems described above are not present: Evolution is biased towards generating gaits capable of handling the 3D environment from the beginning. Thus, Method 3 seems to be the most promising candidate for future investigations.

The need for the support structure during the initial second, as described in the second method, indicates that the CPG network cannot fully handle the start-up of the walking cycle in an appropriate way. Thus, one should, in future work, consider a dedicated controller, either a CPG-based

controller or some other type of controller, for the start-up sequence of the walking cycle. It should then be tuned to enter the walking cycle and hand over to a CPG network in a more smooth way. Then, ultimately, it would be possible to skip totally the support structure.

In this paper only straight-line walking has been considered, i.e. no turning motions were involved. However, one could include such motions using the hip_3 joint in order to change deliberately the direction of walking, preferably by using vision for feedback.

Another topic for future work would be to investigate whether one could evolve the over-all feedback network, without having to pre-specify certain feedback paths, as is currently done. However, such an approach would probably increase the evaluation time considerably, since the likelihood of finding a set of feedback paths in an early generation that generates any gait at all would most probably be very small. On the other hand, evolving the feedback network could lead to better overall performance, compared to specifying the paths *ad hoc*.

REFERENCES

[1] R. A. Brooks, C. Breazeal, M. Marjanovic, and B. Scassellati, "The cog project: Building a humanoid robot," *Computation for Metaphors, Analogy, and Agents*, vol. 1562, pp. 52–87, 1999.

[2] H. Kozima and J. Zlatev, "An epigenetic approach to human-robot communication," in *Proc 9th Int Workshop on Robot and Human Interactive Communication (RO-MAN'00)*, Paris, France, 23–26 Mar. 2000, conference.

[3] T. Minato, M. Shimada, H. Ishiguro, and S. Itakura, "Development of an android robot for studying human-robot interaction." in *Proc 17th Int Conf on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2004*, 2004, pp. 424–434.

[4] A. Takanishi, M. Ishida, Y. Yamazaki, and I. Kato, "The realization of dynamic walking by the biped walking robot WL-10RD," in *Proc Int Conf on Advanced Robotics (ICAR'85)*, 1985, pp. 459–466.

[5] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, "The development of honda humanoid robot," in *Proc Int Conf on Robotics and Automation (ICRA'98)*. IEEE, 1998, pp. 1321–1326.

[6] T. Arakawa and T. Fukuda, "Natural motion generation of biped locomotion robot using hierarchical trajectory generation method consisting of GA, EP layers," in *Proc Int Conf on Robotics and Automation (ICRA'97)*. IEEE, 1997, pp. 211–216.

[7] A. L. Kun and W. T. Miller, "Control of variable speed gaits for a biped robot," *IEEE Robotics & Automation Magazine*, vol. 6, no. 3, pp. 19–29, Sep 1999.

[8] H. Wang, T. T. Lee, and W. A. Gruver, "A neuromorphic controller for a three-link biped robot," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 1, pp. 164–169, Jan/Feb 1992.

[9] G. Taga, Y. Yamaguchi, and H. Shimizu, "Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment," *Biological Cybernetics*, vol. 65, pp. 147–159, 1991.

[10] J. Pettersson, H. Sandholt, and M. Wahde, "A flexible evolutionary method for the generation and implementation of behaviors for humanoid robots," in *Proc 2nd Int Conf on Humanoid Robots (Humanoids'01)*, IEEE-RAS, Waseda University. Tokyo, Japan: Humanoid Robotics Institute, 22-24 Nov. 2001, pp. 279–286.

[11] J. Shan, C. Junshi, and C. Jiapin, "Design of central pattern generator for humanoid robot walking based on multi-objective ga," in *Proc Int Conf on Intelligent Robots and Systems (IROS 2000).*, vol. 3. Takamatsu, Japan: IEEE-RSJ, 2000, conference, pp. 1930–1935.

[12] M. Y. Cheng and C. S. Lin, "Genetic algorithm for control design of biped locomotion," *Journ. of Robotic Systems*, vol. 14, no. 5, pp. 365–373, 1997.

[13] K. Wolff and P. Nordin, "Learning biped locomotion from first principles on a simulated humanoid robot using linear genetic programming," in *Proc Genetic and Evolutionary Computation Conf (GECCO'03)*, ser. LNCS, E. Cantú-Paz, Ed., vol. 2723, AAAI. Chicago: Springer Verlag, 12-16 July 2003, pp. 495–506.

[14] J. Ziegler, J. Barnholt, J. Busch, and W. Banzhaf, "Automatic evolution of control programs for a small humanoid walking robot," in *Proc 5th Int Conf on Climbing and Walking Robots (CLAWAR'02)*, P. Bidaud, Ed. Professional Engineering Publishing, 2002, pp. 109–116.

[15] S. Grillner, "Neural networks for vertebrate locomotion." *Scientific American*, vol. 274, pp. 64–69, 1996.

[16] S. Grillner, T. Deliagina, Ö. Ekeberg, A. El Manira, R. Hill, A. Lansner, G. Orlovsky, and P. Wallen, "Neural networks that coordinate locomotion and body orientation in lamprey," *Trends in Neurosciences*, vol. 18, no. 6, pp. 270–279, 1995.

[17] Ö. Ekeberg, "A combined neuronal and mechanical model of fish swimming," *Biological Cybernetics*, vol. 69, no. 5-6, pp. 363–374, oct 1993.

[18] P. Wallén, Ö. Ekeberg, A. Lansner, L. Brodin, H. Tråvén, and S. Grillner, "A computer-based model for realistic simulations of neural networks. II: The segmental network generating locomotor rhythmicity in the lamprey," *J. Neurophysiol.*, vol. 68, pp. 1939–1950, 1992.

[19] T. G. Brown, "The factors in rhythmic activity of the nervous system." *Proc R Soc London Ser*, vol. 85, pp. 278–289, 1912.

[20] S. Grillner and P. Zangger, "The effect of dorsal root transection on the efferent motor pattern in the cat's hindlimb during locomotion." *Acta Physiol Scand*, vol. 120, pp. 393–405, 1984.

[21] J. Duysens and H. W. A. A. V. de Crommert, "Neural control of locomotion; part 1: The central pattern generator from cats to humans." *Gait and Posture*, vol. 7, no. 2, pp. 131–141, 1998.

[22] G. Taga, "Nonlinear dynamics of the human motor control - real-time and anticipatory adaptation of locomotion and development of movements," in *Proc 1st Int Symp on Adaptive Motion of Animals and Machines (AMAM'00)*, 8–12 Aug. 2000.

[23] T. Reil and P. Husbands, "Evolution of central pattern generators for bipedal walking in a real-time physics environment." *IEEE Transactions in Evolutionary Computation*, vol. 6, no. 2, pp. 159–168, 2002.

[24] H. Kimura, S. Akiyama, and K. Sakurama, "Realization of dynamic walking and running of the quadruped using neural oscillator," *Autonomous Robots*, vol. 7, no. 3, pp. 247–258, 1999.

[25] K. Tsuchiya, S. Aoi, and K. Tsujita, "Locomotion control of a biped locomotion robot using nonlinear oscillators," in *Proc Int Conf on Intelligent Robots and Systems (IROS'03)*. IEEE/RSJ, 2003, pp. 1745–1750.

[26] M. Lewis, F. Tenore, and R. Etienne-Cummings, "CPG design using inhibitory networks," in *Proc Int Conf on Robotics and Automation (ICRA'05)*, IEEE-RAS. Barcelona, Spain: Wiley, 18-22 Apr. 2005.

[27] M. Ogino, Y. Katoh, M. Aono, M. Asada, and K. Hosoda, "Reinforcement learning of humanoid rhythmic walking parameters based on visual information," *Advanced Robotics*, vol. 18, no. 7, pp. 677–697, 2004.

[28] K. Matsuoka, "Mechanisms of frequency and pattern control in the neural rhythm generators." *Biol Cybern*, vol. 56, no. 5–6, pp. 345–353, 1987.

[29] C. Paul and J. Bongard, "The road less travelled: Morphology in the optimization of biped robot locomotion," in *Proc Int Conf on Intelligent Robots and Systems (IROS'01)*, vol. 1. Maui, HI, USA: IEEE/RSJ, 2001, pp. 226–232.

[30] G. M. Shepherd, *Neurobiology*, 3rd ed. Oxford University Press, 1994, ch. 20, pp. 435–451.

[31] T. G. Brown, "The intrinsic factors in the act of progression in the mammal." *Proc R Soc London Ser*, vol. 84, pp. 308–319, 1911.

[32] E. Ott, *Chaos in Dynamical Systems*. New York: Cambridge University Press, 1993.

[33] J. Pettersson, "EvoDyn: A simulation library for behavior-based robotics," Department of Machine and vehicle systems, Chalmers University of Technology, Göteborg, Technical Report, September 2003.

[34] R. Featherstone, *Robot Dynamics Algorithms*. Kluwer Academic Publishers, 1987.

Paper VII

Evolutionary optimization of a bipedal gait in a physical robot

Submitted to

IEEE Transactions on Robotics.

Evolutionary optimization of a bipedal gait in a physical robot

Krister Wolff, David Sandberg, and Mattias Wahde

Department of Applied Mechanics, Chalmers University of Technology
412 96 Göteborg, Sweden

***Abstract*— Evolutionary optimization of a gait for a bipedal robot has been studied, combining structural and parametric modifications of the system responsible for generating the gait. The experiment was conducted using a small 17 DOF humanoid robot, whose actuators consist of 17 servo motors with integrated closed-loop control. In the approach presented here, individuals representing a gait consisted of a sequence of set angles (referred to as states) for the servo motors, as well as ramping times for the transition between states. A hand-coded gait was used as starting point for the optimization procedure: A population of 30 individuals was formed, using the hand-coded gait as a seed. An evolutionary procedure was executed for 30 generations, evaluating individuals on the physical robot. New individuals were generated using mutation only. There were two different mutation operators, namely (1) parametric mutations modifying the actual value of a given gene, and (2) structural mutations inserting a new state between two consecutive states in an individual. The best evolved individual showed an improvement in walking speed of approximately 65%.**

I. Introduction

This paper is concerned with evolutionary optimization of a gait for a bipedal robot. In the coming era of autonomous robots, it is generally believed that humanoid robots, i.e. walking robots with an approximately human-like shape, will play an important role, since such robots can be more naturally adapted to environments primarily designed for people and are also likely to be easier to interact with than wheeled robots with non-humanoid characteristics [1].

However, there are many difficult issues associated with humanoid robots as well, one of the most important being their complex dynamics: In general, humanoid robots require a complex coordination of limbs during walking as well as active balancing even during standstill. Wheeled robots, by contrast, are normally statically stable, and the motor control of such robots can mostly be devoted to the navigation of the robot rather than simply keeping the robot upright.

During the last two decades, a large amount of work has been carried out regarding the generation of stable bipedal gaits. A variety of methods have been used, one of the most popular being the ZMP method [2], [3], in which reference trajectories are generated so as to keep the zero-moment point (ZMP) within the convex hull of the support area generated by the feet of the robot. Evolutionary methods have also been applied to the problem of gait generation [4], [5], [6], [7], with the aim of generating e.g. gaits that are more responsive to sudden perturbations or other unexpected events that may cause significant trouble for gaits based on reference trajectories. However, evolutionary methods generally require that a large number of candidate soluations (individuals) should be tested, a procedure that is very time-consuming if it is carried out in physical robots. Thus, simulations have commonly been used in connection with evolutionary gait generation [4], [5], [6], [7], [8], [9], [10].

However, simulations come with a significant drawback, namely the difficulty of accurately modeling the humanoid robot, which, in turn, leads to severe problems when transferring generated gaits to a physical robot. No matter how carefully a humanoid robot is modelled, there will always be discrepancies between the simulated robot and the physical robot. Since humanoid robots generally have many degrees of freedom (DOF) there are many things that can go wrong when a gait is transferred from a simulated robot to a physical robot. In addition, some phenomena are also very hard to model accurately, such a resonances caused by the interaction between the actuactors and the physical structure of the robot. Thus, evolution directly in hardware does carry several advantages. Incidentally, the difficulty in modeling humanoid robots may be seen as another motivation *in favor* of the use of evolutionary methods (if applied directly to physical robots), since such methods normally do not require a dynamic, or even kinematic, model of the physical robot.

It is thus clear that, if only the evolutionary procedure can be made sufficiently rapid to allow for the generation of a satisfactory bipedal gait after, say, a few hundred (rather than thousands) of evaluated individuals, evolution directly in hardware would be an attractive alternative. A method for speeding up the evolutionary process is, of course, to start from a rough, hand-coded gait that can at least make the robot move, and then proceed to optimize this gait using an evolutionary algorithm (EA). Indeed, such a method was applied successfully by Wolff and Nordin [11]. However, their approach suffered from a significant drawback: The representation of individuals was such that the EA could only modify the *parameters* of the gait, not its *structure*. If no structural modifications are allowed, as in Wolff and Nordin [11], the structure of the initial hand-coded gait (i.e. the starting point for the EA) must be completely accurate in order for the EA to be able to find an optimal gait through parametric optimization. In this paper, the analysis of Wolff and Nordin [11] will be extended to the case of combined structural and parametric evolution.

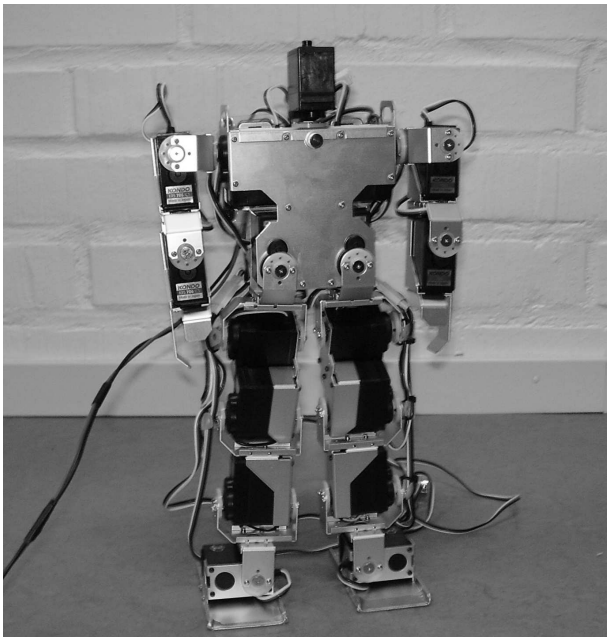

Fig. 1. The Kondo robot.

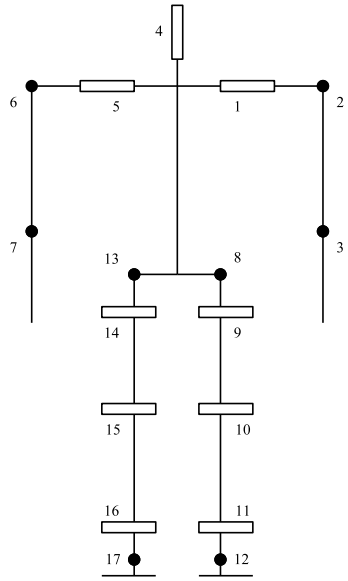

Fig. 2. A schematic view of the Kondo robot used in the experiment. The robot has 8 joints, shown as filled disks in the figure, whose axes of rotation are orthogonal to the frontal plane. These joints rotate in the clockwise direction for increasing set values. With the exception of the joint actuating the head, the remaining joints generate motion in the sagittal plane.

II. Experiment description

A. The robot

The experiment was executed using a small, 17-DOF humanoid robot manufactured by Kondo Kagaku Co. Ltd. [12], and distributed by iXs research corporation [13]. The robot is shown in Fig. 1. Fig. 2 shows a schematic view of the robot, with the definition of the 17 joint angles. The height of the robot is 0.33 m. The actuators of the bipedal robot consist of 17 KRS 786 ICS servo motors with integrated closed-loop control, manufactured by Kondo [12]. Each servo motor has a nominal output torque of 0.85 Nm, and a maximum rotation speed of 6.16 rad/s (0.17 s/ 60 degrees). The controller consists of two RCB-1 boards, also manufactured by Kondo. The RCB-1 board is based on the PIC 16F873A chip from Microchip Technology Inc. [14], and each controller board can control up to 12 servo motors.

B. Experimental setup

The arena in which the robot was evaluated is shown in Fig. 3. The evaluation of an individual consisted of placing the robot at the starting point (the black line shown in the left part of the picture), uploading the gait onto the robot and then running it, starting at time T_0. A photocell was used for measuring passage of the finish line (the black line shown in the right part of the picture) at time T_1. When the finish line was reached, the time $T = T_1 - T_0$ was recorded. In case of failure, i.e. if the robot fell over or strayed too much from the intended path (such that it would not pass the finish line between the measurement points shown in the picture), the evaluation was interrupted. The distance L between the starting line and the finish line was 0.53 m, i.e. around 1.6 times the height of the robot. This value was chosen due to hardware limitations that restrict the number of executed states

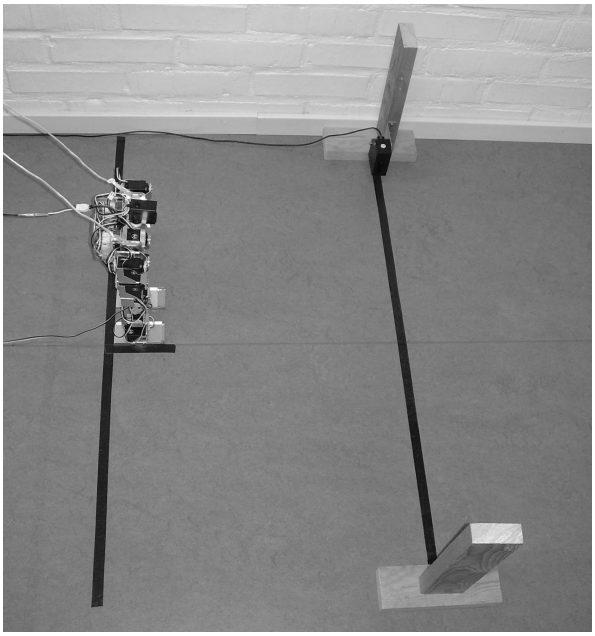

Fig. 3. The arena used in the experiment.

to 100 in any given evaluation, i.e. a maximum of 16 cycles for a gait with 6 states (see below).

C. Gait architecture

In the architecture used here, a gait consists of a sequence of set angles (henceforth referred to as *states*) for the servo motors, as well as ramping times for the transition between states. An example of a gait, namely the standard, hand-coded gait, is shown in Table I. In this table, each row represents a state. The final column of each row shows the ramping parameter, i.e. the speed of the transition from the current state

State	φ_1	φ_2	φ_3	φ_4	φ_5	φ_6	φ_7	φ_8	φ_9	φ_{10}	φ_{11}	φ_{12}	φ_{13}	φ_{14}	φ_{15}	φ_{16}	φ_{17}	R
1	9	11	90	120	150	169	90	*97*	85	49	127	*80*	*103*	69	114	46	*78*	6
2	9	11	90	120	175	169	90	*97*	85	49	127	*79*	*103*	107	172	23	*78*	3
3	30	11	90	90	175	169	90	*97*	85	49	127	*79*	*102*	116	128	73	*76*	3
4	30	11	90	90	175	169	90	*73*	119	82	124	*104*	*79*	88	111	67	*102*	6
5	9	11	90	60	175	169	90	*73*	72	15	150	*104*	*79*	88	111	64	*102*	3
6	9	11	90	90	150	169	90	*73*	65	66	93	*107*	*75*	102	124	64	*102*	3

TABLE I

THE STANDARD GAIT, CONSISTING OF SIX STATES THAT ARE EXECUTED CYCLICALLY. THE PARAMETERS SHOWN IN ITALICS ARE KEPT CONSTANT DURING THE EVOLUTIONARY OPTIMIZATION PROCEDURE, I.E. THEY ARE UNAFFECTED BY PARAMETRIC MUTATIONS, SEE ALSO SUBSECT. II-D.1.

to the next one. The ramping parameter takes values in the range $[0, 7]$, where 0 indicates the fastest possible transition, and 7 the slowest possible transition.

1) The standard gait: The gait used as a starting point for the optimization procedure described below, is given in Table I. As can be seen from the table, the gait consists of six states. When executing the gait, the states are used cyclically, starting from state 1.[1] Since the standard gait (hereafter: SG) is specified using only six states, it is not very smooth. However, it allows the robot to cover the distance L in around 36.4 seconds. A sequence of images, showing the robot in each of the six states, is given in Fig. 4. Note that the SG is statically stable, so that the robot can stay balanced indefinitely in any of the six available states.

D. The evolutionary algorithm

The EA employed in this study was a fairly standard one, except for (1) the absence of a crossover operator and (2) the presence of structural mutations. In this subsection, the encoding scheme, the genetic operators, and the fitness measure will be described.

1) Encoding scheme: The genome of each individual was represented as 17 chromosomes. Each of the first 16 chromosomes encodes the sequence of set angles for a given servo motor j, i.e. the transitions $\varphi_j^{[i]} \rightarrow \varphi_j^{[i+1]},\ i = 1, \ldots, N-1$, where N is the number of states. Due to hardware limitations, the maximum value of N was equal to 10. The final chromosome encodes the speed of transition between the different states, using an integer ramping parameter R as described in Subsect. II-C.

2) Genetic operators: In the formation of new individuals, which took place on a host computer, only mutations were used. Since the evolutionary algorithm started from a functioning gait, it was unlikely that a search involving large mutation steps would generate improved gaits. Thus, *creep mutations* were used instead. In the case of the chromosomes encoding angles (chromosomes 1-16), the mutation operator changed a given gene (set angle) $\varphi_j^{[i]}$ to a new value according to

$$\varphi_j^{[i]} \leftarrow \varphi_j^{[i]} + \Delta, \tag{1}$$

where Δ is a random number in the range $[-\Delta_{\max}, \Delta_{\max}]$. Thus, with small values of $\Delta_{\max}$ the mutation generates a new value near the old value. A $\Delta_{\max}$ value of 5 was used throughout the experiment. In case the new set angle exceeded the allowed limit of variation ($[0, 225]$) for the servo in question, the set angle was adjusted to the limiting value.

Similarly, for chromosome 17, the mutation operator changed a given ramping parameter R according to

$$R \leftarrow R \pm 1, \tag{2}$$

with equal probability for either sign, except for the limiting cases $R = 0$, where only an increase was allowed, and $R = 7$, where only a decrease was allowed.

The parameteric mutations of servo angles and ramping parameters were carried out on gene-by-gene basis, i.e. for each gene a random number $r \in [0, 1]$ was generated, and if r was smaller than the parametric mutation probability p_{p}, the gene in question (servo angle or ramping parameter) was modified.

However, for four servos, namely 8, 12, 13 and 17, the variation in the servo angles described by the SG was kept, i.e. mutations were not allowed for these particular servos. This restriction was introduced since it was found that any mutation of these servo values led to situations in which a foot would touch the ground at an angle (rotation in the frontal plane) , causing the robot to fall. Thus, in the end the EA was allowed to specify the angular variation of 12 servos and one ramp value, for each state, so that the total number of parameters was equal to $13N$.

Structual mutations were allowed as well. Such mutations may, in principle, either decrease or increase the length of the chromosomes. However, during tests preceding the experiment, it was found that removal of states generally led to disastrous results (the robot would fall), except in those rare cases where the removed state was one that had just been added. Thus, removal of states was not used, and the chromosomes were thus only allowed to increase in length.

Structural mutations were used in such a way that, with probability p_{s} a new state was inserted between states i and $i+1$, $i = 1, \ldots, N-1$. In order to avoid destroying the gait, the set angles of the new state were given as

$$\varphi_j^{\mathrm{new}} = \frac{\varphi_j^{[i]} + \varphi_j^{[i+1]}}{2}. \tag{3}$$

Similarly, the gene representing the ramping parameter (inserted in the 17^{th} chromosome) was generated in the same way as the set angles.

[1] However, since the robot starts from the stance shown in Fig. 1, it must first take one step to reach state 1 in the gait. Thus, before the cyclical gait is executed, both the SG and all other evaluated gaits begin by a hand-coded step of the kind just mentioned.

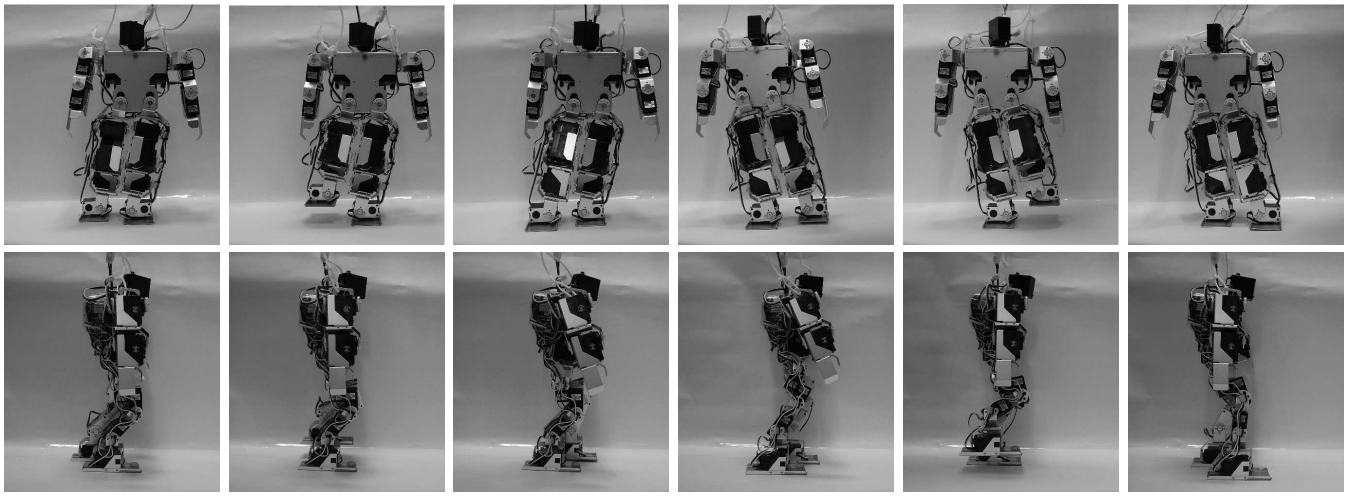

Fig. 4. The six states of the standard gait. The upper row shows the robot seen from the front, and the lower row shows a side view.

3) Fitness measure: In this investigation, the speed of walking combined with strong punishment for instability, has been taken as the measure of quality of a gait. More specifically, the fitness of an individual was taken as

$$f = \frac{T_{\rm SG}}{T}, \tag{4}$$

where T and $T_{\rm SG}$ are the times taken to traverse the distance L for the individual in question and for an individual executing the SG, respectively. This fitness measure was used for individuals that actually reached the finish line, thus obtaining a value for the time taken. Individuals that fell over were given fitness 0, whereas individuals that walked in a strongly curved path (such that they would not break the light beam between the two measurement points on the finish line), were somewhat arbitrarily given fitness 0.1. Such individuals were given a non-zero fitness since an inability to walk in a straight line was considered less severe than an outright instability of the gait in question. However, it should be noted that the fitness values (defined in Eq. (4)) of individuals that reached the finish line was much higher than 0.1. The SG was evaluated 10 times, leading to a value of $T_{\rm SG}$ of 36.4 ± 0.3 s (95% confidence interval). During this time interval the robot executed approximately 16 steps.

III. Results

The experiment was carried out using a population size of 30 individuals. In the formation of the initial population, the genomes of all individuals were first set manually so as to encode the SG. Next, the individuals were mutated, using the procedure that normally takes place *after* the evaluation of a generation, and the resulting set of individuals formed the initial population.

The evolutionary procedure was carried out for a total of 30 generations. Thus, in all, 900 individuals were evaluated. Elitism was employed, i.e. one copy of the best individual in generation g was transferred, unchanged, to generation $g+1$. Each individual was evaluated once, and its fitness value was then stored. When all individuals in a given generation had been evaluated, the next generation was formed using the genetic operators described above. The parametric mutation rate was set to $2/K$, where $K = 13N$ is the number of parameters. The structural mutation rate was set to $1/3(N-1)$.

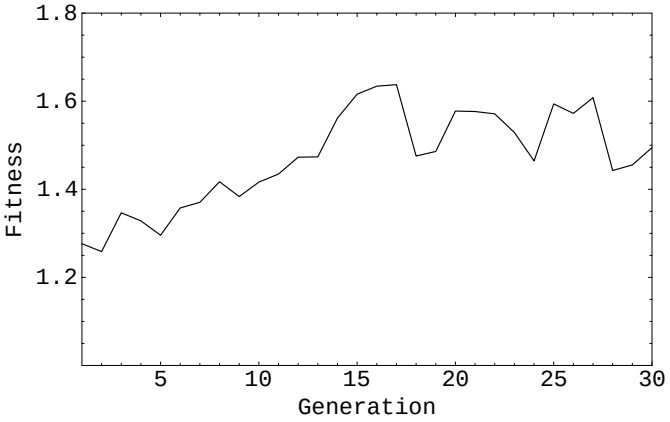

Fig. 5. Fitness of the best individual as a function of generation. Note that the maximum fitness value was obtained in generation 17.

The results are shown in Figs. 5 and 6. Fig. 5 shows the maximum fitness as a function of generation. Note that the (average) fitness of the SG is equal to 1. From the figure, it is clear that the fitness values rise quite rapidly, from around 1.276 to 1.638 over the first 17 generations. Then there is a sudden drop, followed by oscillatory best fitness values roughly in the range 1.5 ± 0.1. In fact, the best individual appeared already in generation 15 where it obtained fitness 1.616. In generation 16 this individual obtained a fitness of 1.634, again making it the best individual in the population. Finally, in generation 17 it reached a fitness value of 1.638, the global maximum fitness value obtained during the EA run. However, in generation 18, the individual fell, and was therefore eliminated from the population. Note that, despite the use of elitism, the maximum fitness value may thus drop from one generation to the next, since the performance of any given individual may vary between evalutions.

The average fitness values are shown in Fig. 6. As is evident

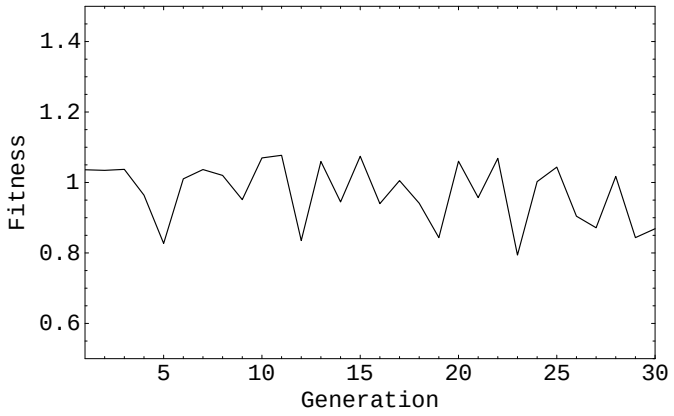

Fig. 6. Average fitness as a function of generation.

from the figure, there is no trend in the average fitness value.

Even though each individual was only evaluated once during the EA run, the genetic material of all individuals was stored so that any individual could be re-evaluated after the completion of the run. Such a re-evaluation was indeed carried out for the best individuals in each generation, and it was found that fitness values (at least for these individuals) typically remained quite stable during re-evaluations: Typical variations were around 2%. However, it should also be noted that, occasionally, even a rather good individual would fall. Of the 30 individuals that were re-evaluated, five fell during at least one re-evaluation.

In general, there was an increasing trend in the number of states. This is not surprising *per se*, since, as described above, states could not be removed. The number of states increased from an average of 6.5 in the first 10 generation to an average of 8.9 in the last 10 generations.

The best gait, generated in generation 15 (but obtaining maximum fitness in generation 17, as mentioned above), is described in Table II. As is indicated in the table, this gait consisted of eight states, i.e. two states were added in addition to the six states present in the SG. The added states were inserted at positions 4 and 5. When the EA run had been completed, the best individual was re-evaluated 10 times, leading to a fitness value of 1.643 ± 0.112 (95% confidence interval). Note that, in order to carry out 10 successful re-evaluations for this individual, 12 attempts were needed. In other words, even the best individual would sometimes fail.

The sideways deviation over the distance L was also measured and was found to be, on average, 0.09 m (always in the same direction), compared to 0.08 m for the SG.

IV. Discussion and conclusion

The most important conclusion that can be drawn from the experiment described above is that it is indeed possible to obtain significant improvements of a functioning but non-optimized bipedal gait using an EA running directly on a physical robot. It is interesting to note that, already after 17 generations, i.e. after the evaluation of around 500 individuals, an improvement in walking speed of around 65% could be obtained.

From Table II, it is clear that the values of the ramping parameter have been somewhat reduced (on average), compared to the values used in the SG, so that the transition between states becomes faster than for the SG. A reasonable hypothesis, in the light of this result, may be that the SG could perhaps be improved simply by reducing the values of the ramping parameter. However, this approach was tested and was shown to lead to a highly unstable gait. Thus, in order for the gait to be improved, an increase in transition speed must be accompanied by modifications of the set angles and, possibly, the number of states, and is thus very difficult to obtain by manual tuning.

In the EA, no crossover operator was used since, first of all, crossover between individuals with different number of states is unlikely to produce improved results. This is so, since the relative timing (in the step cycle) will differ between individuals with different number of states. Thus, a crossover operator might, for example, exchange state 2 between two individuals with, say 6 and 10 states, respectively. However, state 2 in an individual with 6 states corresponds to a completely different part of the gait cycle than state 2 in an individual with 10 states. One could still imagine using crossover between individuals with the *same* number of states, thus creating essentially the equivalent of species. However, not even such a crossover operator was used since, as described in Sect. III, the EA started from a population of rather similar individuals. The use of a crossover operator would quite quickly lead to a loss of diversity due to inbreeding. Finally, the steady increase in the fitness values (at least in the first half of the run), even with only mutation present as a variation operator, made the use of a crossover operator unnecessary.

The dip in fitness values after generation 17 was caused by an unfortunate failure (in generation 18) of the best individual in generation 17: The robot fell, and the corresponding individual was thus eliminated. If multiple evaluations had been used to form a reliable average performance of each individual, this particular individual may have survived and even higher fitness values might then have been obtained. However, the choice of using a single evaluation was motivated by the fact that, even with this limitation, the evaluation of 900 individuals took quite a long time (around 3-4 full working days). It is doubtful whether the use of, say, 3 re-evaluations per individual would have led to a *better* final result, since then only 300 individuals could have been evaluated unless, of course, the experiment was extended in time. Nevertheless, it is important to note that a strong improvement of the original gait (the SG) was obtained even though a single evaluation was carried out for each individual.

An interesting possibility for future work would be to base the fitness value on the heritage of a given individual, i.e. to form its final fitness value as a sum of the current fitness value and that of its ancestors, perhaps using a discounting factor for individuals from earlier generations. In this way, the elimination of a good individual based on a single failure could be avoided, *without* having to carry out time-consuming re-evaluations of each individual.

At a first glance, the absence of a trend in the average fitness values may seem surprising, particularly since the best

State	φ_1	φ_2	φ_3	φ_4	φ_5	φ_6	φ_7	φ_8	φ_9	φ_{10}	φ_{11}	φ_{12}	φ_{13}	φ_{14}	φ_{15}	φ_{16}	φ_{17}	R
1	10	11	90	120	150	169	90	*97*	85	49	127	*80*	*103*	69	114	46	*78*	5
2	9	11	90	120	175	169	90	*97*	85	49	127	*79*	*103*	113	172	23	*78*	0
3	31	11	90	90	175	169	90	*97*	82	49	127	*79*	*102*	121	128	73	*76*	0
4	30	11	90	89	175	169	90	*85*	99	65	125	*91*	*90*	104	122	71	*89*	3
5	30	11	90	90	175	169	90	*79*	108	73	124	*97*	*84*	96	119	68	*95*	4
6	30	11	90	90	175	169	90	*73*	117	82	124	*104*	*79*	88	116	67	*102*	3
7	9	12	90	60	178	172	87	*73*	72	15	154	*104*	*79*	88	111	64	*102*	3
8	9	11	95	90	147	174	90	*73*	65	71	94	*107*	*75*	102	124	64	*102*	0

TABLE II

THE BEST GAIT, CONSISTING OF EIGHT STATES. NOTE THAT THE PARAMETERS SHOWN IN ITALICS ARE UNAFFECTED BY PARAMETRIC MUTATIONS.

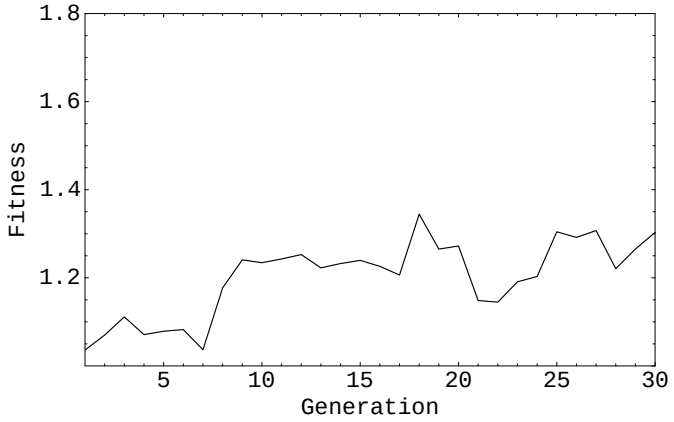

Fig. 7. Average fitness as a function of generation, taken over those individuals that successfully (without falling) traversed the whole distance L, without excessive sideways deviation. See the main text for a further discussion of this figure.

fitness values *do* improve. However, one must keep in mind that the initial population consists of individuals executing a slightly modified version of the rather stable standard gait (with an average fitness of 1), whereas later generations will contain individuals whose gaits are more significantly modified compared to the SG. Some of these individuals will have gaits that outperform the SG whereas other individuals will fail altogether, thus obtaining a fitness value of 0 or 0.1, depending on the mode of failure, as described above. Thus, the numerical average value may be misleading, since it depends on the somewhat arbitrary fitness values (0 and 0.1) assigned to failed individuals. A better measure of the actual average performance can be obtained by measuring the average fitness value of those individuals that do not fail. A plot of these values is shown in Fig. 7. As can be seen from this figure, there is indeed an increase in the average, from 1.036 in the first generation to 1.303 in the last, with a maximum of 1.345 in generation 18. Admittedly, this is an overestimate of the average performance, since the number of failed individuals also rises, from an average of 3.0 in the first 10 generations to 7.2 in the last 10 generations, but it does indicate that the EA is able to improve the performance of those individuals that do not fail completely.

As for the walking speed, the best individual walked a distance of 0.53m in approximately 22.2s, i.e. with a speed of only 0.024 m/s. Scaling from the height of the robot (0.33 m) to the height of a typical human (1.75 m), this would correspond to a speed of roughly 0.12 m/s which, of course, is quite slow. However, what matters is the improvement compared to the SG, with an average speed of only 0.015 m/s. The low *absolute* speed can be attributed to hardware limitations. In fact, even though the robot has a humanoid shape, the placements of joints differs quite significantly from those of a human. Furthermore, the body of the robot is, of course, much more rigid.

As for the robustness of the generated gaits, it was found (as mentioned in Sect. III) that even the best individuals would sometimes fall. Thus, the evolved gaits displayed somewhat lower robustness than the SG. An interesting topic for future work would be to optimize the SG while placing even higher emphasis on the robustness of the generated gaits. In order to be successful, such a study would probably require the use of several re-evaluations of each individual for the formation of the fitness value.

The initial motivation for carrying out the gait generation in physical robots rather than in simulations can be strengthened further by noting that the sideways deviation of the best individual always occurred in the same direction, as a result of the peculiarities of the particular components (and their placement) used in that robot. Had simulations been used, it is unlikely that such an effect would have been correctly modelled, at least without extensive and time-consuming detailed system identification. Thus, even though evolution in physical robots is a complex procedure, in view of the direct applicability of the results obtained, it appears to be well motivated in the case of gait optimization in bipedal robots.

REFERENCES

[1] R. Brooks, "Prospects for human level intelligence for humanoid robots," in *Proceedings of the First International Symposium on Humanoid Robots, HURO-96*, october 1996.

[2] M. Vukobratovic and D. Juricic, "Contributions to the synthesis of biped gait," in *IEEE Transactions on Biomedical Engineering*, vol. BME-16, 1969, pp. 1–6.

[3] A. Takanishi, M. Ishida, Y. Yamazaki, and I. Kato, "The realization of dynamic walking by the biped walking robot WL-10RD," in *Proceedings of the International Conference on Advanced Robotics (ICAR'85)*, 1985, pp. 459–466.

[4] T. Arakawa and T. Fukuda, "Natural motion trajectory generation of biped locomotion robot using genetic algorithm through energy optimization," *Proc. of the 1996 IEEE International Conference on Systems, Man and Cybernetics*, pp. 1495–1500, 1996.

[5] C. Paul and J. Bongard, "The road less travelled: Morphology in the optimization of biped robot locomotion," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2001)*, Hawaii, USA, 2001, pp. 226–232.

[6] J. Pettersson, H. Sandholt, and M. Wahde, "A flexible evolutionary method for the generation and implementation of behaviors for humanoid robots," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robotics*, November 2001, pp. 279–286.
[7] K. Wolff, J. Pettersson, A. Heralic, and M. Wahde, "Structural evolution of central pattern generators for bipedal walking in 3d simulation," in *Proceedings of the 2006 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2006)*, Taipei, Taiwan, October 2006, pp. 227–234.
[8] M. Wahde and J. Pettersson, "A Brief Review of Bipedal Robotics Research," in *Proceedings of the* 8^{th} *UK Mechatronics Forum International Conference (Mechatronics 2002)*, June 2002, pp. 480–488.
[9] M. Y. Cheng and C. S. Lin, "Genetic algorithm for control design of biped locomotion," *Journal of Robotic Systems*, vol. 14, no. 5, pp. 365–373, 1997.
[10] Y. Fujimoto and A. Kawamura, "Simulation of an autonomous biped walking robot includinint. j. on artifi cial intelligence toolsg environmental force interaction," *IEEE Robotics and Automation Magazine*, vol. 5, no. 2, pp. 33–42, June 1998.
[11] K. Wolff and P. Nordin, "Evolution of effi cient gait with humanoids using visual feedback," in *Proceedings of the IEEE-RAS International Conference on Humanoid Robotics*, November 2001, pp. 99–106.
[12] Kondo Kagaku Co. Ltd., http://www.kondo-robot.com.
[13] iXs, http://www.ixs.co.jp/eco-robot.html.
[14] Microchip Technology Inc., http://www.microchip.com.