Institute of Parallel and Distributed Systems
University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3290

Online Evolution and Adaptation of Central Pattern Generators for Multi-robot Organisms

Patrick Alschbach

Course of Study:	Computer Science
Examiner:	Prof. Dr. rer. nat. habil. Paul Levi
Supervisor:	Dipl.-Inf. Florian Schlachter
Commenced:	07.12.2011
Completed:	07.06.2012
CR-Classification:	I.2.6, I.2.9, I.2.11

Acknowledgements

First and for most, I would like to thank the examiner Prof. Dr. rer. nat. habil. Paul Levi at the University of Stuttgart for giving me the chance to be involved in this project.

I would like to thank my supervisor Dipl. Inf. Florian Schlachter. I appreciate the freedom that Florian gave me in choosing the research direction and methods. Along the way I have benefited a lot from the discussion with him.

I am also grateful to Tobias Haberl, who has assumed the proofreading of this thesis. I would like also to thank Sergej Popesku and all the other colleagues in the lab for all the suggestions and kindness help.

Abstract

This thesis deals with possibility to provide a robot organism, consisting of an amount of single smaller robots, with the ability of locomotion. It is integrated into the *SYMBRION* project which is funded by the European Union. The used robots and the simulation environment are a product from this major project for swarm robotics.

The presented locomotion approach uses artificial neural networks which are composed of third generation neurons called "Spiking Neurons". For evaluating the generated motion patterns the artificial neural networks are evolutionary adapted which was realized by using "Evolutionary Acquisition of Neural Topologies". In this thesis the evolutionary engine "EvoRoF", launched by Florian Schlachter of the University of Stuttgart, was used. The findings of this scientific work were included directly in the adjustment process of this evolutionary engine.

Specially the focus of this thesis is on distributed online evolution. Meaning that each robot of the whole organism has its own population of individuals and thus its own set of artificial neural networks.

In the course of the evolutionary process the artificial neural networks start from scratch on directly on the robotic system. There are no networks which were precalculated on a desktop computer.

Contents

List of Figures

List of Tables

Algorithms

List of Abbreviations

ACO	Ant Colony Optimization
AI	Artificial Intelligence
ANN	Artificial Neural Network
CPG	Central Pattern Generator
EA	Evolutionary Algorithm
EANT	Evolutionary Acquisition of Neural Topologies
EvoRoF	Evolutionary Robotic Framework
GNARL	GeNeralized Acquisition of Recurrent Links
MLR	Mesencephalic Locomotor Region
NEAT	NeuroEvolution of Augmented Topologies
OS	Output Signal
SNN	Spiking Neural Networks
SSSA	Scuola Superiore Sant'Anna

1 Introduction

1.1 Motivation

This thesis is embedded in the *SYMBRION* project. The *SYMBRION* project is funded by the European Union and has the goal to develop novel principles for multi-robot organisms. These principles are biological inspired and based on modern computing paradigms. The peculiarity of the project is that the robots are able to dock to each other with the help of specific docking elements. Accordingly the individual robots are able to form different types of organisms.

One of the key questions that will be answered in this thesis is the question of common movement of an organism. In single mode the individual robot can move around with the help of its screws, wheels or chains. These types of drives have inherent limits, for example with climbing of stairs. Furthermore, the distributed synchronized control of locomotion of organism is much more difficult than for a single robot.

A possible approach is to have a look to biology to find potential solutions. Locomotion of animals or human beings is an important skill for their survival. It is the only way to find food and sexual partners or to escape from predators. The ability to change position is evolutionarily adapted over millions of years to different environmental conditions. In spite to this huge relevance this area is only slightly researched and understood.

For this thesis the already discovered biological approaches will be used. So Central Pattern Generators (CPG) represent a starting point to get the organisms into action.

1.2 The Symbrion Project

Funded by the European Union the *SYMBRION* project has its focus on investigating and developing novel principles of adaptation and evolution for symbiotic multi-robot organisms. All the different approaches of the project are bio-inspired and are based on modern computer paradigms. The robot organisms consist of super-large scale swarms of robots with the peculiarity that the individual robot is

able to dock to each other. With the help of specific docking elements the robots can configure different structured organisms by their own. The dynamical aggregated organisms collectively interact with the physical world with the help of their sensors and actuators. In this way, artificial robotic organisms become self-configuring, self-healing, self-optimizing and self-protecting. The robots are able to reprogram themselves without human supervision and form new functionality.

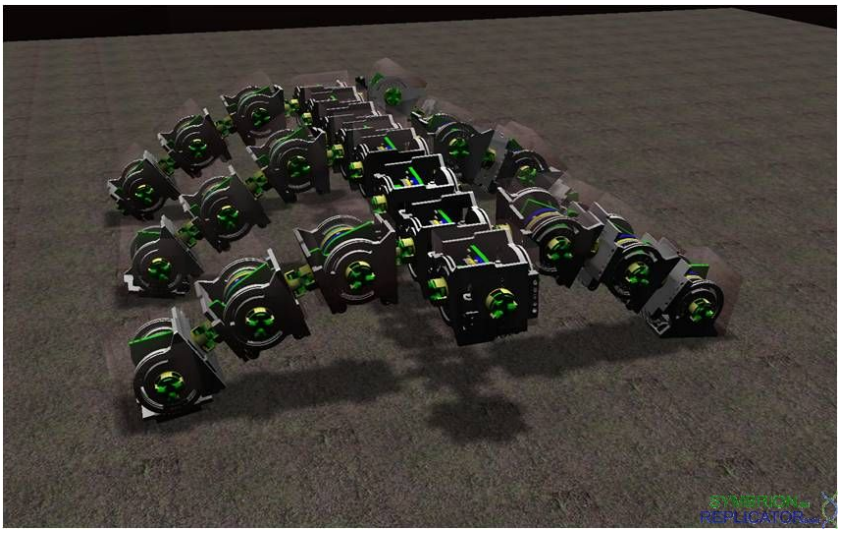


Figure 1.1: A multi-robot organism consisting of 24 single robots which are connected to one big organism called hexapod. This organism can move with the help of his six legs. For example this hexapod is able to go up a stairway. A task a single robot can not cope.

1.3 Related Work

This thesis alludes to a topic which is already described in [Bar10] by Barth. Barth was engaged in designing Central Pattern Generators (CPG) for the modular robotic system of the *SYMBRION* project. In his thesis, Barth compares different widely used modules of CPGs. To derive multiple advantages of the different modules a hybrid approach was used for implementation. The basic idea was to evolve a neural network that drives different oscillators. Barth used GeNeralized Acquisition of Recurrent Links (GNARL) algorithm for the engineering of the networks. For more information about GNARL, see [SAP94]. Robotic sensor data as input for the neural network was only discussed, but not used. Thus, the approach of Barth is not able to react to environmental changes, like obstacles. This is a significant disadvantage in real life.

Another discourse that deals with the topic of Central Pattern Generators of modular robots is explained by Haasdijk, Rusu and Eiben in [HRE10]. The scientific work of Haasdijk et al. is also embedded in the *SYMBRION* project. Like in [Bar10] the CPGs are designed for modular robotic organisms. Haasdijk et al. uses the HyperNEAT (NeuroEvolution of Augmented Topologies) technique to evolve the large-scale artificial neural networks (ANNs). For more information about HyperNEAT, see [BKv09]. A big advantage over the scientific paper of Barth is the integration of the sensor feedback that makes it possible to react to obstacles. In this theoretical approach the evolutionary process does not create an ANN, which produces the signals for the motors of the robots. The output of the artificial neural net are the base angle, amplitude and phase difference for a sine wave - as in most similar work. This defined sine wave is used as control input for the motors. An approach that makes the evolutionary process much faster because it substantially reduces the parameter space. However it has the disadvantage that only sine waves can be generated. Additional Haasdijk et al. did not develop a higher control mechanism to control for example direction and speed of the organism.

Another scholarly piece that is noteworthy at this point is [LFS92]. In the scientific statement of Lewis et al. already artificial neural networks were used to control motor pattern generators for a six legged robot. Staged evolution is utilized to accomplish this goal which is defined implicit by a fitness function. Thus, the evolved artificial neural network is getting more complex over time continuously. Algorithmic a single oscillator is evolved initially. In a second step evolution iteratively adapts the motion pattern generator of the robot.

An alternative to the output of the artificial neural network of the already mentioned papers, is shown in [MNSI04] by Mori, Naamura, Sato and Ishii . Mori et al. designed a Central Pattern Generator for a monolithic bipedal robot. The difference with other scholarly pieces is that they interpret the output of the artificial neural network as torque for the actuators. However, this approach has been developed for a monolithic robot, which contrasts to the modular approach of this thesis.

In the scientific work [KKY$^+$03] and [KKY$^+$05] by Kurokawa and Kamimura et al. automatic generation of motion patterns for modular robots are represented. Kurokawa and Kamimura et al. used the M-TRAN II robot to show their results. Especially in [KKY$^+$05] the Central Pattern Generators are implemented as distributed neural oscillators. It must be noted that only the parameters for the oscillators are evolved. As part of the disclosure locomotion has been evolved for different types of organisms. [KKY$^+$05] only uses the current state of the oscillators as sensory input. There is no higher information about the environment that can be used for motion generation. This leads again to the previously mentioned

hitch of avoiding collision with obstacles. An interesting result of the two publications is, that it would make sense to prohibit global communication. Kurokawa and Kamimura et al. approved only communication between directly connected modules. A fact, that would correspond to the aspect of distributed and autonomy of the distinct modules.

Based on the biological model Auke Jan Ijspeert uses connectionist oscillators in [IHW99]. As a template for the salamander and lamprey locomotion pattern, Ekebergs Central Pattern Generator model was used. This is described in [Eke93]. Ijspeert et al. tried to copy the movement of real animals what does not necessarily lead to distributed approaches, as it would essential for modular robotics. More on this topic can be found in [Ijs01] and [Ijs08] explained by Jispeert.

In [MI05] are important notes to mention. Marbach et al. evolve Central Pattern Generators for locomotion control of modular robots, which are similar to the task of this thesis. By using YaMoR (Yet another Modular Robot) Marbach includes morphology in the self-organizing process. The announcement assumes symmetrical organisms and movements. This limits the search space considerably and contradicts a compilation of a random organism. In this scientific paper the parameters of a sine oscillator are determined by a genetic algorithm.

Furthermore, Izhikevich et al. plays an important role for this thesis. In [Izh03] and [Izh04] the third generation of neural network models, the Spiking Neural Networks (SNN), are introduced. A model that combines the biologically plausibility of Hodgkin-Huxley-type dynamics and the computational efficiency of integrate-and-fire neurons. For additional information see [Maa97] by Wolfgang Maass et al. A more detailed explanation about the Spiking Neurons can be found in Chapter 2.2.1

Kassahun et al. explains in [KS05] the Evolutionary Acquisition of Neural Topologies (EANT). With the help of this method it is possible to evolve the structures and weights of neural networks. A more detailed explanation, about the Evolutionary Acquisition of Neural Topologies, can be found in Chapter 2.2.2

A new biological approach to control modular robotics are represented in [HSSC10]. Hamann et al. control the modular robots of the *SYMBRION* project with the help of virtual hormones. These hormones are represented as a mathematic model in the robot. It might be a new approach that could be taken up and used in future work. The functionality is briefly described in Chapter 2.3.

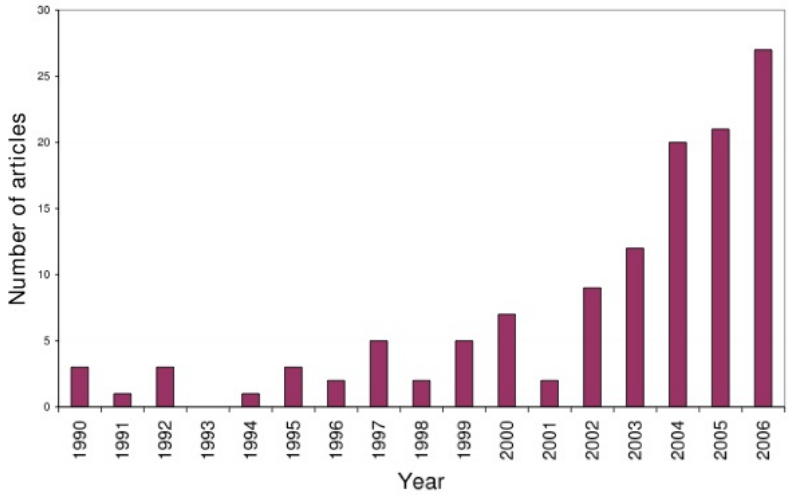


Figure 1.2: Number of articles per year whose abstract contains the terms "robot" and "central pattern generator" or "CPG" in the IEEE Explore database, from 1990 to 2006. Taken from [Ijs08]

1.4 Goals

In contrast to [Bar10] and [HRE10], this thesis will incorporate the identified facts. Any type of oscillations will be produced, so as not to restrict the degrees of freedom. The consequence of doing that is a much bigger search space and a slower evolutionary process. Sensor feedback is a fact which can not be waived because obstacle avoidance is an ability that the robots need to survive in real world.

In contrast to the referred works, this thesis tries to ensure the following points of implementation:

- There will be no restriction to the organisms shape or movement.
- The artificial neural networks will evolutionarily adapted with the help of a fitness function.
- The artificial neural network will be distributed over all modules of the organism.
- Spiking Neural Networks will be used and modified with the help of EANT.

1.5 Structure

The thesis is structured in the following way:

Chapter 1 – Introduction:
Gives a motivation for the thesis and briefly presents the Symbrion project and related scientific work of other authors.

Chapter 2 – Basics:
Explains the basic aspects of robotics especially a focus is set to modular robotics and swarm robotics. Methods of artificial intelligence, like neural networks and evolutionary algorithms, are introduced.

Chapter 3 – Implementation:
Gives an introduction to the used software environment. The implementation of the fitness functions and the evolved controller of this thesis are introduced.

Chapter 4 – Experiments:
Presents the performed experiments and their results.

Chapter 5 – Summary:
There is a summary of the results and an outlook on possible future work.

2 Basics

2.1 Robotics

The word “robotics” was derived from the word robot, which was first introduced by Czech writer Karel Čapek in his play R.U.R (Rossum’s Universal Robots - 1921) [Zun11]. Even before the introduction of the word there were already various types of robots. In the first century and earlier there have been the first descriptions of robot-like automata that were described, for example by Heron of Alexandria. The engineers and inventors were attempted to build self-operating machines which were powered by air pressure, steam and water. The first designs were made for humanoid robots.

As a pioneer, Leonardo da Vinci already put some sketches out in 1495. The design was probably based on anatomical research recorded in his Vitruvian Man. Jacques de Vaucanson exhibited between 1738 and 1739 several life-sized automatons like a flute player, a pipe player and a duck. In 1898 Nikola Tesla demonstrates the first radio-controlled torpedo based on the patents of Teleautomation. 27 years after Karel Čapek published his play the first simple robots were developed with exhibited complex biological behavior. They were named Elmer and Elsie and were created by William Grey Walter of the Burden Neurological Institute at Bristol [Sab11]. Elmer and Elsie could sense light and contact with external objects. With the help of the sensor stimuli they navigate through the environment. In the second half of the 20th century the first autonomous commercial industrial robots were introduced. The name of the first robot was “Unimate” which was developed by George Devol in the 1950s. The development of these digitally operated and programmable robots have evolved over the years up to today’s PUMA - Programmable Universal Manipulation Arm. Furthermore, new robots have been designed for different tasks and environments.

Today, robotics is a rapidly growing field in different research directions. The researchers design and build new robots that serve various practical purposes. Robots are used to perform jobs more cheaply or with greater accuracy and reliability than humans. In addition, they are also employed for jobs which are too dirty, dangerous, or dull to be suitable for humans.

2.1.1 Modular Robotics

"Modular robotics" is one of the newer research direction in the field of robotics. A modular robot is a new breed of robot that is designed to increase the utilization of the robots by modularizing the robot. The functionality and effectiveness of a modular robot is easier to increase compared to conventional robots. One of the main advantage is the variable morphology. This kind of robot typically uses the same conventional actuators and sensors found in fixed-morphology robots, but these robots are able to deliberately change their own shape. In order to do this they are equipped with special connectors. With the help of these connectors they can form organisms which consists of multiple modules.

Butler and Rizzi show some advantages of this modular approach in [BR]. One of the positive aspects which are explained in their work is the modularity used for locomotion. For example a snake-shaped robot is better optimized for narrow tubes while a wheeled robot should be better to overcome long distances with less energy. So the modular robots are able to deliberately change their own shape depending on the current situation and are adapting to the new circumstances and tasks.

An other aspect is the modularity for manipulation. Different tasks require different kind of manipulation tools. The idea of the modular robotics is that the robots can cooperate to solve these tasks. The robots can connect together to form a new manipulator like an arm without the intervention of humans.

A third aspect is the modularity for geometric reconfiguration. Modular robots can be adjusted to fit different production tasks in manufacturing systems.

A fourth and final important aspect that will be discussed here is the modularity for robustness. Defective robots can easily eject from organism and be replaced by functional robots. This is probably the biggest advantage of modular about traditional robotics. Previously the whole defective robot has to be replaced and repaired. Under these circumstances it was not available for a relatively long time. This is especially important in production scenarios, as well as scenarios where a robot can not easily be accessed, for example in space travel. In the latter case, automatic reconfiguration is highly desirable.

In the *SYMBRION* project the modularity for robustness and modularity for locomotion are the focused aspects. This work contributes both of them by providing an efficient way to achieve locomotion with an arbitrary number of modules and shapes of the robotic organism.

2.1.2 Swarm Robotics

Swarm robotics is a new approach to the coordination of multi robot systems. This robot systems consist of a large number of mostly simple physical robots. The idea of this approach is that the desired collective behavior emerges from the interaction between the robots and interaction of robots with the environment. Template for these approaches are biological studies of insects, ants and sphere in nature where swarm behavior occurs. Results of the research are considered in the design and behavior of the robots. The robotic behavior is inspired but not limited by the emergent behavior observed in social insects and is called "swarm intelligence". Even simple rules can produce complex swarm behavior of the robots.

One example in nature that can be a template for robotics and will be shown here are the so-called ant colony optimization (ACO) algorithms. ACO algorithms were conceived to find the shortest route in traveling salesman problem, see [DS04] and [FM08] for a more detailed explanation. The ants optimize the path between their nest and a food source by using pheromones. On ways that are often gone, the pheromone concentration will increase. The pheromones which were distributed away from this paths will fade with the time. Following ants try to follow the traces of odorous and thus increase the track again - positive feedback loop. The result will be a trace represents the shortest path. This only works if a critical certain number of ants are involved. If there are not enough ants, the pheromone trail disappears too quickly and the ants are not able to follow the trace again.

Unlike distributed robotic systems in general, swarm robotics emphasizes a large number of robots and promotes scalability. A key-component is the direct or indirect communication between the individuals, e.g. with the help of pheromones (ants) or by wireless transmission systems like infrared (robots) etc.

Also, the *SYMBRION* project deals with the issue of swarm robotics. This work does not focus directly on the swarm aspect of modular robotics. The main aspect will be the locomotion of the robotic organisms.

2.2 Artificial Intelligence

Artificial Intelligence (AI) is a subsection of computer science. Poole et al. described Artificial Intelligence as "the study and design of intelligent agents" in [PMG98]. The focus of research is the automation of intelligent behavior. While the "strong" artificial intelligence concerned with the imitation of human behavior, the focus of the "weak" artificial intelligence is solving problems of practical use. Meant here are the problems that require a certain amount of intelligence to solve. Ultimately it is about the simulation of intelligent behavior by means of mathematics and com-

puter science. Rather, various methods of science, like neurology or psychology, are used to produce intelligent behavior. In the field of robotics the basic idea is to create systems that can understand the behavior of intelligent beings.

2.2.1 Artificial Neural Networks

In the course of 50 years of research, Artificial Intelligence has developed a large number of tools to solve the most difficult problems in computer science. Based on the field of neurology one of these tools is the Artificial Neural Networks (ANNs). An Artificial Neural Network is a mathematical model that is inspired by the structure of biological neural networks. The ANNs consists of a group of interconnected artificial neurons. The interconnections have specific weights and correspond to the synapses known from biology. Neurons have an intrinsic activation function. That function determine for which activation from its input links, the neuron activates its output links. Often this is specified by a threshold value Θ.

Learning algorithms can be used to train this neural networks. This algorithms, such as EANT (see 2.2.2), modify the structure or the link weights of the ANN. After a training phase, they are able to recognize patterns, classify input data or control robots. For robotics the advantage of ANNs are the possibility to directly link the sensors with the actuators. A nice simple example of this is the Braitenberg vehicle, see Figure 2.1.

Spiking Neural Networks

A specific type of artificial neural network is the Spiking Neural Network (SNN). They form the third generation of neural networks. The model that produces spiking and bursting behavior combines the biologically plausibility of Hodgkin-Huxley-type dynamics and the computational efficiency of integrate-and-fire neurons. Using the accurate Hodgkin-Huxley-type model is computational prohibitive. There were only a handful of neurons to simulate possible in real time. In contrast, using integrate-and-fire model is computational effective. But this model is unrealistically simple and not good enough for simulating rich spiking and bursting dynamics.

In [Izh03] a simple mathematical spiking model is represented, that efficiently combines these two models. Depending on four parameters the explained model reproduces spiking and bursting behavior, shown in Figure 2.2.

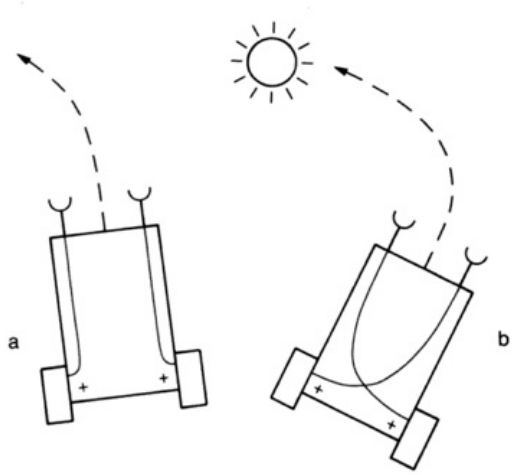


Figure 2.1: Braitenberg vehicle with two light sensors in the front. The sensors are directly linked to the motors of the wheels. A higher sensor value leads to a faster rotating motor. In Fig. (a) the vehicle moves away from light sources, in Fig. (b) the vehicle moves to the light sources. Picture taken from [Bra86].

Central Pattern Generators

The Central Pattern Generators (CPG) are neural networks producing rhythmic patterned output without sensory feedback. Biological researches have shown that the networks responsible for rhythmic locomotion are distributed throughout the lower thoracic and lumber regions of the spinal cord. Examples of movements whose unconscious automatic process made possible by CPGs are walking, running or swimming. The rhythmic movements are based on the alternating activation of flexors and extensors of the corresponding areas of the body. These actions are triggered by an excitatory stimulus from the brain stem called tonic drive. The responsible center of the brain is known as Mesencephalic Locomotor Region (MLR). A weak tonic drive leads to an oscillation with low frequency, while a strong tonic drive increases the oscillations frequency.

2.2.2 Evolutionary Algorithms

An Evolutionary Algorithm (EA) is an optimization method which is inspired by the biological evolution. An initial population P of individuals p is characterized by different properties. The characteristic of the individuals have to maintain against different selection factors. A fitness function rates how well a individual is optimized for a specific task. Rated as well adjusted individuals may reproduce and

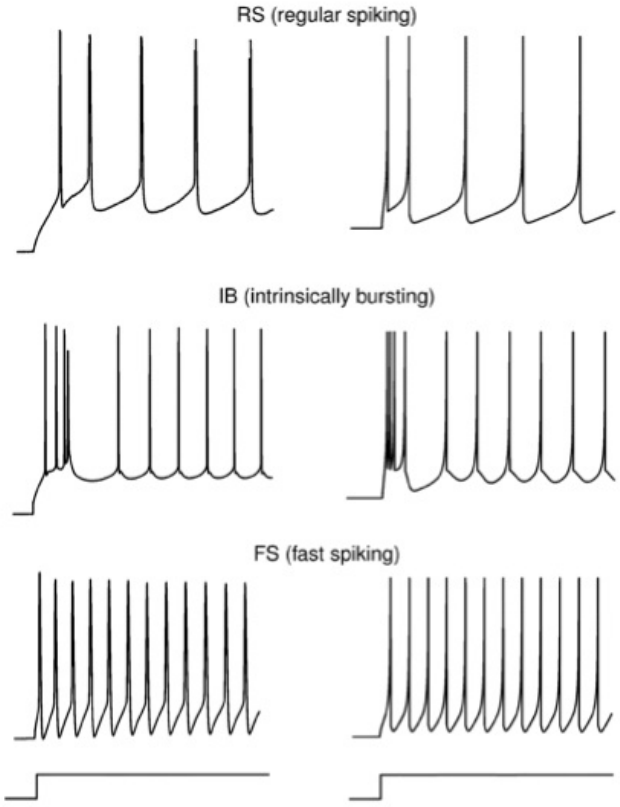


Figure 2.2: Simple spiking neural network model that can reproduce firing patterns of neurons. Picture taken from [Izh03].

pass their traits to the next generation. The next generation receives a part of the genome of both parents. This leads to a recombination of genetic material. In addition to recombination is a certain chance of a mutation. By the mutation individual properties can be changed randomly. As a result, over several runs, the population is more and more adapted to the specific task. For a more detailed explanation to Evolutionary Algorithms, see [ES08].

Common Genetic Encoding

An important recent achievement has presented Kassahun et al. In [KEM+07] Common Genetic Encoding (CGE) is described. This is a method to encode artificial neural networks with significant advantages over previously known procedures. An important step if one has the desire to use artificial neural networks with evolution, because these networks must be encoded there before. CGE has useful properties that makes it suitable for evolving this neural networks. The known methods

of encoding neural networks does not allow a direct evaluation. These methods always need to decode the phenotype from the genotype first. In contrast, CGE allows the implicit evaluation of an encoded phenotype. On the other hand, it is easily possible to decode the CGE encoded phenotype structure of a network. Because the topology is implicitly encoded in the genotype's gene-order. Kassahun et al. show that all major operation of the evolutionary algorithms, as for example mutation and crossing-over, are possible with CGEs. Thus, this method provides the basis for Evolutionary Acquisition of Neural Topologies (EANT). Additional information about CGE can be found in [KMEK09].

Evolutionary Acquisition of Neural Topologies

Evolutionary Acquisition of Neural Topologies (EANT) is an evolutionary reinforcement learning system. It is suitable for learning and adaptation to the environment through interaction. The system evolves both the structures and weights of artificial neural networks. EANT starts the evolutionary process with an artificial neural network of minimal structure. Through the evolutionary path the network increases its complexity. EANT uses the above mentioned CGEs. The evolutionary operations are performed on the encoded linear genome. In comparison to other methods EANT shows remarkably good results for standard benchmark problems. With fewer iterations it achieves considerable results. More information about EANT can be found in [KS05].

2.3 Hormone-Based Controller

In [HSSC10] an artificial homeostatic hormone system is introduced. Hamann et al. explain a bio-inspired control mechanism for robotic systems. The approach is inspired by chemical signal-processing and hormone control in animals. With the help of hormones, the single robot, as well as the whole organism can be controlled. This is precisely in the field of modular robotics very useful as a kind of messaging system. In addition, the controllers can be evolved, what makes them adaptive to different tasks.

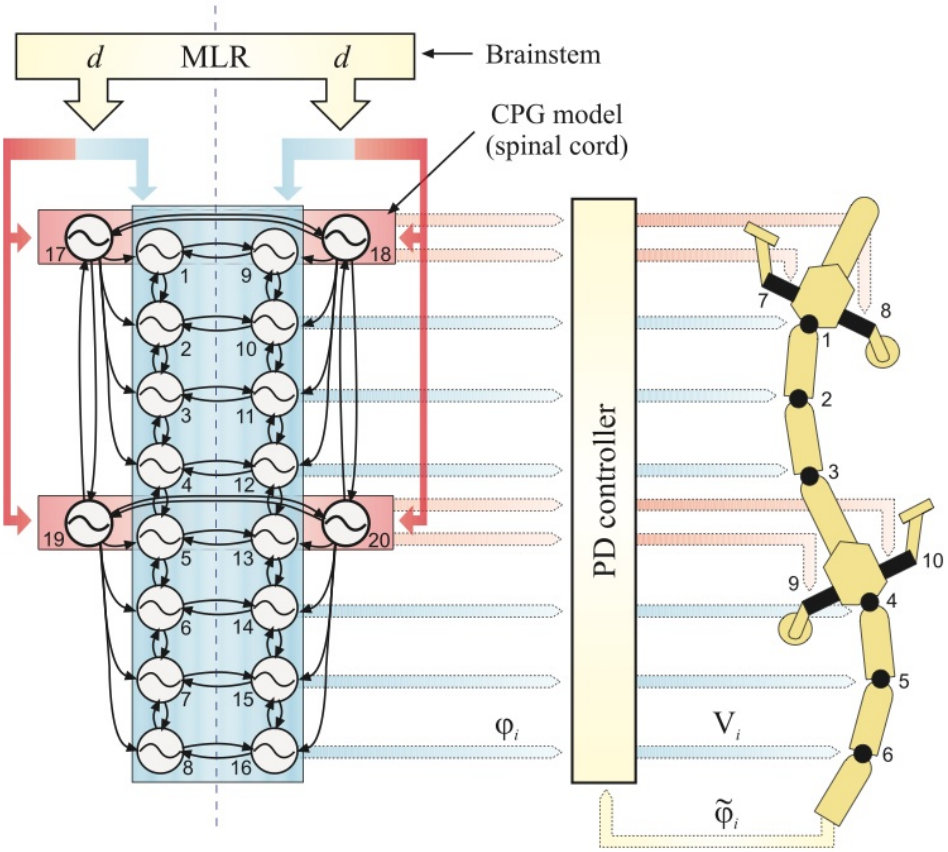


Figure 2.3: Salamander CPG model tested with a amphibious salamander-like robot (Ijspeert, Crespi, Ryzcko, and Cabelguen 2007). The 20 amplitude-controlled phaseoscillators receive a drive d signal. This signal represents the descending stimulation from the Mesencephalic Locomotor Region (MLR) in the brain stem. The outputs of the CPG are desired joint angle positions φ_i. Picture taken from [Ijs08].

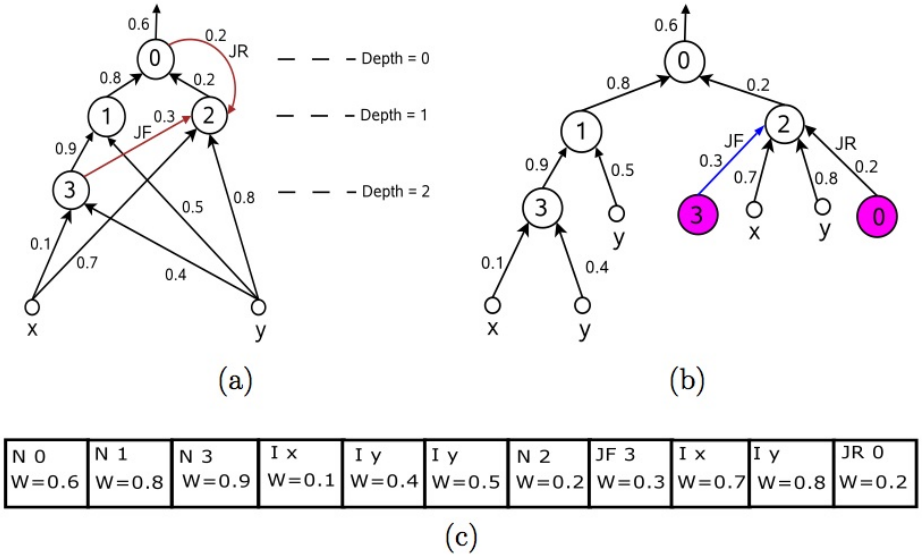


Figure 2.4: An example of encoding an artificial neural network using a linear genome. Fig. (a) An example artificial neural network to be encoded. Fig. (b) The artificial neural network interpreted as a tree structure. Fig. (c) The linear genome encoding the artificial neural network shown in (a). Pictures taken from [KMEK09]. The abbreviations are explained in the scientific paper.

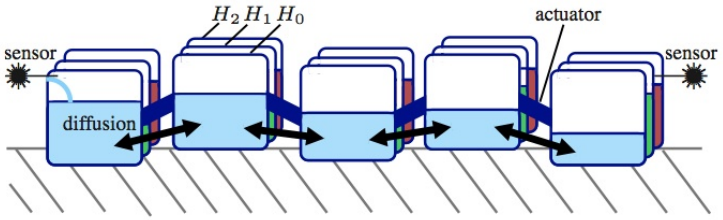


Figure 2.5: Sketch of the hormone dynamics and diffusion processes in a robotic organism. Each module holds different hormones with different concentrations and hormones diffuse through the whole organism. Picture taken from [HSSC10].

3 Implementation

3.1 Software Environment

3.1.1 Robot3D

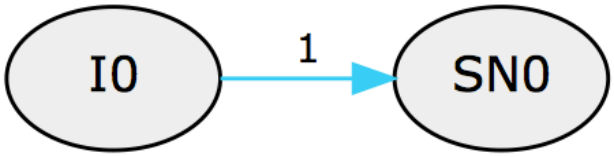
For simulation purpose the Robot3D simulator was used. Robot3D is a free 3D robot simulator built on top of the open-source game engine Delta3D which is available on the launchpad site. The Robot3D simulator is created within the European *REPLICATOR* and *SYMBRION* projects. Prime developer is Lutz Winkler from the "Karlsruhe Institute of Technology". The simulator makes use of OpenSceneGraph, ODE, OpenGL, OpenAL, CEGUI and YARP. The simulator can run multiple robots in parallel and is especially tailored toward modular robots. See [Rob10].

3.1.2 Evolutionary Robotic Framework

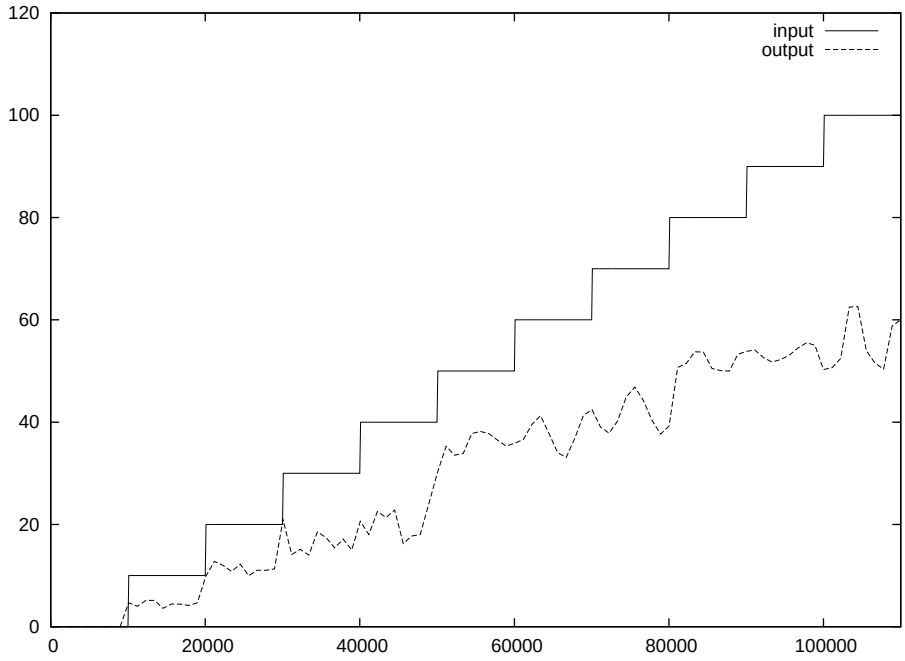
The controller of this thesis uses the Evolutionary Robotic Framework (EvoRoF). EvoRoF is a framework that provides the evolutionary process for robotic systems on the basis of EANT. The prime developer is Florian Schlachter from the University of Stuttgart, who was supported by Katja Deuschl and this thesis. The framework was developed, adapted and adjusted continuously parallel to this work. EvoRoF makes it possible to use neural networks combined with evolution. By the time it is possible to use Artificial Neural Networks or Spiking Neural Networks. During the evolutionary process the framework changes the network structure automatically with the help of parametric and structural mutations. The evolutionary process is directed by integrated selectable fitness functions which are responsible for the selection mechanism on population. In addition, integrated loggers make it possible to document the results.

3.1.3 Behavior of Spiking Neurons

The implemented Spiking Neurons have a characteristic behavior by producing spikes. For demonstration purpose an input neuron of a simple Spiking Neuron Network was activated with different input values. In Figure 3.1 (b) can been seen that the count of output spikes increase with the input value.



(a) The neural network for testing the spike activity



(b) Different activation values give a different number of spikes as output.

Figure 3.1: Behavior of the implemented Spiking Neurons.

X-axis: time steps [0, +∞].
Y-axis: input into the I0 neuron of the ANN [0, +∞].

3.1.4 Implemented Fitness Functions

At this point, the fitness functions will be presented because these are essential to the outcome of the following experiments. In this thesis three fitness functions have been implemented, which will be described below in more detail.

Sine Wave Function

This fitness function is used to evaluate a neural network which is able to produce sinusoidal output signals. The implementation of this function is presented in Algorithm 3.1. Algorithmically first the correct sine value is calculated and saved in variable a. In a second step, the output value of the neural network it determined and saved to variable b. The subtraction of the two values a and b yields the distance of the true value to the calculated one. The fitness value is defined by the mathematical function shown in Figure 3.2. If the distance is small the fitness function rated it high. Otherwise the valuation tends to zero.

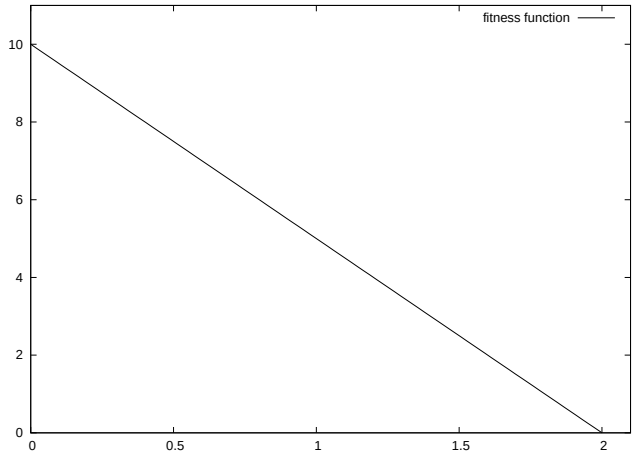


Figure 3.2: This pictures shows the fitness function for the evaluation of a sine wave. If the evaluated value of the neural network is close to the real sine value the fitness function rated it high.

X-axis: distance to the desired value [0, +2].
Y-axis: return value as a parameter of quality [0, +10].

```
void FitnessSineWave::updateFitness( void ){
   fitness += - 5.0 * sqrt(pow((sin(((*worldmodel->getVariableData())[2]
       * PI) / 180.0) - (*worldmodel->getVariableData())[1]), 2.0)) +
       10.0;
  }
}
```

Algorithms 3.1: Fitness Function - Sine Wave Function

```
void FitnessDistance::updateFitness( void )
{
  curPos = worldmodel->getCurrentPosition();
  float distance = sqrt((curPos[0]-lastPos[0]) * (curPos[0]-lastPos[0]) +
      (curPos[1]-lastPos[1]) * (curPos[1]-lastPos[1]));
  fitness += distance;
  lastPos = curPos;
}
```

Algorithms 3.2: Fitness Function - Distance Function

Distance Function

This distance fitness function is used to teach an artificial neural network locomotion implicitly. The implementation of this function can be shown in Algorithm 3.2 The fitness value is determined using the absolute value function in a 2-dimensional space. A mathematical representation of this function can be seen in Figure 3.3. The fitness function rewards a vast progress of the organism with the corresponding higher fitness values. It would be possible to extend this function to the third dimension. With the disadvantage that up- and downward movements of a leg will also lead to a positive evaluation without locomotion of the robot.

Straight On Function

The straight fitness function is an alternative to the distance fitness function. The difference to the previously described function is that this one is direction-specific. The Algorithm 3.2 evaluates a locomotion of the organism in X-direction positive. A mathematical representation of this evaluation process is shown in Figure 3.4.

```
float FitnessStraightOn::getFinalFitness( void )
{
  curPos = worldmodel->getCurrentPosition();
  float fitness = (curPos[0] - lastPos[0]);
  lastPos = curPos;
  if (fitness > 0.0) {
    return 10 * fitness;
  } else {
    return 0.0;
  }
}
```

Algorithms 3.3: Fitness Function - Straight On Function

3.1.5 Implemented Selection Functions

In EvoRoF (see Chapter 3.1.2) are different selection functions implemented. This functions select the best individuals in population which are allowed to come to thenext generation. There are different approaches to find this specific individuals. In this thesis are only two different selection functions are used and implemented.

Elitism

The Elitism Selection Function copies a defined percentage of the best individuals and overwrites the worse individuals of the population. Good and worse individuals are evaluated by the fitness functions. After the replacing process all individuals were mutated to generate diversity.

Elitism and Remain

Elitism and Remain Selection Function is an advantage of the Elitism Selection Function. The selection function copies a defined number of the best individuals and overwrites the worse individuals, as the Elitism Selection. Alternative to the Elitism Selection Function in this one the best individuals are not structural mutated, because this mutation could destroy the evaluated knowledge of this individuals. The idea is that only the parametric mutation adapts this individuals to the task.

3.2 Development

3.2.1 Development of a "Sine Wave Generator"

First a neural network should be evaluated that represents an sine wave generator as it is known from electrical engineering. Because in many scientific works about Central Pattern Generators sine wave generators are used. In most cases of these approaches, only the parameters are evaluated with the help of the artificial neural networks. Especially for snake-like organisms this sinusoidal motion is an advantage. In the experiment, see Chapter 4.1 - Evolving a Sine Wave Generator, will be shown that this kind of motion can also be generated. So this approach is at least as powerful as the approaches which are using the sine-generators.

3.2.2 Development of a "Variable - Sine Wave Generator"

In a second step after a sine-generator has been engendered, an artificial neural network should be evaluated that can also generate variable sine wave frequencies. Different frequencies of the sine waves would make it possible to move the organism with different speeds. This would be the point of application for a higher control mechanism which controls the organism. For the results of this experiment, see Chapter 4.2 - Evolving a Variable Frequency Sine Wave Generator.

3.2.3 Development of a Central Pattern Generator

In the first approach for evolving locomotion, see Chapter 4.3 - Evolving Locomotion, a controller is designed that runs separate on each robot of the organism. So each robot has its own artificial neural network which is individually evaluated. This represents a fundamental approach to swarm robotics in which it is desired to have an autonomous robot. The controller uses the Evolutionary Robotic Framework (see Chapter 3.1.2 - EvoRoF) which has already been described. Within the framework, Spiking Neural Networks were used. The network gets a so-called tonic drive (see Chapter 2.2.1 - Central Pattern Generator) and the four actuator positions of the connected surrounding robots as input. The actuator positions of the neighboring robots serve as an interface between the modules of the organism and will be zero if there is no robot connected. These values are shared using the internal messaging system. The idea is that the evaluated network provides outputs that can be mapped to specific actuator position. It would be expected that the resulting networks depends on the position of the robot in the organism. Individual modules would possibly take over different functions, for example, that of a joint.

3.3 Implementation

This section will shortly introduce the two algorithms that were used in subsequent experiments to generate the sine wave generator signals and the locomotion control signals.

3.3.1 Sine Wave Generator

Algorithm 3.4 shows the implementation of the controller that should generate a sinusoidal signal. At the beginning the world model and the evolutionary engine are initialized. Two float arrays are filled with potential output values of the artificial neural network. This was done in order to reduce the search space of the problem which makes the evolution 20 times faster. Each individual of the population gets a certain time to generate the output signals. By calling the update function of the evolutionary engine the neural network mutates and the signals at the output neurons are calculated. If a spike was generated from the network the corresponding counter variable is incremented. Is the counter greater than one, after five time steps, the corresponding output signal is assumed to be true. The resulting bit vector is interpreted as a binary number that leads to the selection of the above-mentioned array values. This calculated value is compared with a normal sinusoidal signal and rated by the implemented fitness function.
The illustrated version shows the algorithm to generate a variable frequency. The input values of the artificial neural network change during evaluation. For the continuous sine wave signal controller the input signal does not change.

3.3.2 Locomotion Controller

Algorithm 3.5 shows the implementation of the controller for organism locomotion. The used artificial neural network receives a so-called tonic drive for stimulating the network. As in the algorithm for the sine generator this one works with a predefined array. In addition, the associated robots are communicating with each other and share their angular positions. These angular positions are fed into the respective local neural network. Also in this approach, the output neurons of the artificial neural network are counted and interpreted as a binary value. This binary value represents the position and hence the associated value in the predefined array. Using this value, the hinge element is driven.
The evolutionary engine rated how well the robot moves with the help of the angular positions. This drives the evolution in the preferred direction - to locomotion of the robot.

```
int main(int argc, char *argv[]) {
 WorldModel * wm = new WorldModel(0,2);
 EvoEngine *evo = new EvoEngine();
 float val[2] = {0.7, 1.0};
 float sig[4] = {0.0, -1.0, 1.0 , 0.0};
 std::vector<float> inputVector;
 std::vector<float> outputVector;
 evo->initEvoEngine((char*) "/home/alschbpk/projects/sinewave/island.cfg",
      1, wm, &inputVector, &outputVector);
 wm->setVariableData(&outputVector);

 int steps = evo->getNumberOfSteps();
 int spike0 = 0;
 int spike1 = 0;
 int spike2 = 0;

 inputVector[0] = 0.0;

 for (int i=0; i < steps; i++) {
  evo->update();
  if (outputVector[0] > 0.0) { spike0++; }
  if (outputVector[1] > 0.0) { spike1++; }
  if (outputVector[2] > 0.0) { spike2++; }
  if (i%5 == 0) {
  if(i%240 > 139) {
    if(i%240 > 200) {
    inputVector[0] = 14.0; outputVector[4] += 45.0;
    } else {
     inputVector[0] = 0.0; outputVector[4] = 0.0;
    }
  } else {
    if (i%240 > 59) {
    inputVector[0] = 7.0; outputVector[4] += 45.0/2.0;
    } else {
    inputVector[0] = 0.0; outputVector[4] = 0.0;
    }
 }
    if (spike0 > 1) { spike0 = 1; }
    if (spike1 > 1) { spike1 = 1; }
    if (spike2 > 1) { spike2 = 1; }

    float a = (sig[spike0 + (2 * spike1)]);
    float b = (val[spike2]);
    outputVector[3] = a*b;
    spike0 = spike1 = spike2 = 0;
  }
  if (outputVector[4] == 360.0) { outputVector[4] = 0.0; }
  evo->evaluateFitness();
 }
}
```

Algorithms 3.4: Implementation SineWave Generator

```
void LocomotionAgent::onTickMessage() {
  inputVector[4] = 50.0; // TONIC
  float f[8] = {-0.3,0.3,0.0,0.6,-0.6,0.0,1.0,-1.0};
  ticks++;

  updateWorldModel(); //UPDATE WORLDMODEL

  //INPUT NN[1..4]: OUTPUTS FROM THE SURROUNDING NEURAL NETWORKS
  Message *msg;
  while ((msg = receiveNextOrgMessage()) != NULL){
    Msg *m = (Msg *)msg->getData();
    float x = (25.0 * (m->val)) + 25.0;

    if (!(msg->getSender().compare("0"))) {
      inputVector[0] = x;
    } else if (!(msg->getSender().compare("1"))) {
      inputVector[1] = x;
    } else if (!(msg->getSender().compare("2"))) {
      inputVector[2] = x;
    } else if (!(msg->getSender().compare("3"))) {
      inputVector[3] = x;
    }
  }
  evoEngine->update();

  //COUNT SPIKES IN ONE FRAME (10 TICKS)
  if (outputVector[0] > 0.0) { spikes0++; }
  if (outputVector[1] > 0.0) { spikes1++; }
  if (outputVector[2] > 0.0) { spikes2++; }

  if (ticks % 10 == 0 ) {
    if (spikes0 > 0) { spikes0 = 1; } else { spikes0 = 0; }
    if (spikes1 > 0) { spikes1 = 1; } else { spikes1 = 0; }
    if (spikes2 > 0) { spikes2 = 1; } else { spikes2 = 0; }

    float v = f[spikes0 + (2*spikes1) + (4*spikes2)];
    moveActuator(0, v);
    spikes0 = spikes1 = spikes2 = 0;

    //SEND MESSAGE WITH OUTPUT DATA
    Msg m;
    m.val = v;

    inputVector[0] = inputVector[1] = inputVector[2] = inputVector[3] =
        0.0;

    sendOrgMessage("A", &m, sizeof(m));
    sendOrgMessage("B", &m, sizeof(m));
    sendOrgMessage("C", &m, sizeof(m));
    sendOrgMessage("D", &m, sizeof(m));
  }
  evoEngine->evaluateFitness();
}
```

Algorithms 3.5: Implementation Locomotion Controller

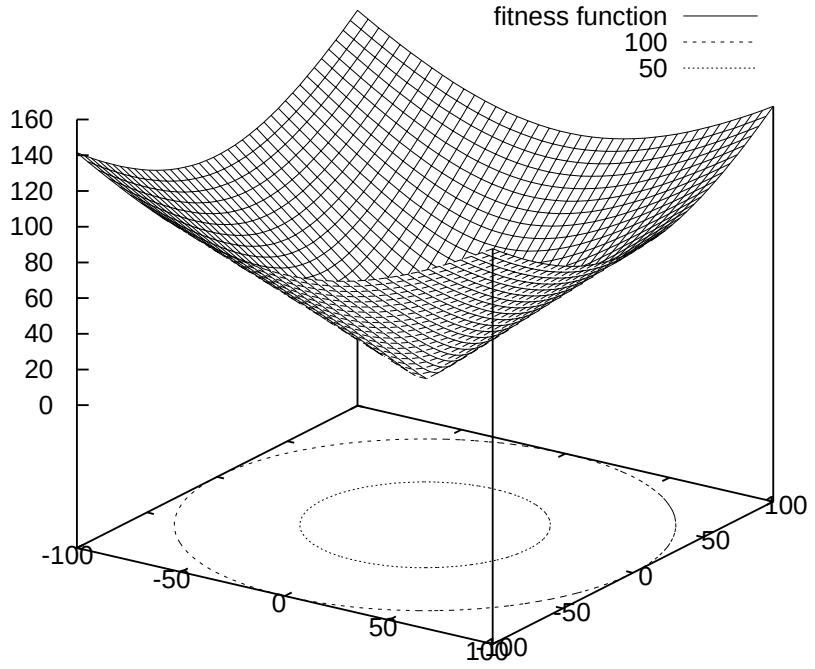


Figure 3.3: This pictures shows the distance fitness function for the evaluation of locomotion. The fitness function evaluates a wide progressing with higher values, regardless in which direction.

X-axis: movement of the robot in X direction [-∞, +∞].
Y-axis: movement of the robot in Y direction [-∞, +∞].
Z-axis: return value as a parameter of quality [0, +∞].

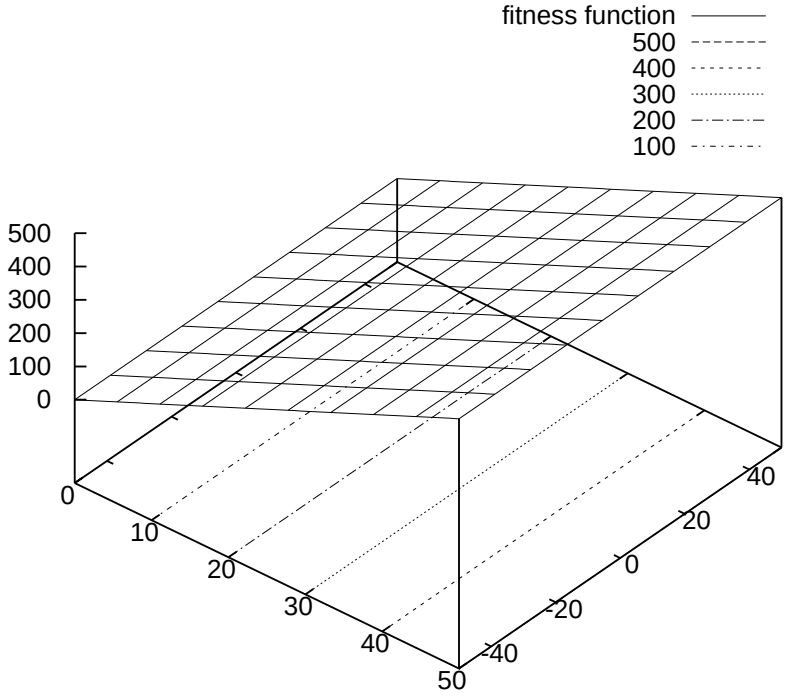


Figure 3.4: This pictures shows the "straight on" fitness function for the evaluation of locomotion in a specific direction. The fitness function evaluates a wide progressing in X-direction with higher values.

X-axis: movement of the robot in X direction [-∞, +∞].
Y-axis: movement of the robot in Y direction [-∞, +∞].
Z-axis: return value as a parameter of quality [0, +∞].

4 Experiments

4.1 Evolving a Sine Wave Generator

4.1.1 Fitness Function

For the first experiment "Evolving a Sine Wave Generator" the specifically for this experiment implemented sine wave fitness function was used. More information about the algorithmic realization of this function can be found in Chapter 3.1.4 - Implemented Fitness Functions.

4.1.2 Parameters

The parameters which were set in the config file can be seen in Table 4.1.

The used artificial neural network has only one input neuron which is a "bias" neuron that is necessary for the activation of the network. Normally one output neuron would be enough for the evaluated signal output. However, in this experiment were three output neurons elected because of performance reasons. For more information about the basements see subchapter "Results - From Spike to Numeric Value".

The population size was set to 100 individuals. All individuals were evaluated for 100 time steps and parametric mutated after each generation. After every tenth generation a structural mutation was performed instead of a parametric one.

For selecting the well adapted individuals the selection function "Elitism and Remain" was taken. More information about that selection mechanism can be found in Chapter 3.1.5.

4.1.3 Simulation

For this kind of experiment no simulation environment was needed. Because of performance reasons the simulation was dispensed and the evaluated values of the artificial neural network were written directly in to a log file.

Parameter	Value	Description
Evolutionary Algorithm Type	SNN	defines the type of network
Using Phenotype Mapping	1	activates phenotype mapping, note CGE can operate directly on genome
Population Size of Island	100	size of the start population of an island
Number of Input Neurons	1	number of input neurons
Number of Output Neurons	3	number of output neurons
Number of Initial Mutations	10	number of mutations which should be performed to a random start genome
Number of Steps for Evaluation	100	number of steps a genome will be evaluated before switching to the next one
Number Of Generations	200	total number of generations
Number of Exploitation Steps	10	number of exploitation steps to perform before a exploration step follows
Default Learning Rate	0.3	Default learning rate for neuron if genome is generated randomly
Default Mutation Probability	0.3	Default mutation probability for a neuron if genome is generated randomly
Fitness Function Type	SINEWAVE	defines the type of fitness function to use
SelectionMode	ELITISMaREMAIN	defines the type of selection mechanism
Selection Parameter	20.0	parameter to pass to the selection function
EnableParametricMutation	ON	enables parametric mutation
EnableStructuralMutation	ON	enables structural mutation
NewNeuronToLinkRatio	0.5	0.4 means during structural mutation, 40% probability for new neurons and 60% probability of a new link

Table 4.1: Parameters for Evolving Sine Wave.

4.1.4 Result

The result of the experiment shows that it is possible to produce sinusoidal signals with the help of the implemented artificial neural networks which are using the spiking neurons. Consequently it can be assumed that the SNN approach is equal powerful as the other approaches with the already frequently used sine wave generators.

One of the computed output signals of a generated artificial neural network can be seen in Figure 4.1 (b). It reflects the result of an evolutionary process for a period of 200 generations with 100 individuals. The best evaluated artificial neural network produces rhythmic signals nearly corresponding to the discretized shape of a sine wave. This represents an almost perfect illustration of the desired signal. It can be assumed that it also would be possible to produce the requested signal absolute correctly with the assignment of additional time respectively more generations.

The relatively simple artificial neural network appertaining to the specified output signal can be seen in Figure 4.1 (a). The reason for this simplicity is the manner of implementation and the mapping of the generated spiking neuron output signals. A fact that should be explained in the following subsection:

From Spike to Numeric Value

To get from the representation by spikes to a numeric value the spikes have to be counted over a period of time. The implemented spiking neurons have to regenerate themselves after each firing before a new spike can be produced. This is the reason, not in every time step a spike can be generated, which should be clarified in Figure 3.1 - Behavior of the implemented Spiking Neurons.

For example with a bias activation of 20.0 course of 100 time steps approximately 10 spikes are generated. This leads in the case of the sinusoidal signal i.e. to the following mathematical representation:

$$f: \{n \in \aleph \mid 0 \leq n \leq 10: n \; count \; of \; spikes\} \rightarrow [-1.0, +1.0]$$

$$i.e. \; f(n) = 1/5 * n - 1$$

A discretization with ten equidistant interpolation points would require one hundred time steps. For more accuracy more spikes have to be counted over a longer time which would slow down the evolutionary process considerably. Because of that time problem a different type of representation has been used in this thesis. This kind of representation reduces the search space dramatically. However, it re-

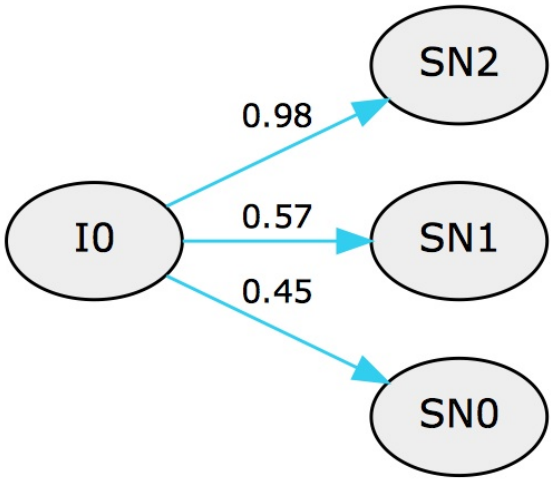
quires more output neurons and hence more physical memory.
The approach is to start with an array of length 2^n with potential output values of the artificial neural network. The length of the array directly determines the accuracy of the discretization. To select an appropriate value of the array n output neurons are needed. This n spiking neurons produce or produce not a spike in a given time window which is many times smaller than 100 time steps. In this elaboration the window is adjusted to 5 time steps which leads to a 20 times faster run time. If there are more than one spike produced by the spiking output neuron during this period of five steps this output signal is interpreted as being present. This kind of output of the artificial neural network can be interpreted as bit vector which represents a binary number. Transferred to the decimal number system represents the binary number one position in the predefined array and thus the associated output value. This approach makes it possible to evaluate a larger number of generations and individuals in less time.

As already mentioned a high accuracy demands many output neurons. But at this point it has to be also noted that for many cases 100% accuracy is not required. Hence, even small predefined arrays are enough to obtain acceptable results, as shown in this and the following experiments. It is always a compromise between speed and accuracy. For that reason, in this study the decision was made to speed. It can be seen in Figure 4.2 that the fitness values of the individuals increase quickly and lead to good and acceptable results.

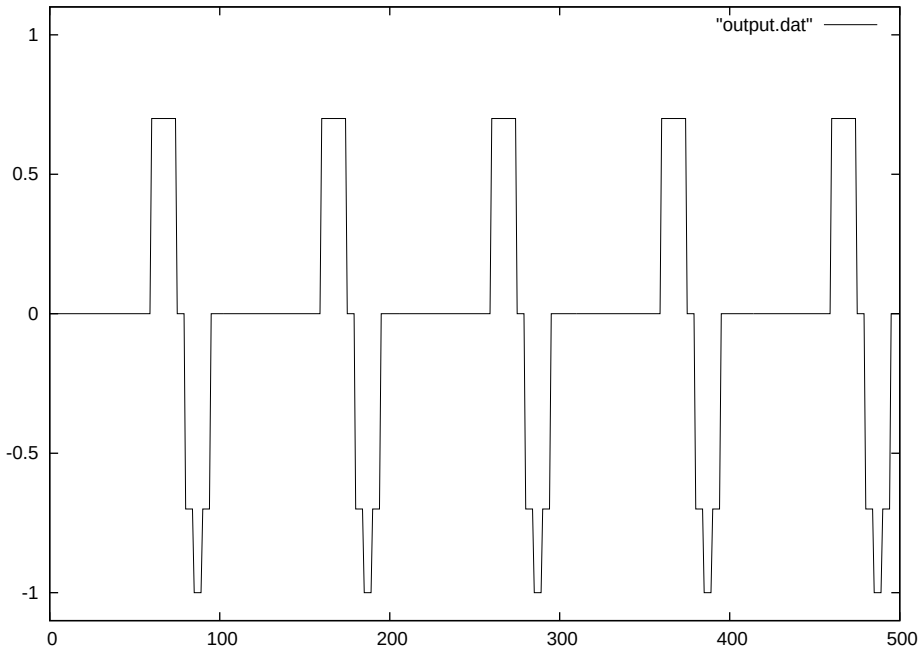
Due to the decision of this type of representation the resulting artificial neural networks are less complex, see Figure 4.1 (a). Depending on how the predefined arrays are chosen it is the ability to select the right values from the array that has to be learned from the ANN. In this experiment it could be demonstrated that for this task no hidden layer is needed. The artificial neural network is able to select the corresponding values from the predefines array directly. The fitness values which are documented in Figure 4.2 suggest that an acceptable fitness level is reached already after a short time of 30 generations.

4.1.5 Averages over 40 runs

In order to exclude individual random results the experiment was repeated 40 times. In Figure 4.2 the recorded average fitness values are presented. The X-axis correspond to the individual generations while the Y-axis represents the fitness value which is determined by the implemented fitness function. It is noticeable that the averaged fitness value titled “average average” (dashed line) is reduced depending on each structural mutation after every tenth generation. A drop of fitness is perceived as bad, but these structural mutations are required to make the artificial neural networks more diverse. Structural mutations make it possible that



(a) The best evaluated network after 200 generations.



(b) The evaluated output signal of the evolved artificial neural network.

Figure 4.1: An evolved artificial neural network and its sinusoidal output signal generated after an evolutionary process for a period of 200 generations.

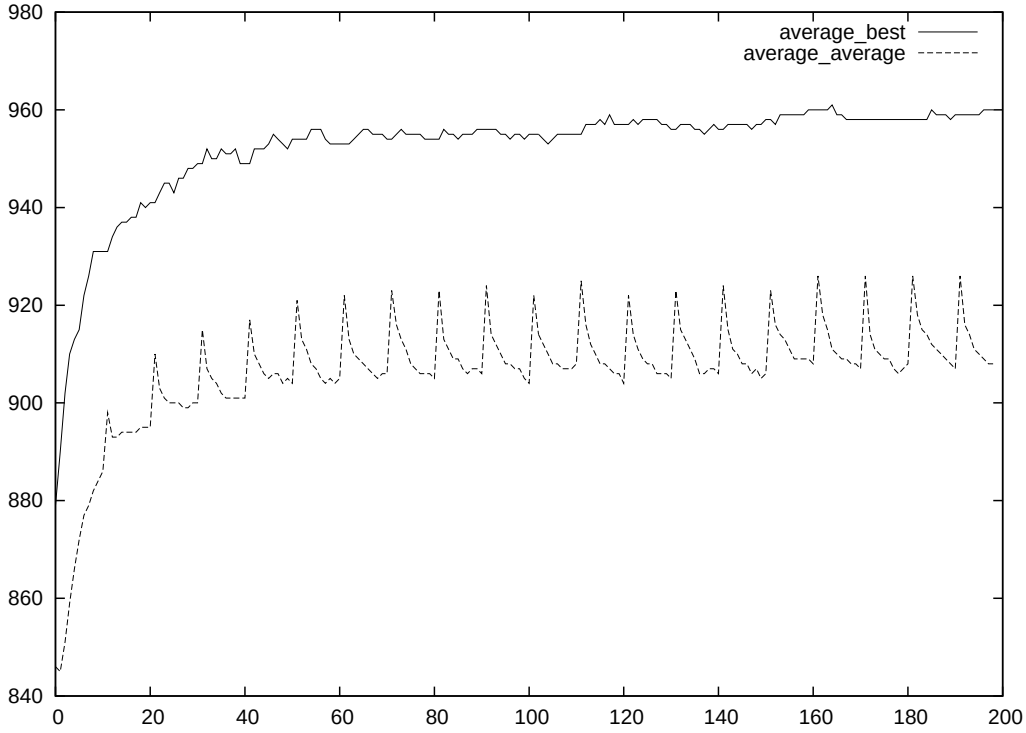


Figure 4.2: The diagram of the averaged fitness values of the entire population (dashed line) and the averaged fitness of the best in the respective population (solid line). Both are averaged over 40 runs.

the network optimization function escapes from local minima in search space. Additional the subsequent parametric mutations generate again an increase in fitness. It is very difficult to get a evidence of an optimal ratio between parametric and structural mutation. Under certain circumstances this ratio can vary addicted to the task. Looking at the trend of the best individuals, titled “average best” (solid line), the chosen ratio should be sufficient.

For example the evaluated sample artificial neural network (4.1 - First Experiment) has an evaluated fitness value of 991,79 of 1000.0 possible. This is an acceptable good result that was already reached after 200 generations.

4.2 Evolving a Variable - Sine Wave Generator

4.2.1 Fitness Function

For the second experiment "Evolving a Variable - Sine Wave Generator" the same fitness function as in the first experiment was used, because the reference signal is equal. More information about the implementation of this function can be found in Chapter 3.1.4 - Implemented Fitness Functions.

4.2.2 Parameters

The parameters which were set in the config file can be seen in Table 4.2.

In this experiment the used artificial neural network has again only one input neuron. But the difference to the previous experiment is that the input value is modified during the evolutionary process. The number of output neurons is similar to the experiment before on the basis of equal performance arguments. For more information about that basements see subchapter 4.1.4 “Results - From Spike to Numeric Value” of the first experiment.

The population size was set to 100 individuals. All individuals were evaluated for 100 time steps and parametric mutated after each generation. After every tenth generation a structural mutation was performed instead of a parametric one.

For selecting the well adapted individuals the selection function “Elitism and Remain” was taken. More information about that selection mechanism can be found in Chapter 3.1.5.

4.2.3 Simulation

For this kind of experiment no simulation environment was needed. Because for performance reasons the simulation was dispensed and the evaluated values of the artificial neural network were written directly to a log file.

4.2.4 Results

The result of the second experiment shows that it is possible to produce sinusoidal signals with different frequencies with the help of the implemented artificial neural networks with the particularity that these networks are using spiking neurons. Consequently now it should be possible to evaluate Central Pattern Generator with this kind of artificial neural networks. In the first experiment is demonstrated that this approach is equal powerful as the approaches with the sine wave generators. The results of the second experiment extends the functionality of the demonstrated

Parameter	Value	Description
Evolutionary Algorithm Type	SNN	defines the type of network
Using Phenotype Mapping	1	activates phenotype mapping, note CGE can operate directly on genome
Population Size of Island	100	size of the start population of an island
Number of Input Neurons	1	number of input neurons
Number of Output Neurons	3	number of output neurons
Number of Initial Mutations	10	number of mutations which should be performed to a random start genome
Number of Steps for Evaluation	240	number of steps a genome will be evaluated before switching to the next one
Number Of Generations	200	total number of generations
Number of Exploitation Steps	10	number of exploitation steps to perform before a exploration step follows
Default Learning Rate	0.3	Default learning rate for neuron if genome is generated randomly
Default Mutation Probability	0.3	Default mutation probability for a neuron if genome is generated randomly
Fitness Function Type	SINEWAVE	defines the type of fitness function to use
SelectionMode	ELITISMaREMAIN	defines the type of selection mechanism
Selection Parameter	20.0	parameter to pass to the selection function
EnableParametricMutation	ON	enables parametric mutation
EnableStructuralMutation	ON	enables structural mutation
NewNeuronToLinkRatio	0.5	0.4 means during structural mutation, 40% probability for new neurons and 60% probability of a new link

Table 4.2: Parameters for Evolving Sine Wave with Variable Frequency.

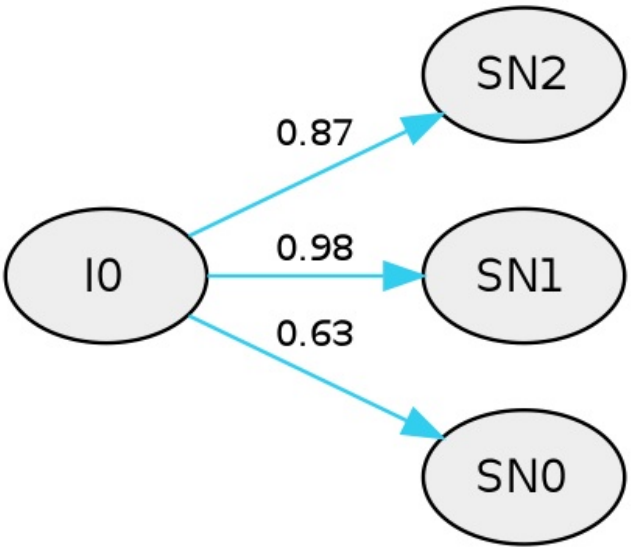


Figure 4.3: An evolved artificial neural network after an evolutionary process for a period of 200 generations. This network is able to produce two different frequencies of sine waves depending on the input signal which is applied to input I0. It should be noted that this network is associated with a predefined array to make this behavior possible with its simple structure.

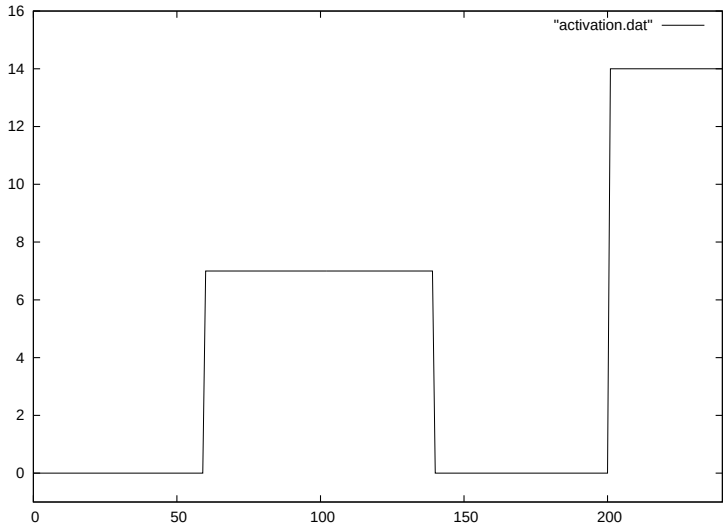
spiking neural networks with additional frequency variability. This variability makes it possible to control various parts of the robotic organism in different ways. A steering movement should be possible and different speeds of locomotion would be generable.

In Figure 4.4 (b) the result of the evolutionary process after 200 generations is exposed. The calculated fitness value for this network is 2151.62 of 2400.0 possible. This is an acceptable good result reached already after that short time looking to the numerical dump of the network and the real sinus. The evolved artificial neural network in Figure 4.3 produces rhythmic sinusoidal output signals according to the injected input signal which can be seen in Figure 4.4 (a).

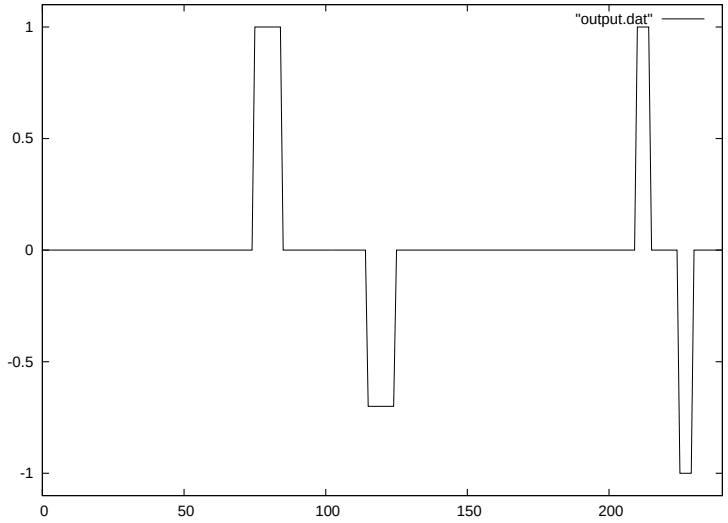
4.2.5 Averages over 40 runs

As in the previous experiment also in this one 40 runs were performed in order to exclude random results.

Figure 4.5 shows the averaged fitness values of the best individuals (solid line) and



(a) The injected input signal of the artificial neural network.



(b) The evaluated output signal of the artificial neural network.

Figure 4.4: The input-dependent (a) sinusoidal output signal (b) generated by the best artificial neural network after an evolutionary process for a period of 200 generations.

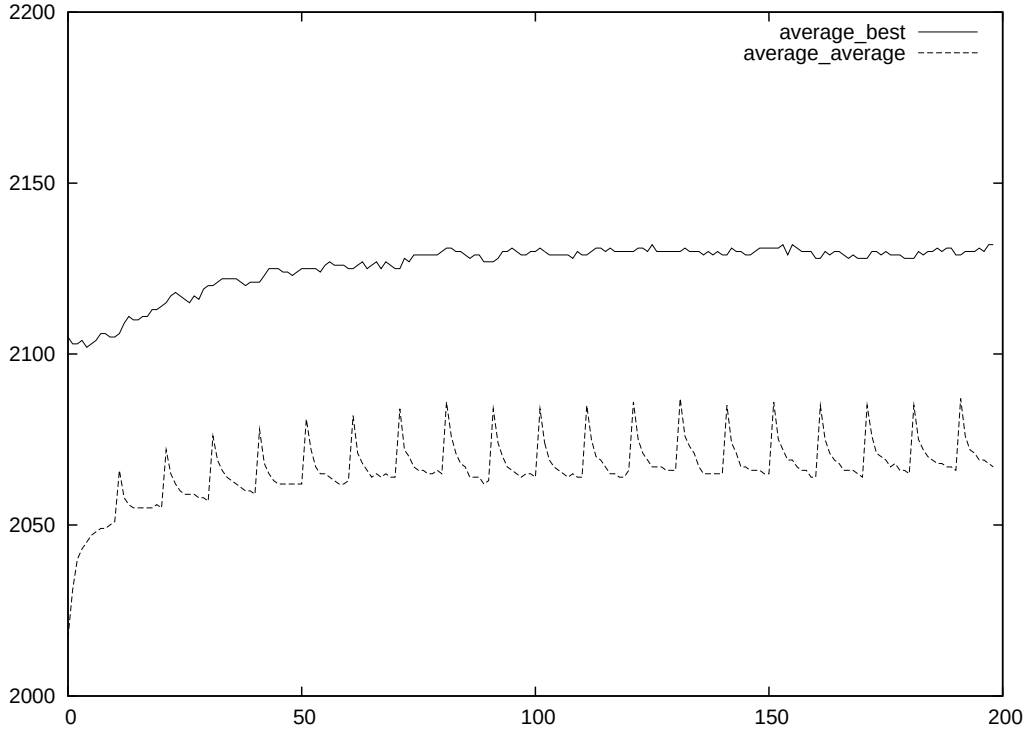


Figure 4.5: The diagram of the averaged fitness values of the entire population (dashed line) and the averaged fitness of the best in the respective population (solid line). Both are averaged over 40 runs.

the averaged averages (dashed line) of the 40 runs for a period of 200 generations. The X-axis correspond to the individual generations while the Y-axis represents the fitness value which is determined by the implemented fitness function. As well as in the first experiment it could be shown that the fitness values of the best individuals increase continuously. These fitness values will slowly approximate the maximum which supports the evolutionary approach.

Just as before the characteristics of the averaged averages (dashed line) is the same. That calculate fitness is reduced depending on each structural mutation after every tenth generation. This can be explained by the same argument of the structural mutation as before, see Chapter 4.1.5 - Average over 40 runs.

In conclusion of the two realized experiments can be said that the artificial neural networks produce acceptable good results after a short time within 200 generations. These networks are able to generate sinusoidal signals with different frequencies depending on the input signals.

This makes the presented approach powerful enough to use it for organism locomotion with the help of Central Pattern Generators (CPG).

4.3 Evolving Locomotion

4.3.1 Fitness Function

The following experiments deal with the topic of locomotion with the help of CPGs. For this reason a fitness function for all experiments was used which is able to rate the advancement of the organism in a virtual arena. The previous described "Straight On" fitness function was used for all of these experiments. For more information about the implementation of that function, see Chapter 3.1.4 - Implemented Fitness Functions.

4.3.2 Simulation

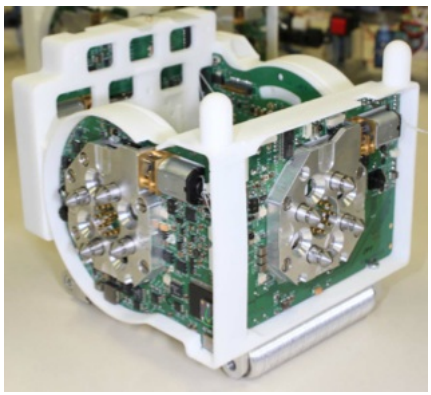
For simulation purpose the Robot3D (Chapter 3.1.1) environment was used all the time. This kind of simulator makes it possible to simulate the physical conditions and the interaction between the robot and its surrounding. The different robot platforms of the Symbrion project and their installed sensors and actuators are directly available in simulation environment. Furthermore it is possible to dock and undock the single robots to an organism like in an authentic scenario in real world. A fact this simulator has abundantly clear advantages over comparable simulators.

4.3.3 Results of the Snake-Shaped Organism

Figure 4.6: A snake-shaped robot organism consisting of five individual autonomous robots from the University of Karlsruhe.

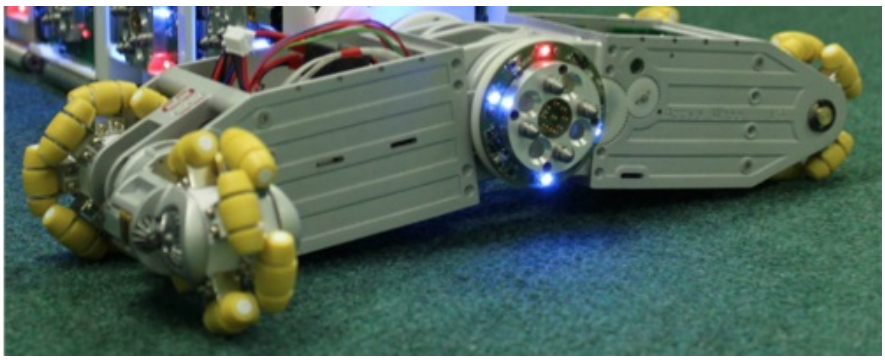
1 2 3 4 5

Table 4.3: The distributed identification numbers for each robot of the organism.



(a) Kabot from the University of Karlsruhe.

(b) Scout robot from SSSA - Italy.



(c) Active Wheel from the University of Stuttgart.

Figure 4.7: An overview of the three robot platforms of the Symbrion project. Pictures taken from [KLM+11].

With the snake-shaped organism pictured in Figure 4.6 were two documented runs performed and analyzed in the following section.

The used robot organism consisted of five individual autonomous robots from the University of Karlsruhe (Figure 4.7 a) that are fixed docked to each other. This kind of robot is one of the three available robot platforms developed during the Symbrion project. In Figure 4.7 the remaining robots of the project can be appraised.

At this point it must be emphasized that each robot has its own population of individuals. The size of that population is defined in the configuration file. Thus, each robot will evaluate its own artificial neural network. This represents a distributed heterogeneous approach with hard on-board evolution.

The idea behind this structure is not to be dependent on a control module. Each robot will keep its own artificial neural network which will be rated for each robot by the respective fitness function. Only the angular positions of the directly connected neighbors are fed into its own artificial neural network. This is achieved via the internal implemented messaging system.

On the basis of this approach the position of each robot in the organism is of relevance. It is assumed that the evolutionary process emerges certain functions of the individual robot depending on its own position in the entire organism. For example a robot accepts a responsibility to be a knee element or a part of the backbone.

First Run: Locomotion for a Snake-Shaped Organism

For sake of completeness, each run the list of parameters were added because only little changes would lead to different results. The results were not reproducible without this parameter listing.

Parameters

For the first experimental run for evolving locomotion the parameters in Table 4.4 were set. The parameters were chosen as in the previous experiment.
All robots of the entire organism are "equipped" with a population of 10 individuals. Each individual is evaluated and has the chance to show how well the current used artificial neural network is adapted to the task of locomotion during this set time period of 800 time steps. Because the evolutionary process is not terminating the "Number Of Generations" parameter is set to undefined but in real the simulation was aborted after 30 generations.

The individuals which are rated high by the already mentioned "Straight On" fitness function were selected by the "Elitism" selection mechanism. For more information about that kind of selection see Chapter 3.1.5 - Implemented Selection Functions.

Morphologically the ANNs consist of five input neurons and three output neurons. One of that five neurons has the function of the "bias" neuron and gets an uniform input signal to stimulate the network. The remaining four are acting as input neurons for the angle positions of the neighboring robots. These angles are transmitted via the implemented internal messaging system.
As before only one output neuron is enough to control the hinge of the robot. In this approach three neurons were elected because of the already mentioned performance reasons.

The ratio between parametric and structural mutation were chosen as in the first two experiments. Every tenth generation a structural instead of a parametric mutation is arranged. However, there is a small difference in the starting condition. In this experiment initially five structural mutations were performed to start with a greater diversity of artificial neural networks.

Discussion

This section should briefly presents the results of the evolutionary process respectively the success of snake-like locomotion.

At the beginning of the experiment the robot organism consist of five individual robots is lying flat on the floor in the virtual arena. Accompanying photos of the initial starting generation and its behavior can be seen in Figure 4.8. On each robot the ANNs were initialized and from now on each individual gets the chance to demonstrate its skills.

In Figure 4.9 the expected (solid line) and the real angular positions (dashed line) of the individual robots were recorded for a time period of 800 ticks. This time period corresponds to the duration of the assessment of one individual on each of the robots. At time step 0 all hinges are at a central position. Meaning that all initial artificial neural network produces output signals which leads the hinges to position: -1. This corresponds to a hinge angle of -90 degrees. For this reason the snake-shaped organism curls at the beginning.

Amazingly already in the subsequent generations can be seen that the organism performs a snake-like locomotion. For example in Figure 4.10 the organism is moving sinusoidally over the ground already in the second generation.

Looking at the same generation to the hinge angles (Figure 4.11) different rhythm patterns for the individual modules are discernible. Some of the recorded rhythms are synchronized to each other while others behave asynchronously but phase shifted. Additional robots in the middle of the organism show greater amplitudes in contrast to the peripheral positions. Perhaps the several elements have already taken over different functions.

In Figure 4.17 the data has been documented over a period of 1000 time steps. At step 224800 the evolutionary engine changes the individual on each robot of the organism. This results in different output patterns which are generated by different ANNs.

Parameter	Value	Description
Evolutionary Algorithm Type	SNN	defines the type of network
Using Phenotype Mapping	1	activates phenotype mapping, note CGE can operate directly on genome
Population Size of Island	10	size of the start population of an island
Number of Input Neurons	5	number of input neurons
Number of Output Neurons	3	number of output neurons
Number of Initial Mutations	5	number of mutations which should be performed to a random start genome
Number of Steps for Evaluation	800	number of steps a genome will be evaluated before switching to the next one
Number Of Generations	undefined	total number of generations
Number of Exploitation Steps	10	number of exploitation steps to perform before a exploration step follows
Default Learning Rate	0.3	Default learning rate for neuron if genome is generated randomly
Default Mutation Probability	0.3	Default mutation probability for a neuron if genome is generated randomly
Fitness Function Type	STRAIGHTON	defines the type of fitness function to use
SelectionMode	ELITISM	defines the type of selection mechanism
Selection Parameter	20.0	parameter to pass to the selection function
EnableParametricMutation	ON	enables parametric mutation
EnableStructuralMutation	ON	enables structural mutation
NewNeuronToLinkRatio	0.5	0.4 means during structural mutation, 40% probability for new neurons and 60% probability of a new link

Table 4.4: First Run: Parameters for Evolving Locomotion with a Snake-Shaped Organism.

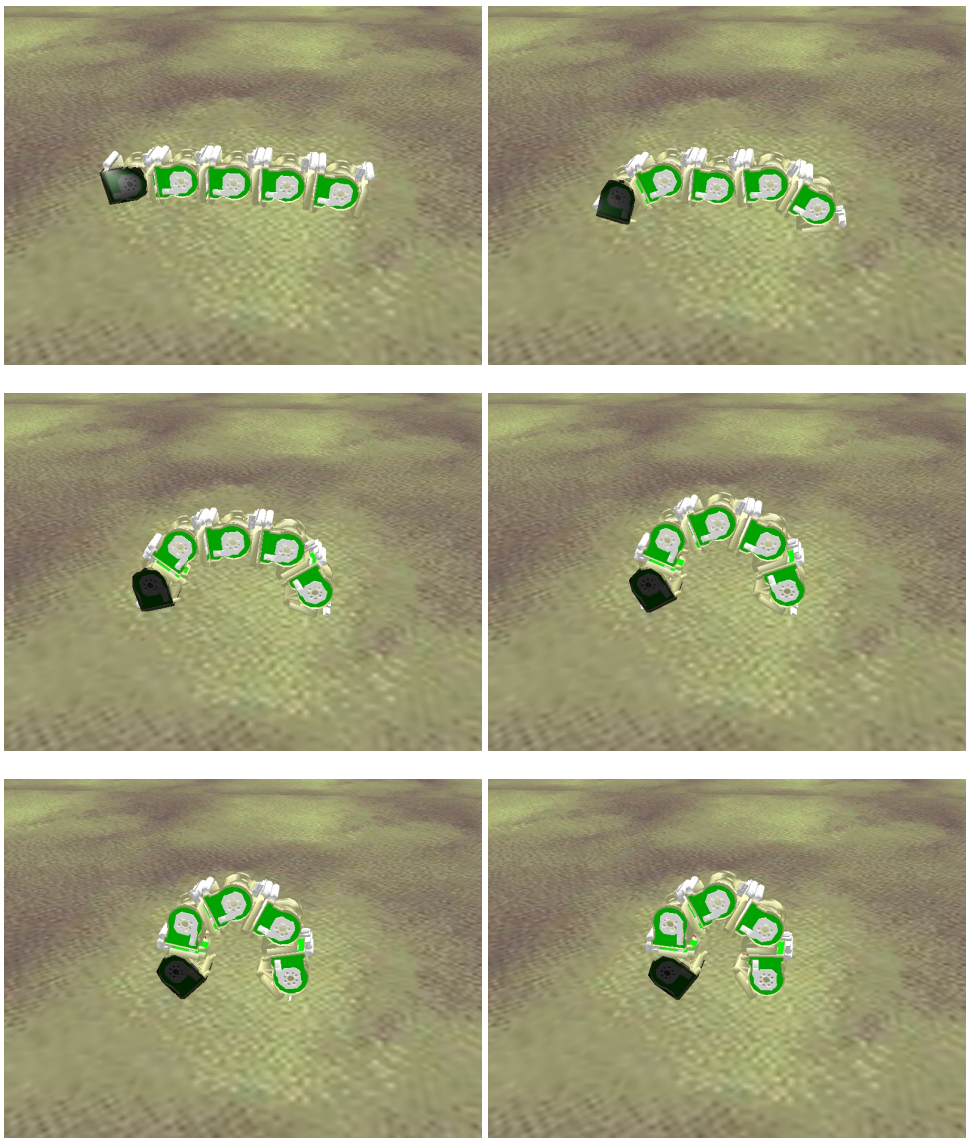


Figure 4.8: First Generation: Evolving Locomotion for a Snake-Shaped Organism.

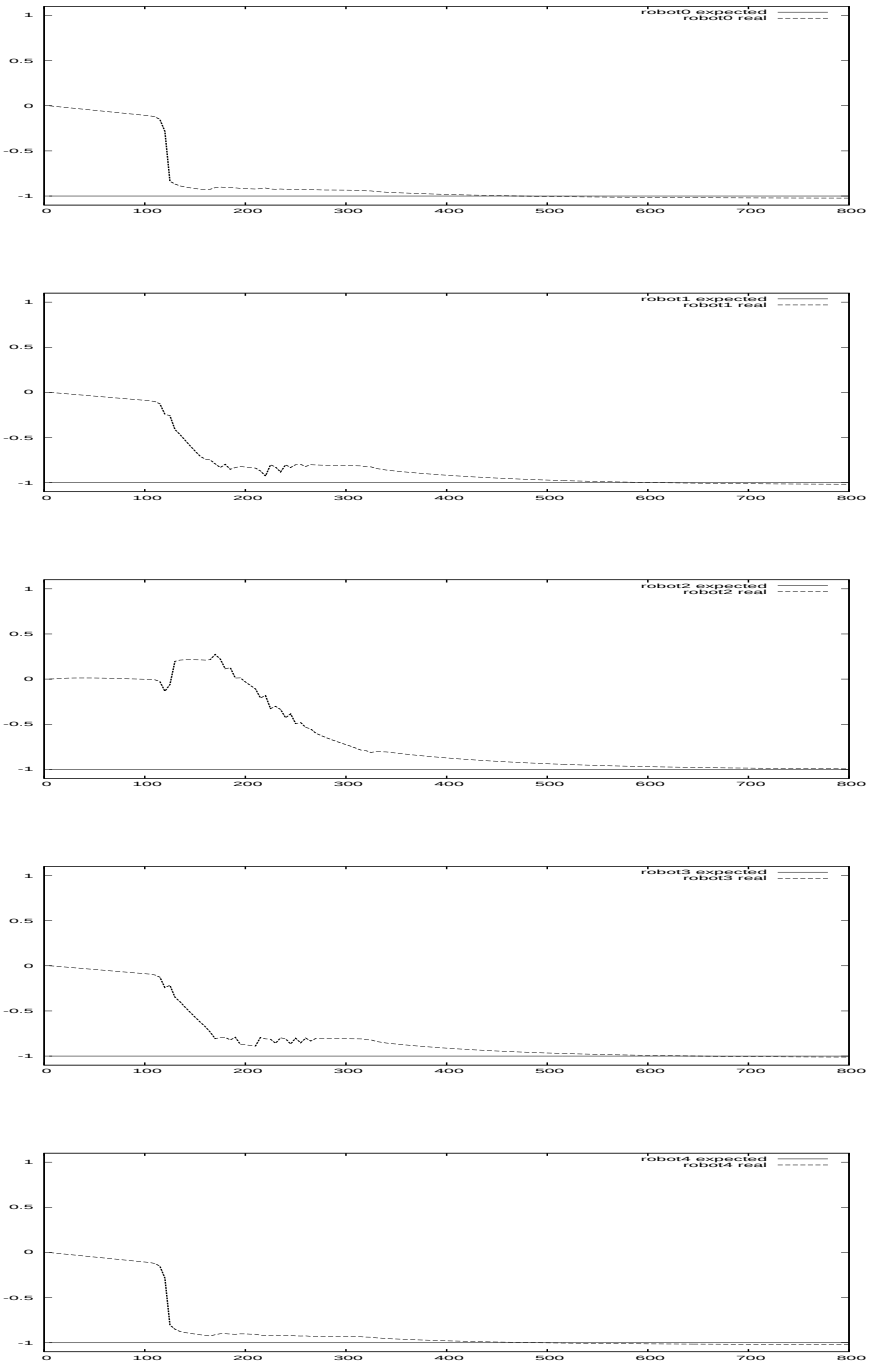


Figure 4.9: First Generation: Individual Actuator Positions of the Snake-Shaped Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

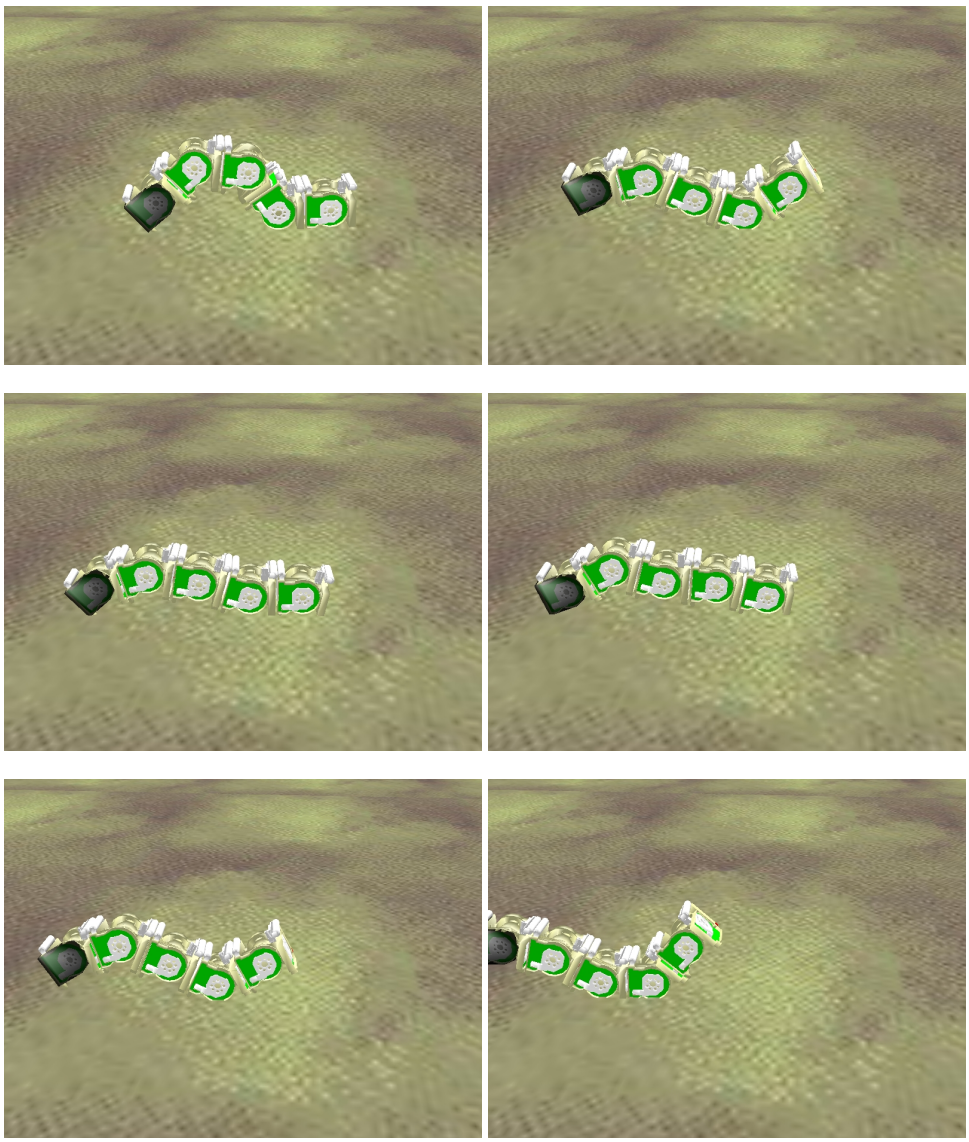


Figure 4.10: Second Generation: Evolving Locomotion for a Snake-Shaped Organism.

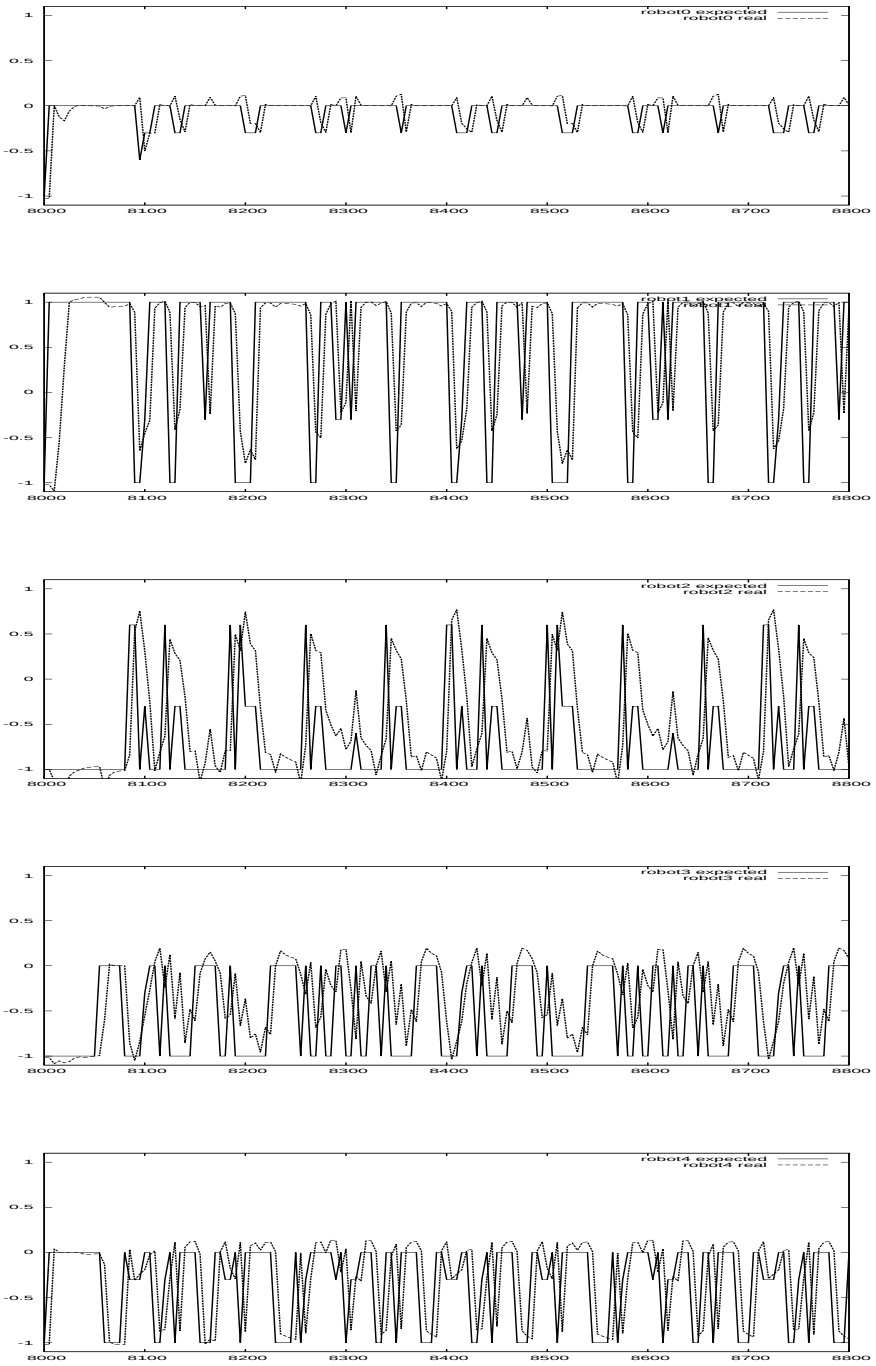


Figure 4.11: Second Generation: Individual Actuator Positions of the Snake-Shaped Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

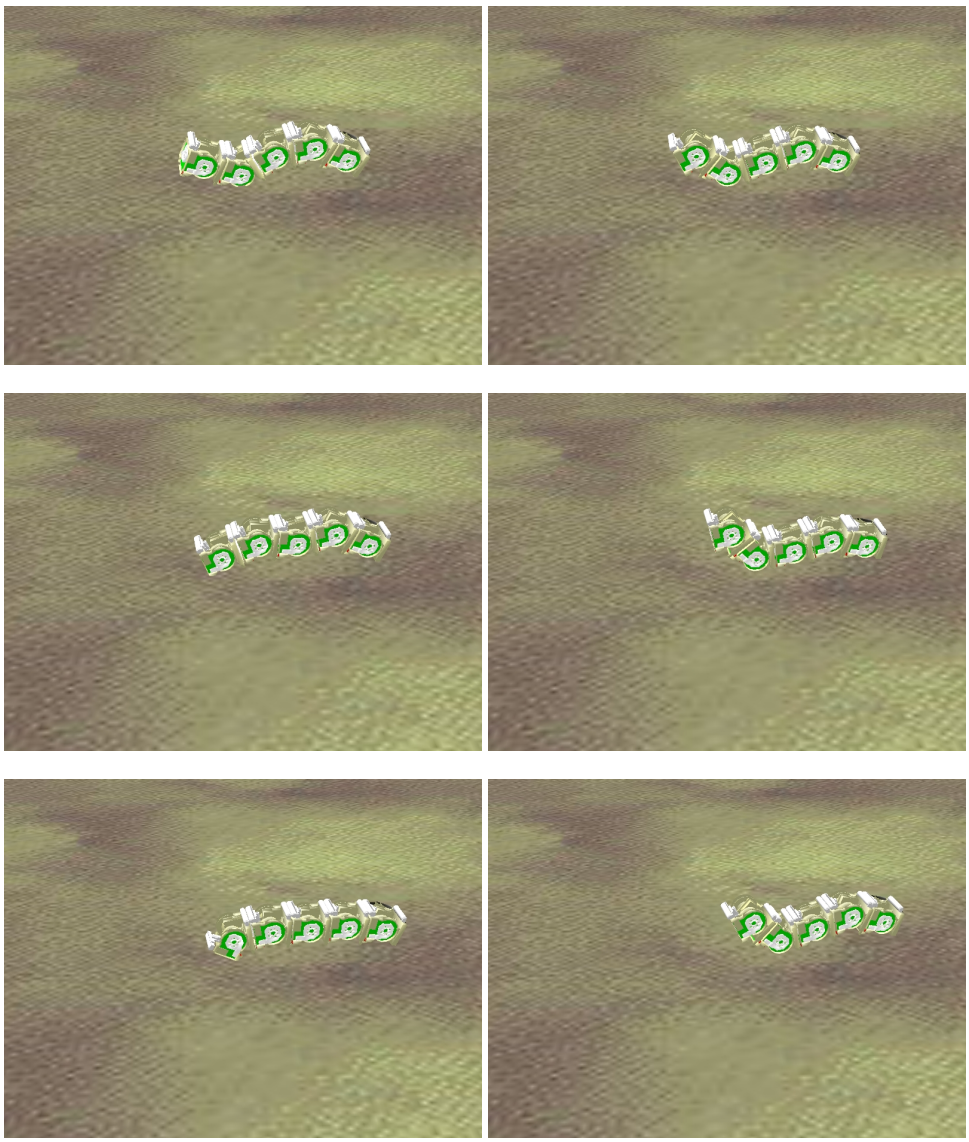


Figure 4.12: Fourth Generation: Evolving Locomotion for a Snake-Shaped Organism.

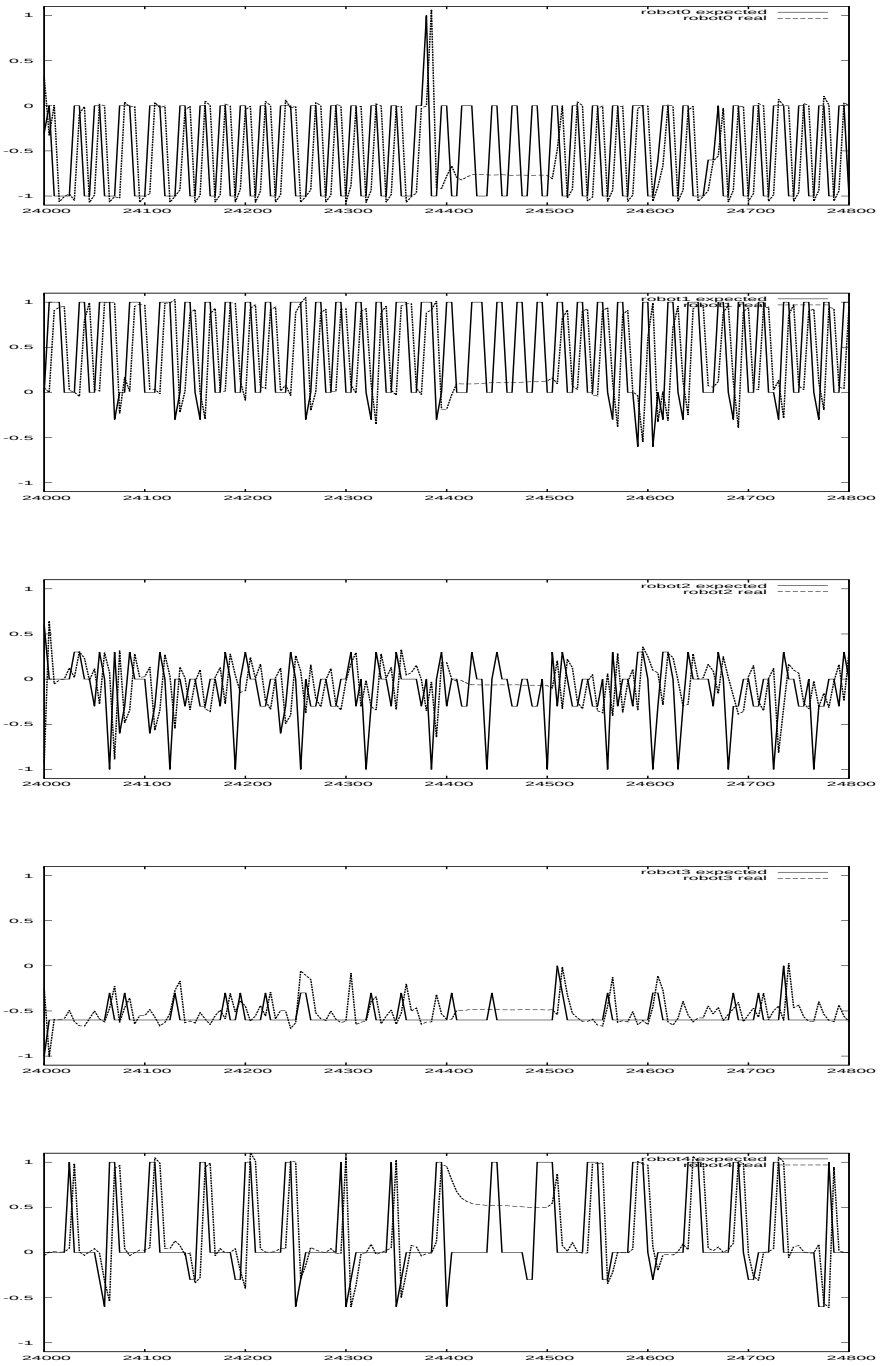


Figure 4.13: Fourth Generation: Individual Actuator Positions of the Snake-Shaped Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

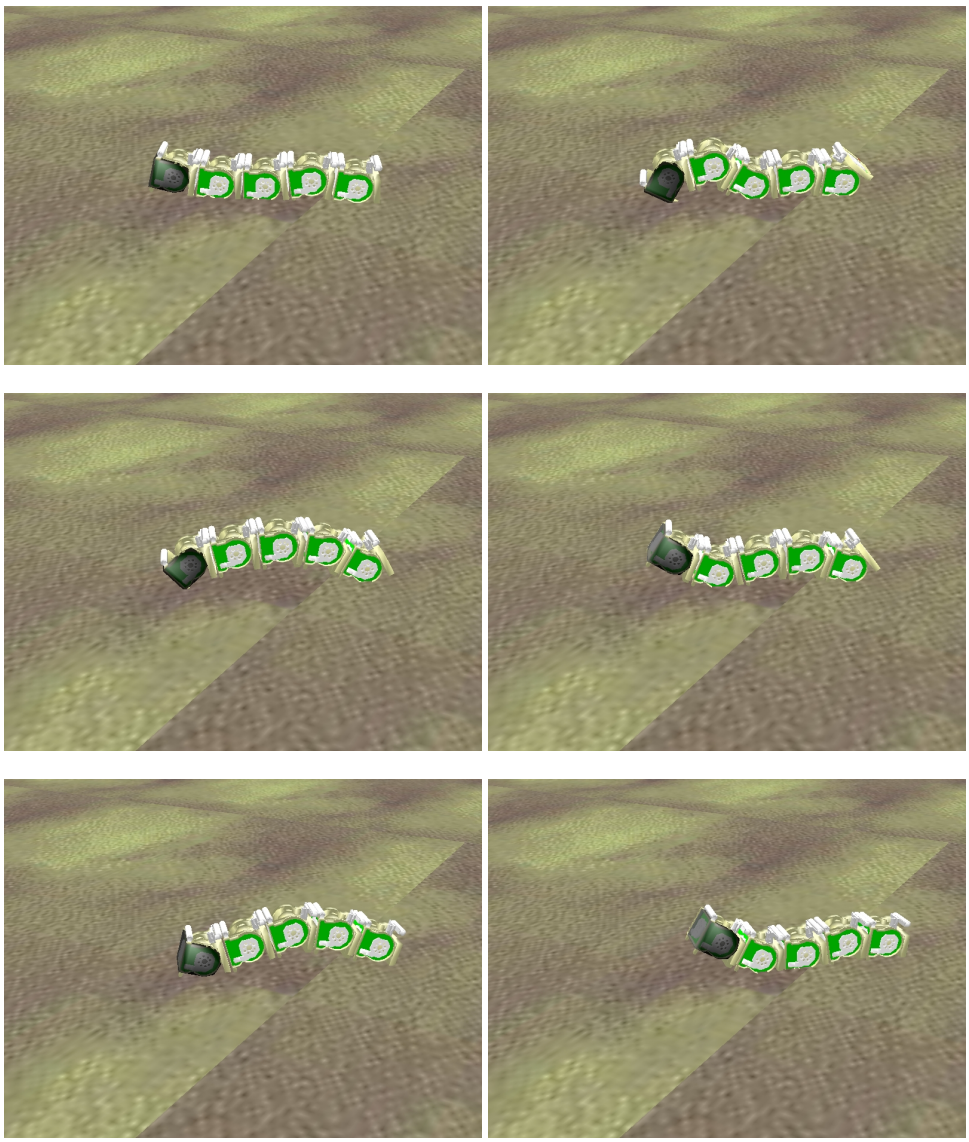


Figure 4.14: Sixth Generation: Evolving Locomotion for a Snake-Shaped Organism.

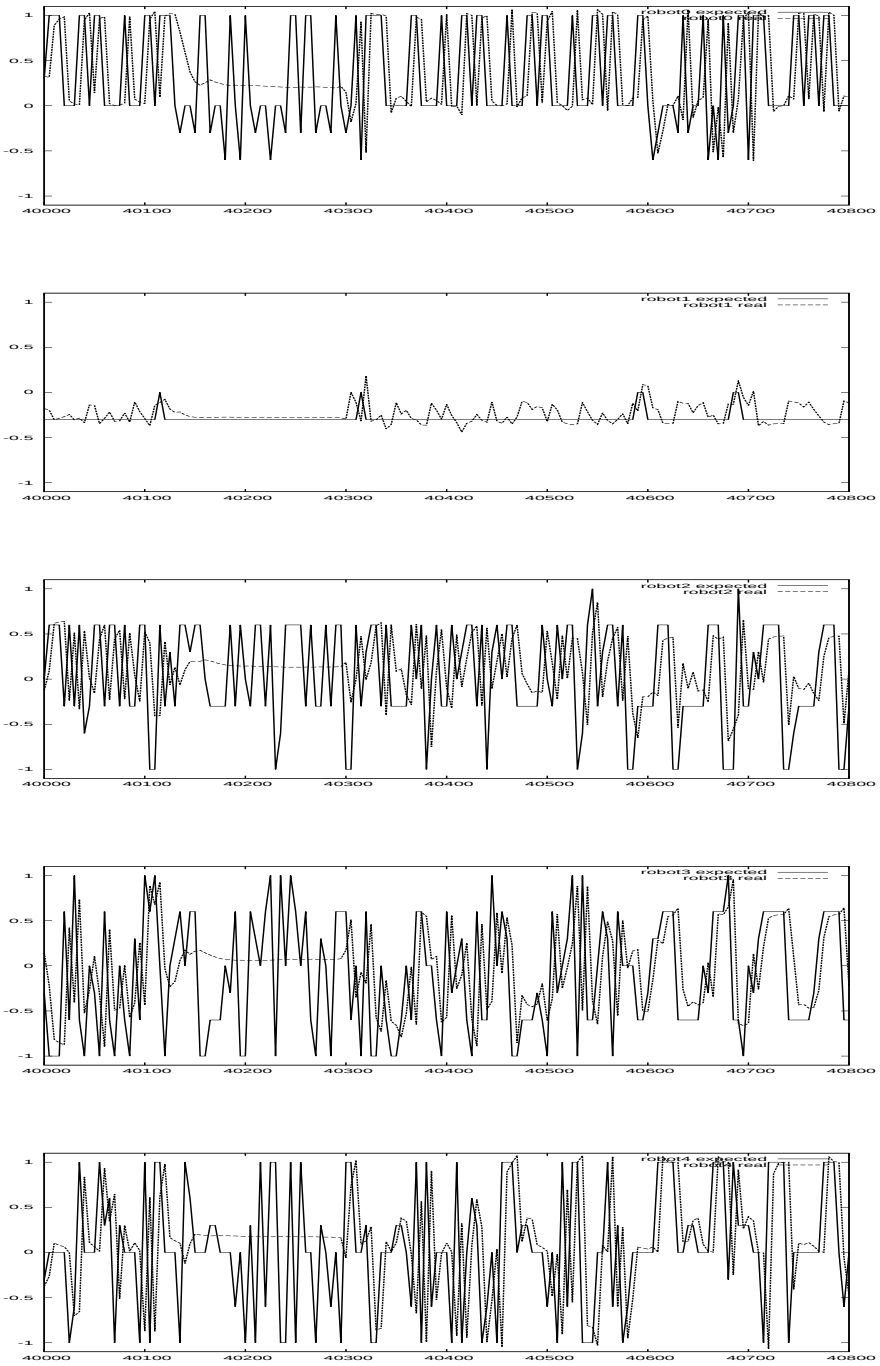


Figure 4.15: Sixth Generation: Individual Actuator Positions of the Snake-Shaped Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

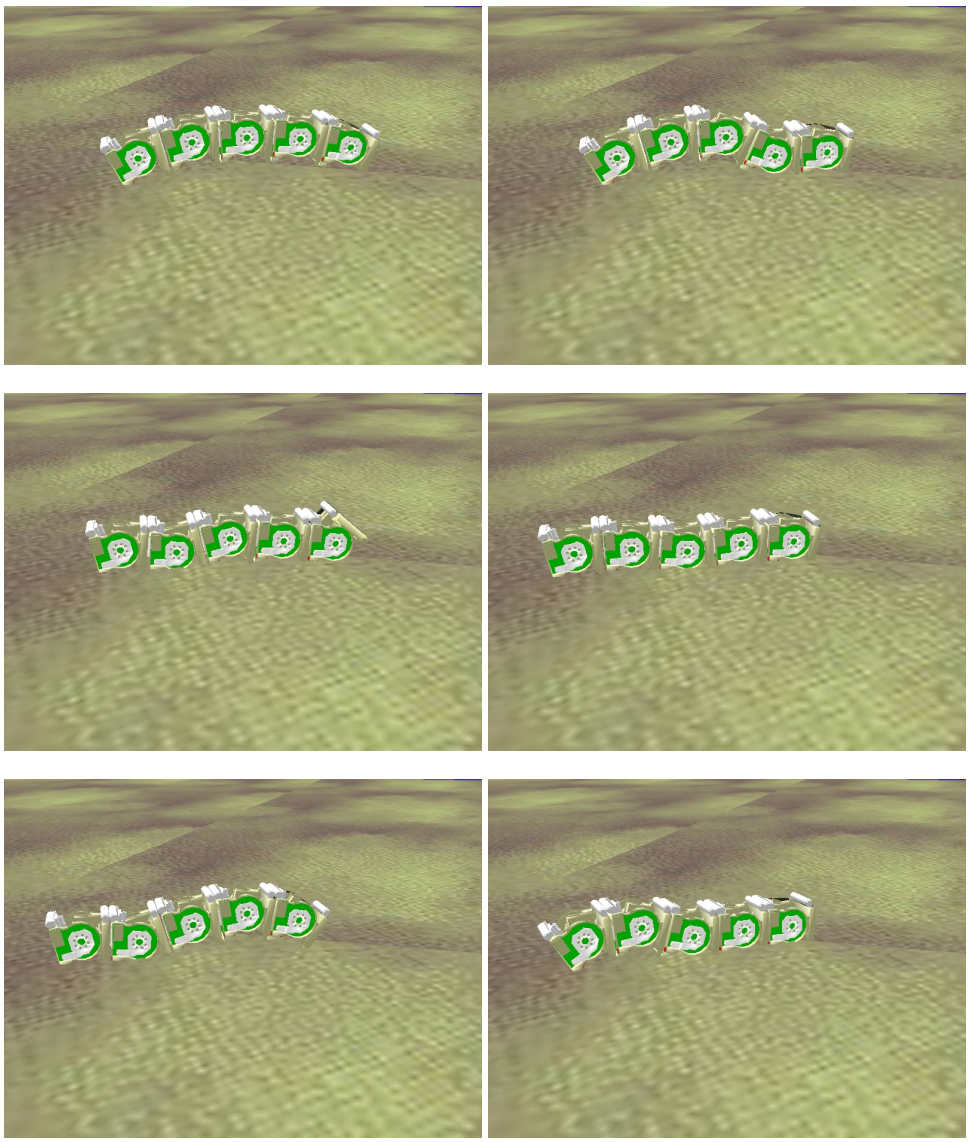


Figure 4.16: 29th Generation: Evolving Locomotion for a Snake-Shaped Organism.

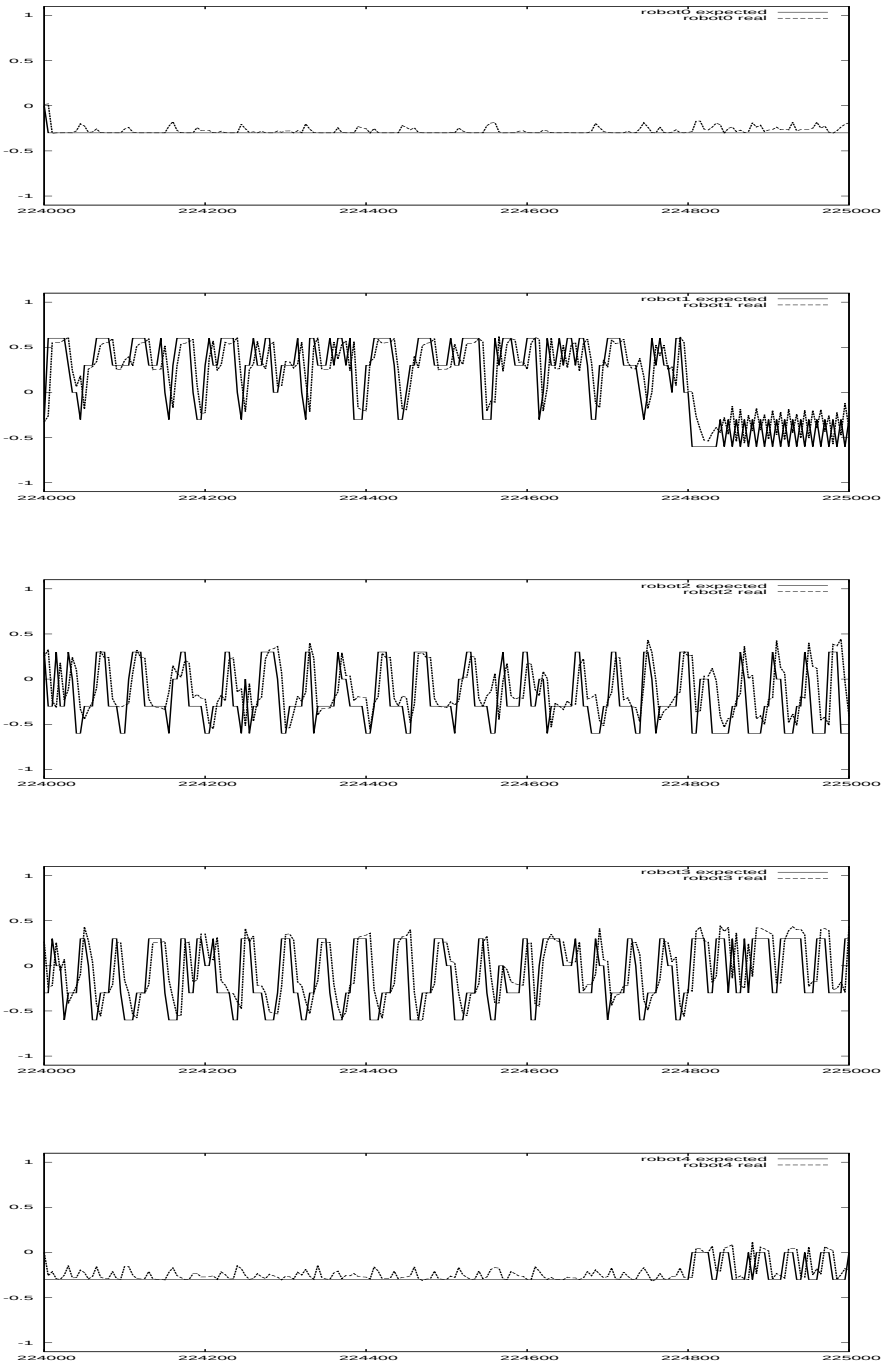


Figure 4.17: 29th Generation: Individual Actuator Positions of the Snake-Shaped Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

Summary

In summary it could be shown that the snake-shaped organism is able to move across the arena (Figure 4.18). The organism starts at point $[0, 0]$ and travels a longer distance during the evolutionary process. Directly at the beginning of the first run the organism covered a long and straight distance which indicates a good gait. Additionally, the evaluated motion pattern looks like a snake.

The good locomotion at the beginning can be identified in Figure 4.19, too. Already between the third and fourth generation the best fitness values were achieved.

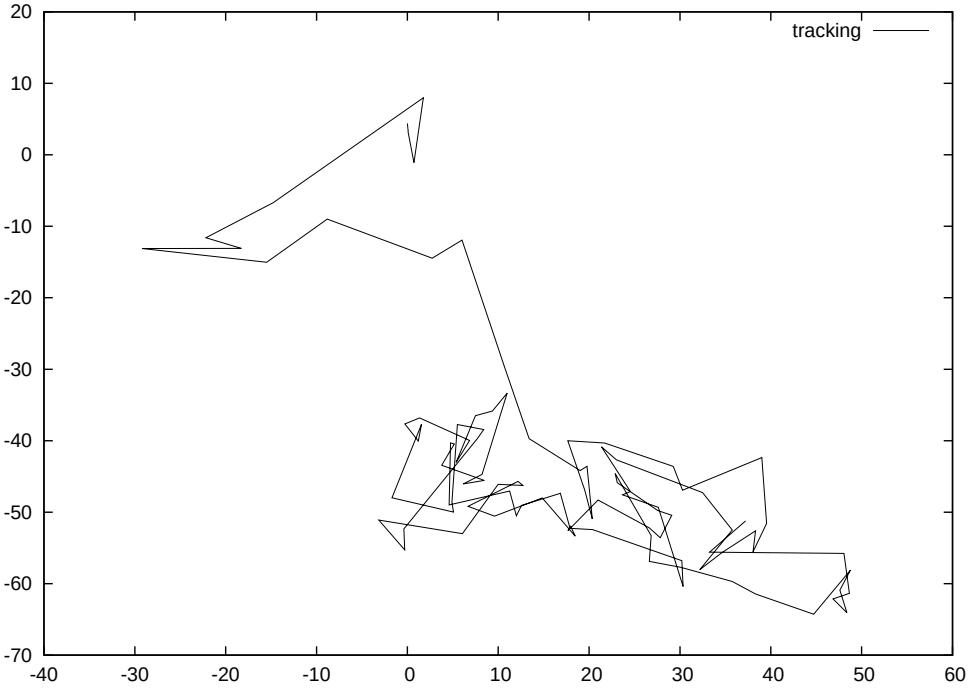


Figure 4.18: First tracking of the snake-shaped organism on the virtual arena.

X-axis: X direction [-∞, +∞].
Y-axis: Y direction [-∞, +∞].

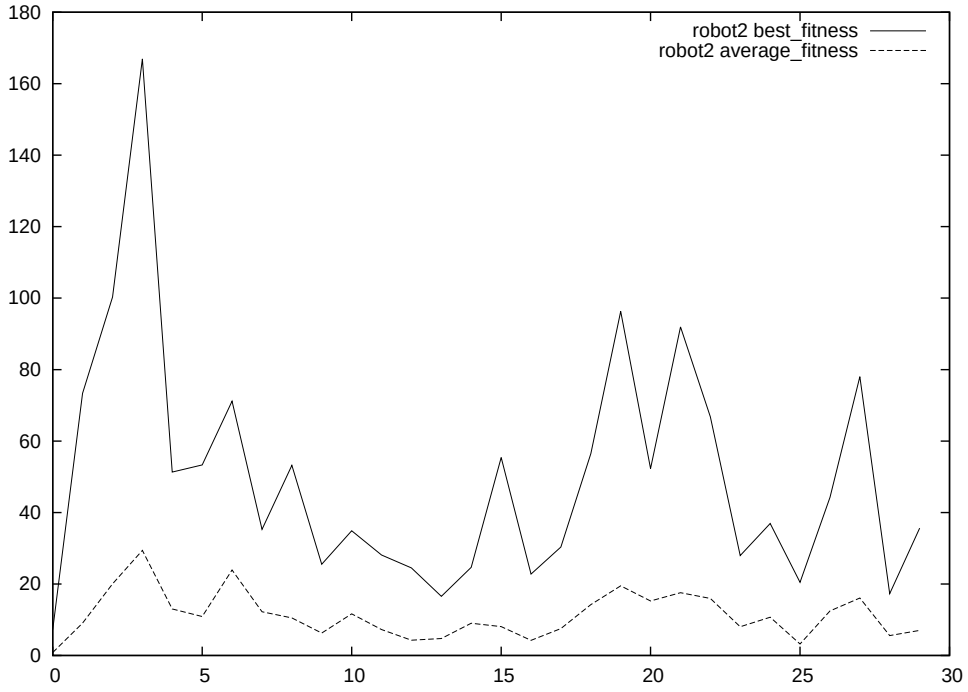


Figure 4.19: First fitness values for the snake-shaped organism for each generation.

X-axis: generation [0, +∞].
Y-axis: evaluated fitness value [0, +∞].

Second Run: Locomotion for a Snake-Shaped Organism

Because of the long duration of the evolutionary process this experiment could not repeated for 40 times as in the first two experiments of this thesis. For verification of the achieved result of the first run a second run with the same snake-shaped organism was performed.

Note that a second repetition with the same result can not be regarded as a general proof of the behavior.

Parameters

For the second experimental run for evolving locomotion the parameters in Table 4.5 were set. Notice that the parameters were not changed from the first run. This was done to exclude that a modification of these parameters do not lead to a variation of the result of the experiment.

Summary

Just as in the first pass the snake-shaped organism crawls partly straight across the arena floor. The tracking data in Figure 4.20 shows its movement behavior. In the second run it was also possible to evaluate motion patterns which leads to locomotion of the organism.

It should also be mentioned that the interesting parts of the tracking data are the straight walking lines. These are the regions where the individual robots respectively their motion patterns generated by the artificial neural networks on each robot interact functionally and form adequate locomotion.

An uncontrolled spinning in circles is caused by a combination of behavioral patterns of different artificial neural networks which are distributed on the robots. Over the time these artificial neural networks have evolved in different evolutionary directions. A fact that might be happened during the entire evolutionary process. However, such combinations will be extinguished by the implemented selection mechanisms. This fact is also the explanation why the fitness values will not increase continuously and that are the dues which have to be paid for the distributed approach of this thesis.

Figure 4.21 shows the just contemplated history of the evaluated fitness values of each generation. Generally speaking the diagram looks similar to that of the first run. Conspicuously and inexplicably is the halt of the whole organism in 23rd generation. Perhaps an evolutionary accident.

Parameter	Value	Description
Evolutionary Algorithm Type	SNN	defines the type of network
Using Phenotype Mapping	1	activates phenotype mapping, note CGE can operate directly on genome
Population Size of Island	10	size of the start population of an island
Number of Input Neurons	5	number of input neurons
Number of Output Neurons	3	number of output neurons
Number of Initial Mutations	5	number of mutations which should be performed to a random start genome
Number of Steps for Evaluation	800	number of steps a genome will be evaluated before switching to the next one
Number Of Generations	undefined	total number of generations
Number of Exploitation Steps	10	number of exploitation steps to perform before a exploration step follows
Default Learning Rate	0.3	Default learning rate for neuron if genome is generated randomly
Default Mutation Probability	0.3	Default mutation probability for a neuron if genome is generated randomly
Fitness Function Type	STRAIGHTON	defines the type of fitness function to use
SelectionMode	ELITISM	defines the type of selection mechanism
Selection Parameter	20.0	parameter to pass to the selection function
EnableParametricMutation	ON	enables parametric mutation
EnableStructuralMutation	ON	enables structural mutation
NewNeuronToLinkRatio	0.5	0.4 means during structural mutation, 40% probability for new neurons and 60% probability of a new link

Table 4.5: Second Run: Parameters for Evolving Locomotion with a Snake-Shaped Organism.

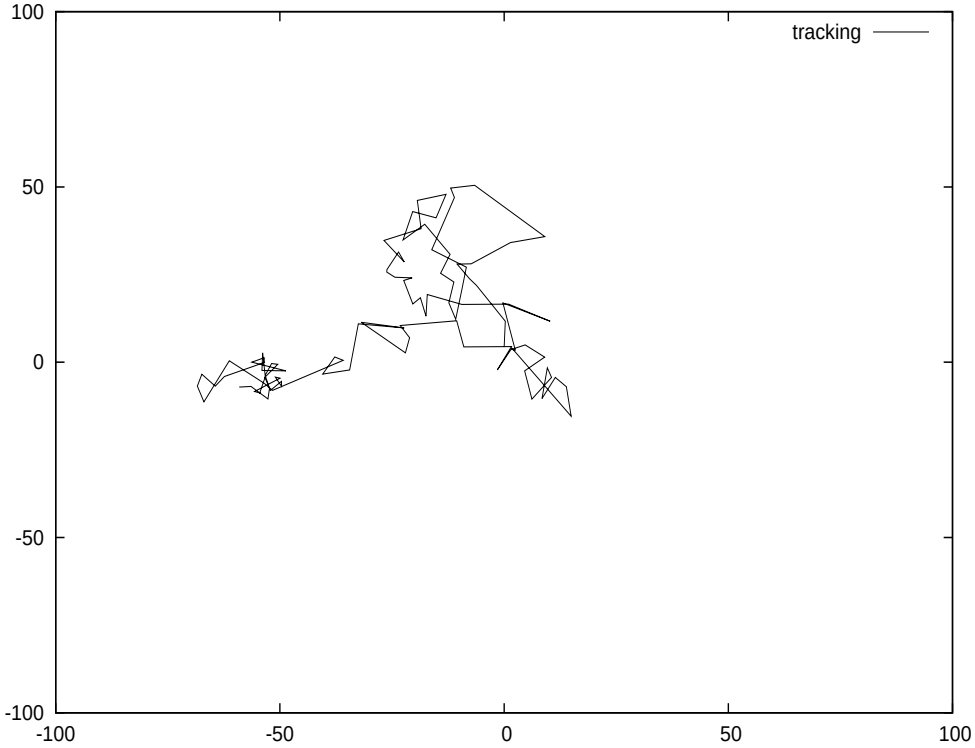


Figure 4.20: Second tracking of the snake-shaped organism on the virtual arena.

X-axis: X direction [-∞, +∞].
Y-axis: Y direction [-∞, +∞].

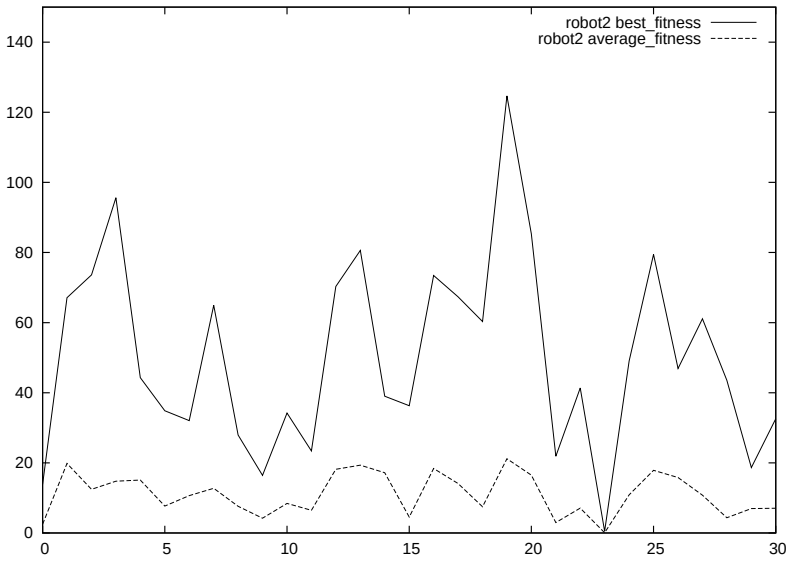


Figure 4.21: Second fitness values for the snake-shaped organism for each generation.

X-axis: generation [0, +∞].
Y-axis: evaluated fitness value [0, +∞].

Example of an evaluated Artificial Neural Network

For demonstration purpose in Figure 4.22 a generated ANN is plotted. The illustrated artificial neural network originates from the robot which is marked black in the figures and has been generated during the second generation.

The network consists of the five input neurons "I" marked as I0, I1, I2, I3 and I4. Additional there are the three output spiking neurons SN0, SN1 and SN2. The "SN" output neurons are identifiable with their black border. Neuron SN3 is a special spiking neuron that was added by a structural mutation. It was not included right from the start but was inserted only during the evolutionary process.

The numbers at the connecting lines are the corresponding weights of this edge. The weights can also be negative which leads to an inhibition of the subsequent neuron. Because of that reason there were no distinct between activation or inhibitory spiking neurons in the implementation needed.

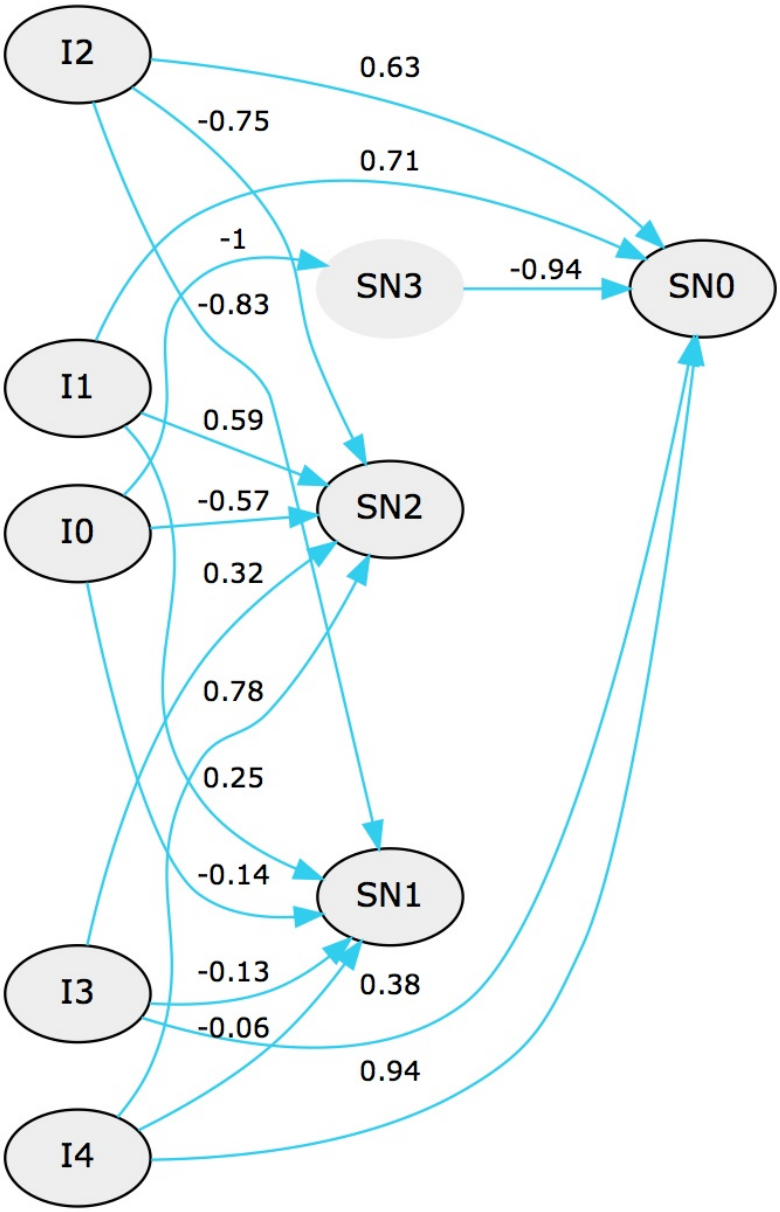


Figure 4.22: Example - Evolved Neural Network.

4.3.4 Results of the Quadruped Organism

With the quadruped organism pictured in Figure 4.23 were also two documented runs performed and analyzed in the following section.

The used robot organism consists of seven individual autonomous robots from the University of Karlsruhe. Each robot is fixed connected to its neighbor with the help of specific docking elements.

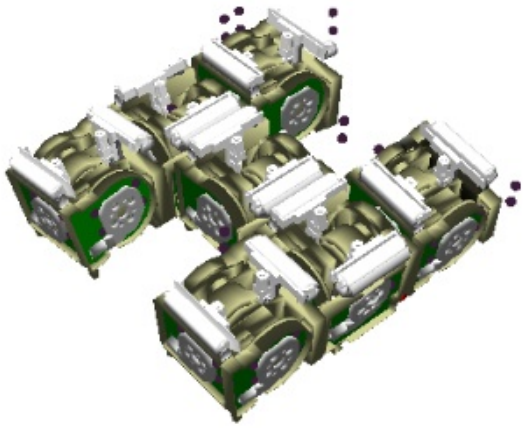


Figure 4.23: A quadruped robot organism consisting of seven individual autonomous robots from the University of Karlsruhe. This kind of robot is one of the three available robot platforms developed during the Symbrion project. The remaining platforms can be seen in Figure 4.7.

2		6
1	4	5
3		7

Table 4.6: The distributed identification numbers for each robot of the organism.

First Run: Locomotion for a Quadruped Organism

For sake of completeness, each run the list of parameters were added because only little changes would lead to different results. The results were not reproducible without this parameter listing.

Parameters

For the first experimental run for evolving locomotion the parameters in Table 4.7 were set.

Most of the parameters are chosen as in the previous experiment with the same arguments as already mentioned in Chapter 4.3.3. Contrary to the first experiment the evaluation steps were increased to 1000 time steps in order to give the individuals more time to prove themselves.

Because it is not clarified how the best ratio between structural and parametric mutation is in this experiment an attempt was made with only five exploitation steps. With this amendment every fifth generation a structural mutation is performed instead of a parametric one. What kind of consequences that entails is unknown.

Discussion

As in the experiment 4.3.3 the quadruped organisms lies flat on the arena floor initially. This organism is curling at the beginning because of the already-described reasons. Descriptive images of the behavior in the first generation are shown in Figure 4.24 and the corresponding joint angles in Figure 4.25 and 4.26. In the subsequent pictures the rest of the documented generations can be seen.

The documented diagrams are sorted by robot identification number. Adjacent diagrams are not necessarily associated with adjacent robots in the organism. The topology can be seen in Figure 4.6.

Examining the second generation (4.27) the organism already performs rhythmic movements. A fact which will be shown on the basis of the recorded angular positions of the joints. For example the backbone robots with the identification one and four perform synchronous movements in opposite direction looking at the significant peaks.

Comparing the feet of the organism a synchronous movement of the front feet which affects the robot two and robot three is determined. The movement corresponds to a forward pulling and can be seen right in the middle of the Figure 4.27. Precisely observing robot three with robot six indicates an synchronous behavior. These robots are placed diagonally opposite to each other. A movement that looks similar to that of a salamander, see Figure 2.3.

From generation to generation the movement will be further developed and char-

Parameter	Value	Description
Evolutionary Algorithm Type	SNN	defines the type of network
Using Phenotype Mapping	1	activates phenotype mapping, note CGE can operate directly on genome
Population Size of Island	10	size of the start population of an island
Number of Input Neurons	5	number of input neurons
Number of Output Neurons	3	number of output neurons
Number of Initial Mutations	5	number of mutations which should be performed to a random start genome
Number of Steps for Evaluation	1000	number of steps a genome will be evaluated before switching to the next one
Number Of Generations	undefined	total number of generations
Number of Exploitation Steps	5	number of exploitation steps to perform before a exploration step follows
Default Learning Rate	0.3	Default learning rate for neuron if genome is generated randomly
Default Mutation Probability	0.3	Default mutation probability for a neuron if genome is generated randomly
Fitness Function Type	STRAIGHTON	defines the type of fitness function to use
SelectionMode	ELITISM	defines the type of selection mechanism
Selection Parameter	20.0	parameter to pass to the selection function
EnableParametricMutation	ON	enables parametric mutation
EnableStructuralMutation	ON	enables structural mutation
NewNeuronToLinkRatio	0.5	0.4 means during structural mutation, 40% probability for new neurons and 60% probability of a new link

Table 4.7: First Run: Parameters for Evolving Locomotion with a Quadruped Organism.

acterized. Thus in the tenth generation the following behavior can be described: Now the backbone robots one and four are still synchronous but the amplitude of robot one is smaller than before. The front feet are still synchronous and try to pull the organisms. However, the rear feet work asynchronous and perform their movements out of phase. In contrast to previous generations the arranged diagonally robots three and six are now asynchronous.

Summary

In summary it could be shown that the quadruped organism is able to move across the arena (Figure 4.39). The organism starts at point [0,0] as in the experiments before. It moved a long distance through the virtual arena with a lot of straight runs. As already known these straight drives are an indicator of good movement.

This good movements can also be seen in the fitness chart, see Table 4.40. The fitness values of the best organism (solid line) and the average (dashed line) are shown for all individuals for all 30 generations. The recorded fitness values are fluctuating but in average they are quite large.

A good result for locomotion of a quadruped organism because this is not a simple movement. It is very difficult to move four legs on such way that an adequate action results.

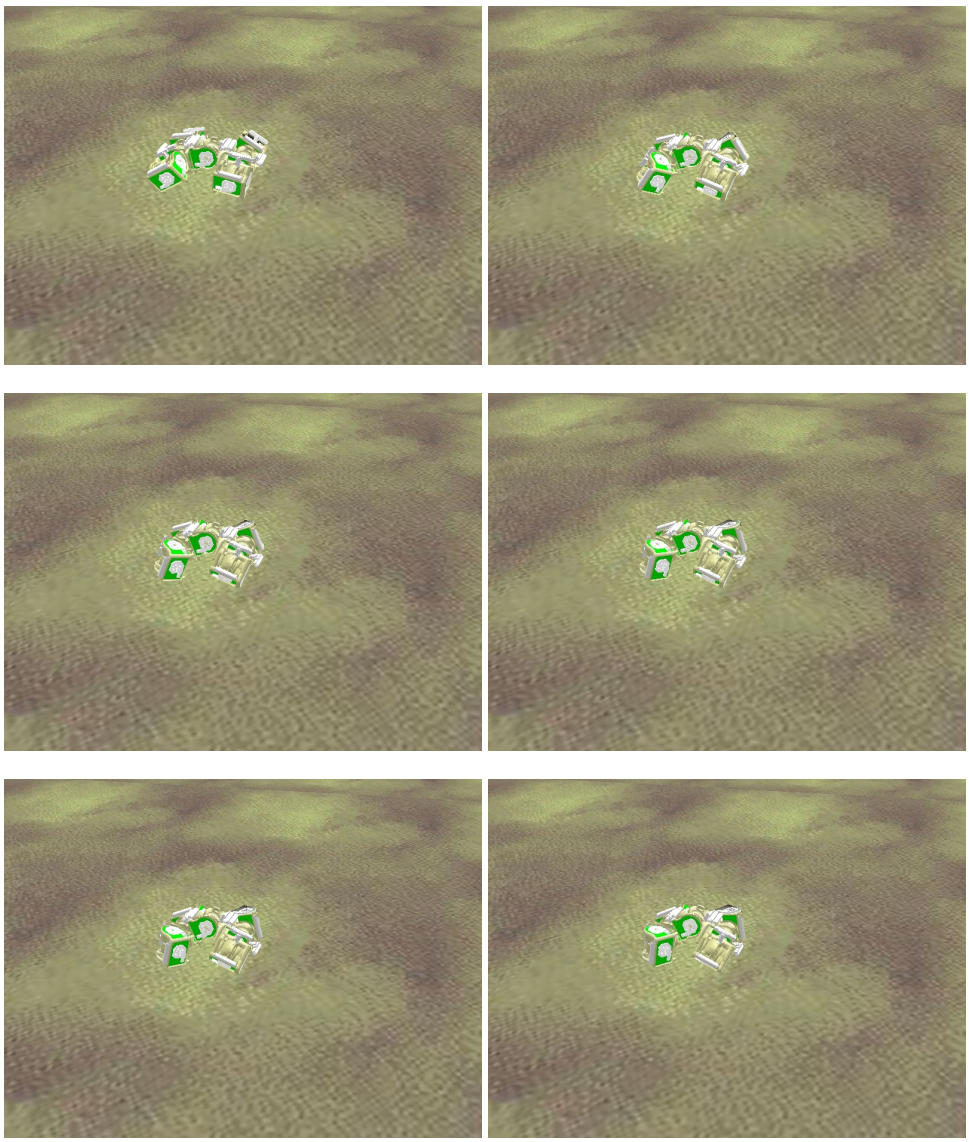


Figure 4.24: First Generation: Evolving Locomotion for a Quadruped Organism.

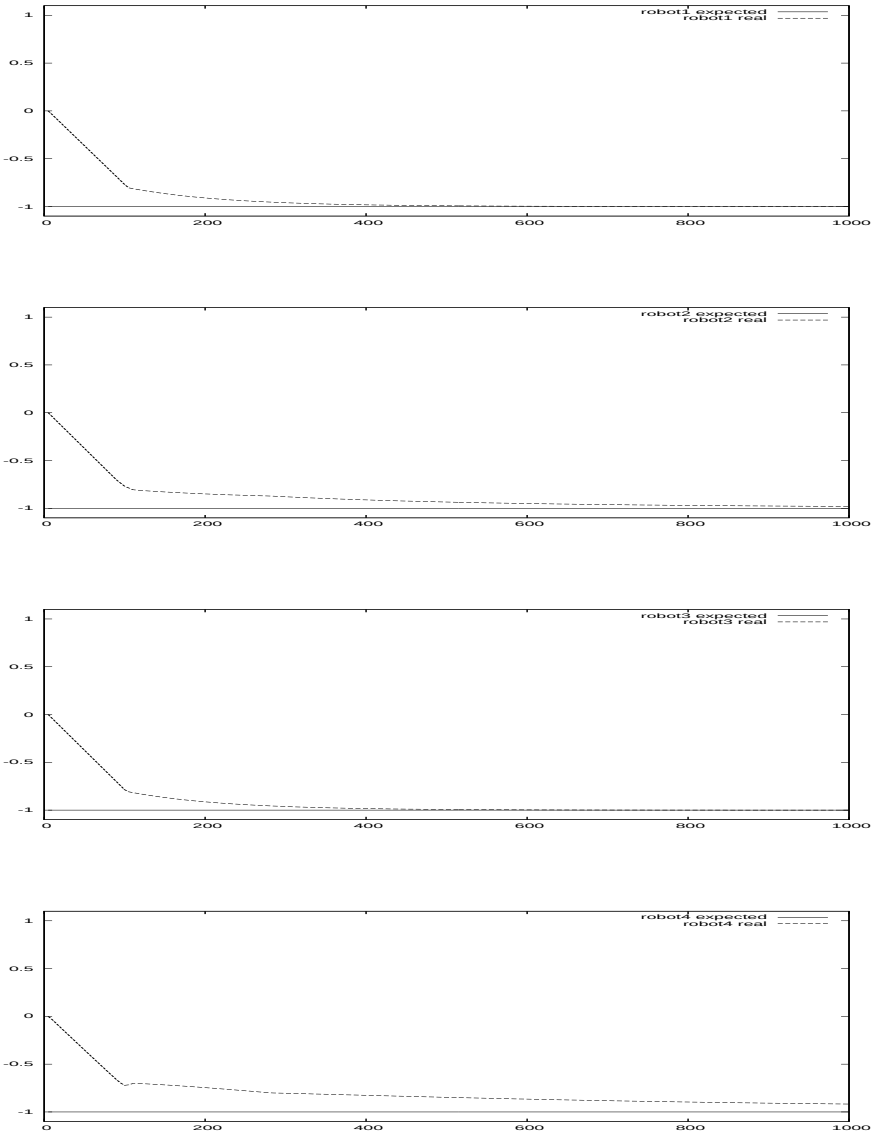


Figure 4.25: First Generation: Individual Actuator Positions 1 - 4 of the Quadruped Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

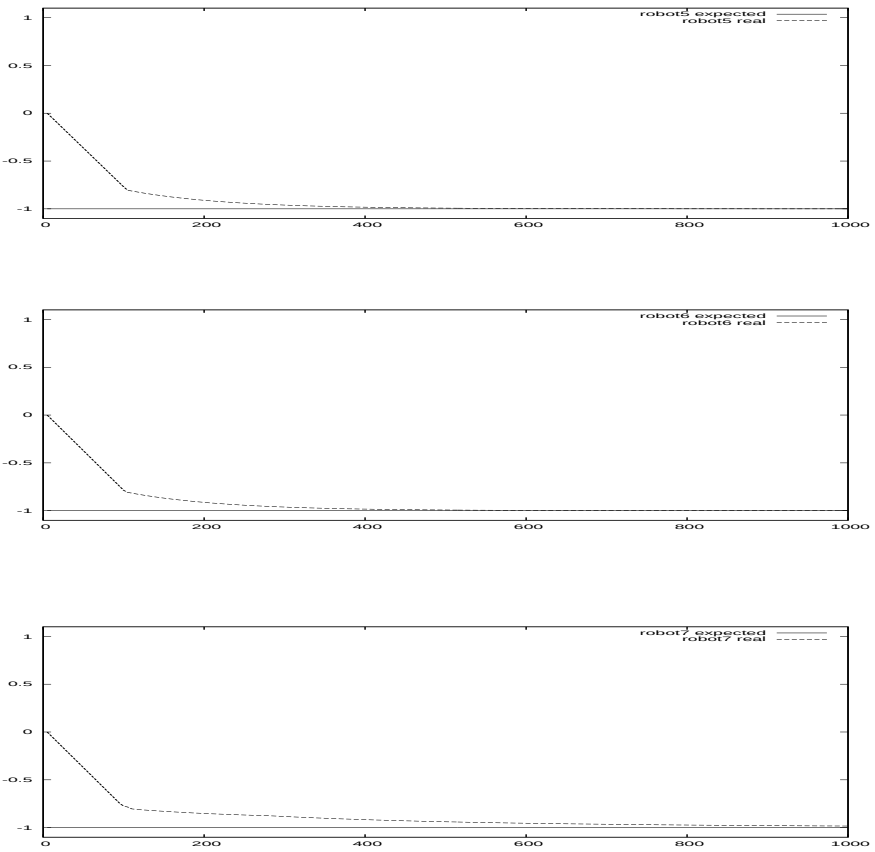


Figure 4.26: First Generation: Individual Actuator Positions 5 - 7 of the Quadruped Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

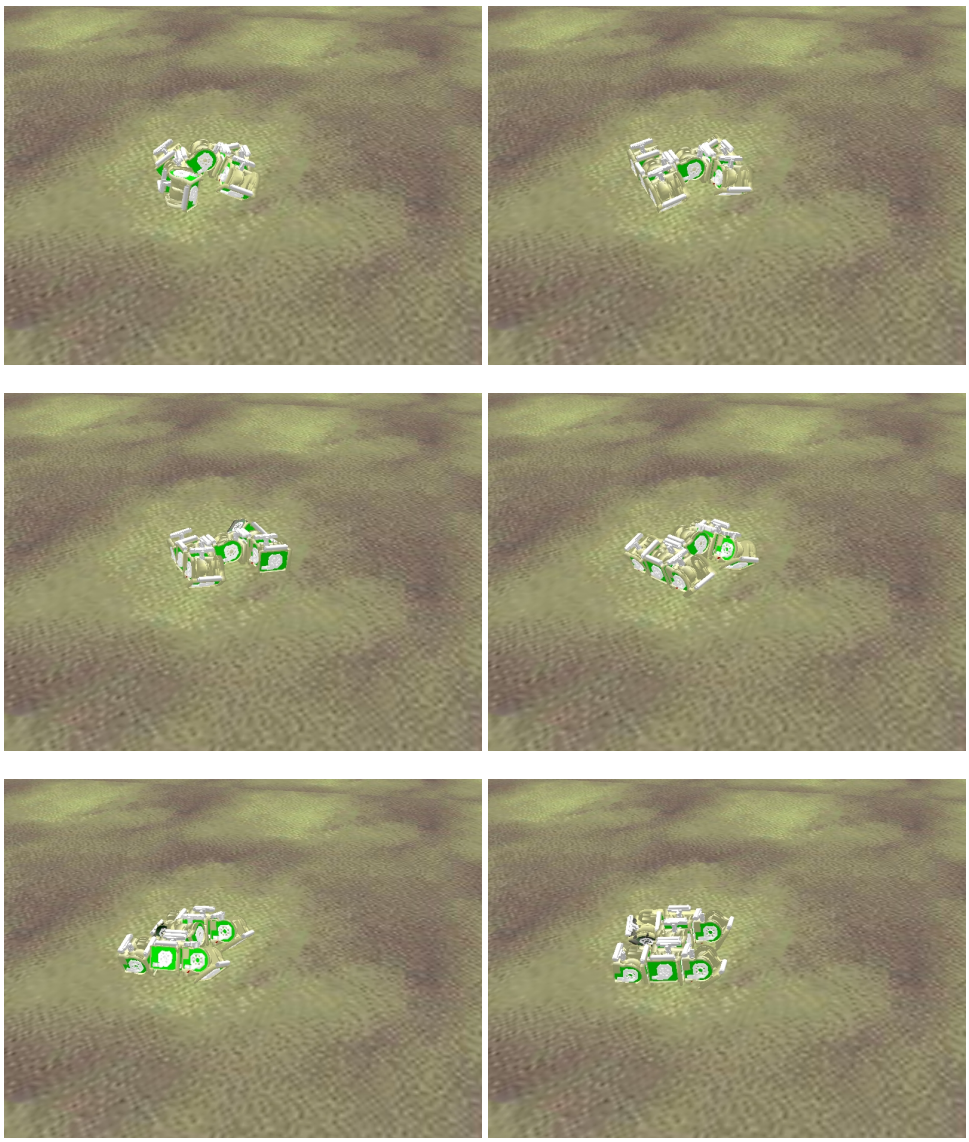


Figure 4.27: Second Generation: Evolving Locomotion for a Quadruped Organism.

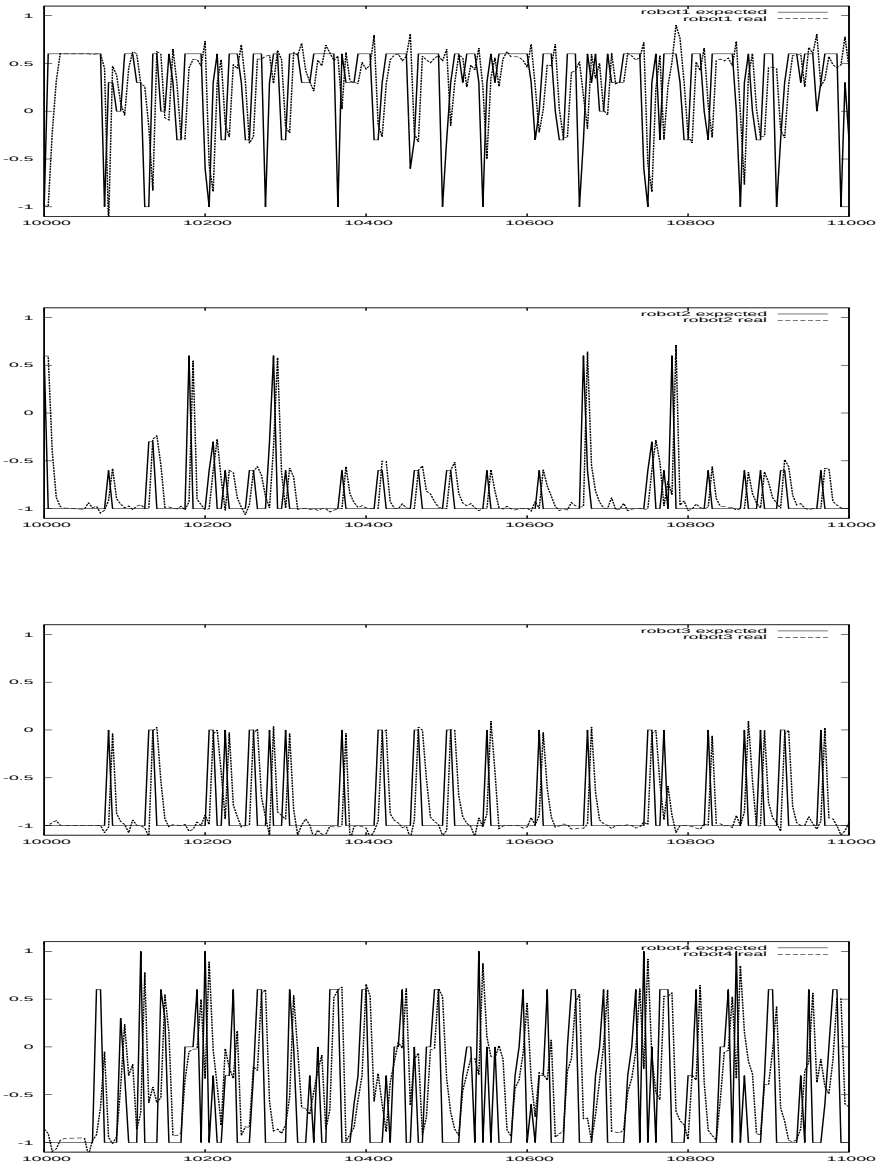


Figure 4.28: Second Generation: Individual Actuator Positions 1 - 4 of the Quadruped Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

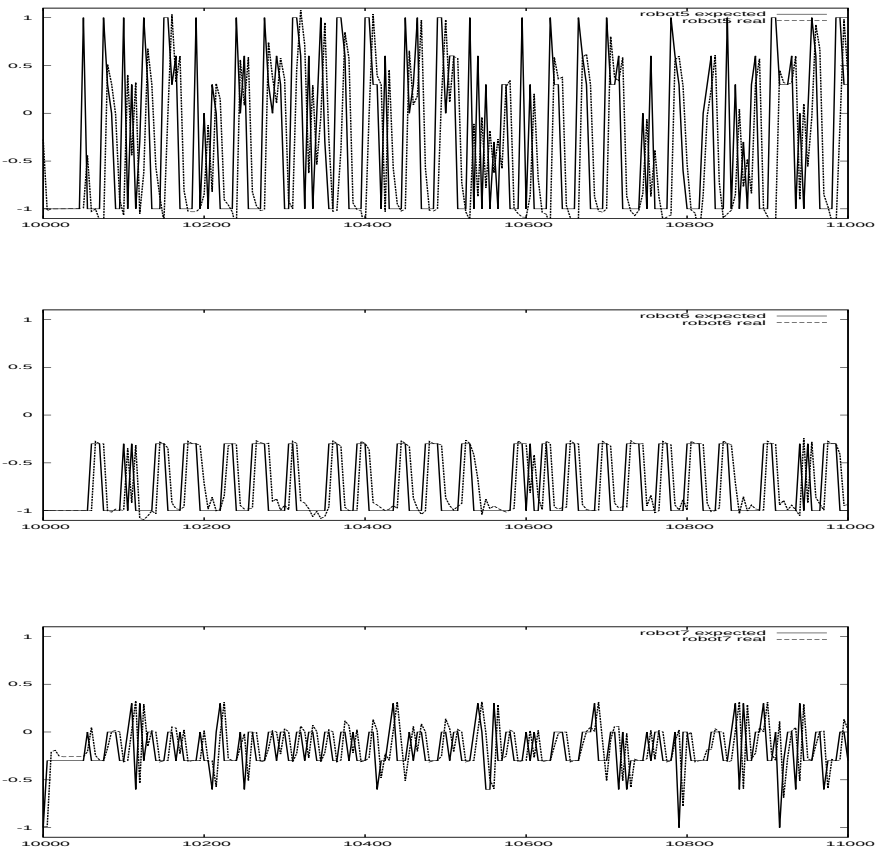


Figure 4.29: Second Generation: Individual Actuator Positions 5 - 7 of the Quadruped Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

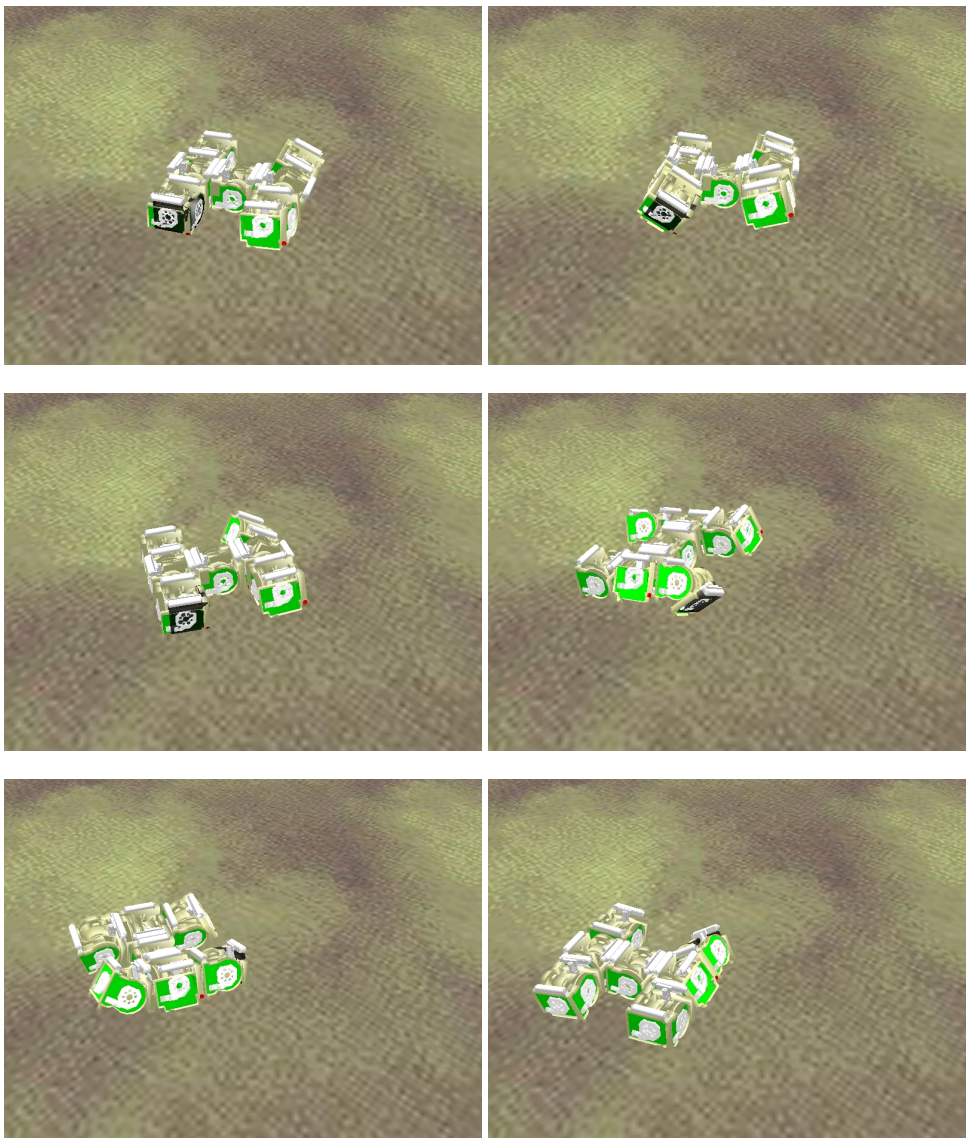


Figure 4.30: Fourth Generation: Evolving Locomotion for a Quadruped Organism.

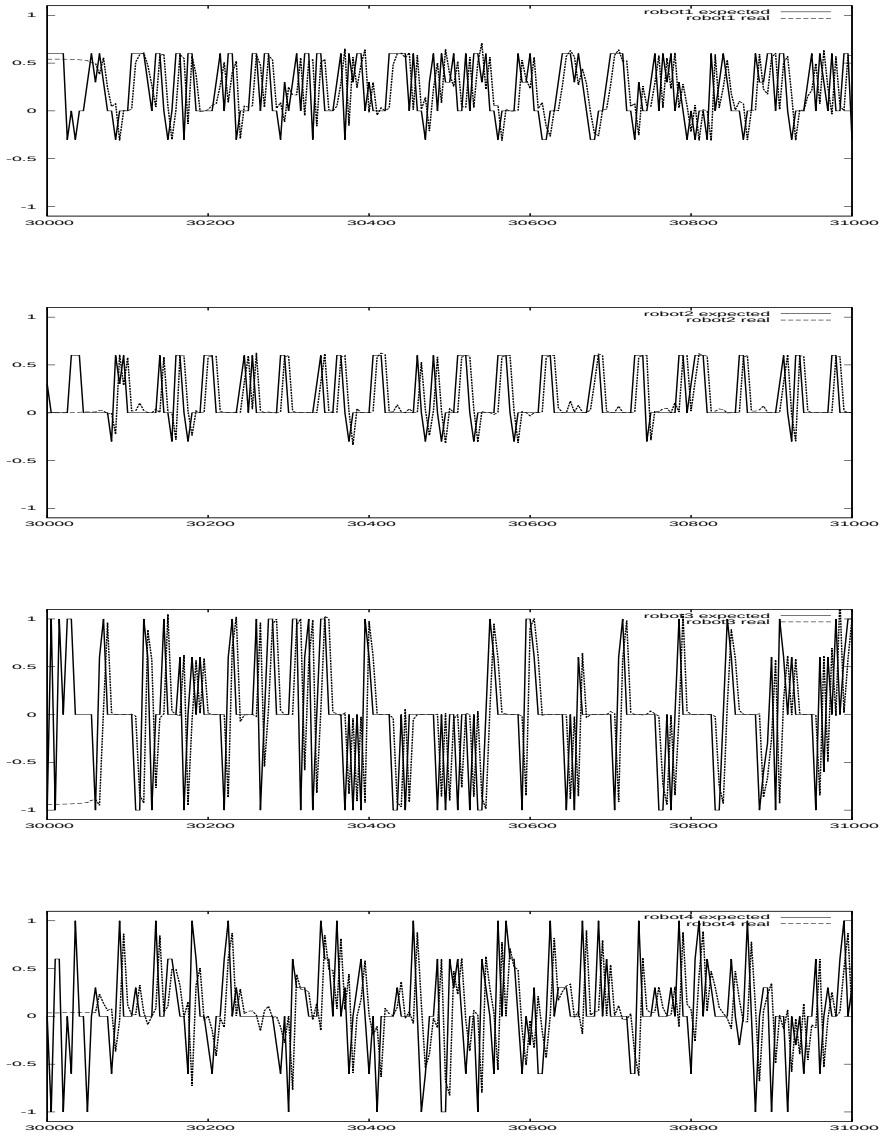


Figure 4.31: Fourth Generation: Individual Actuator Positions 1 - 4 of the Quadruped Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

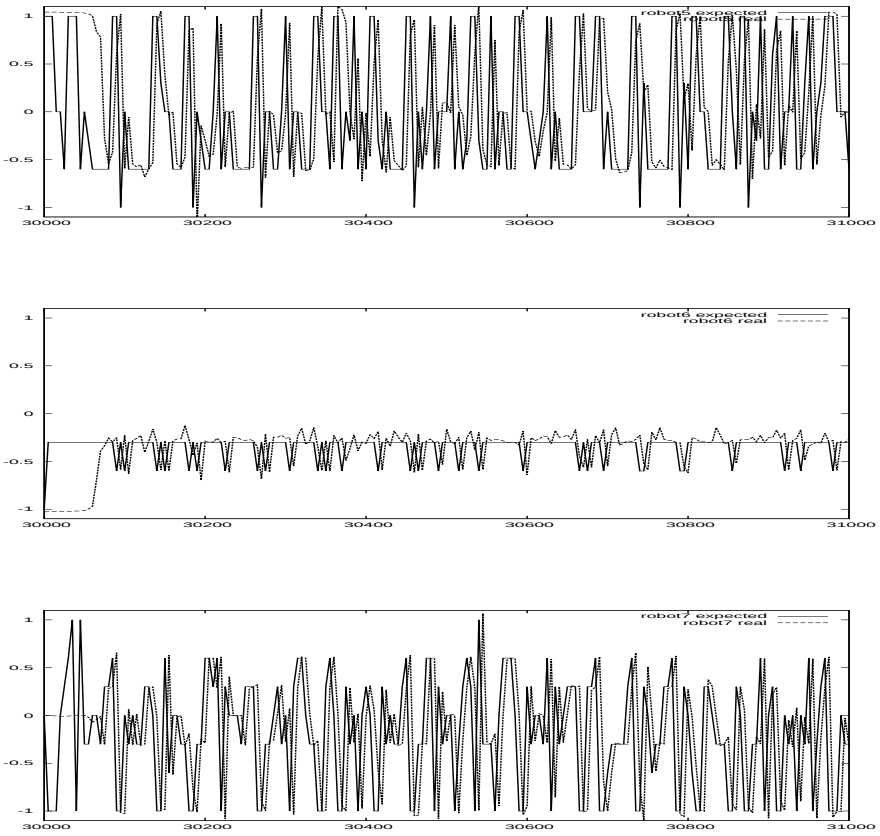


Figure 4.32: Fourth Generation: Individual Actuator Positions 5 - 7 of the Quadruped Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

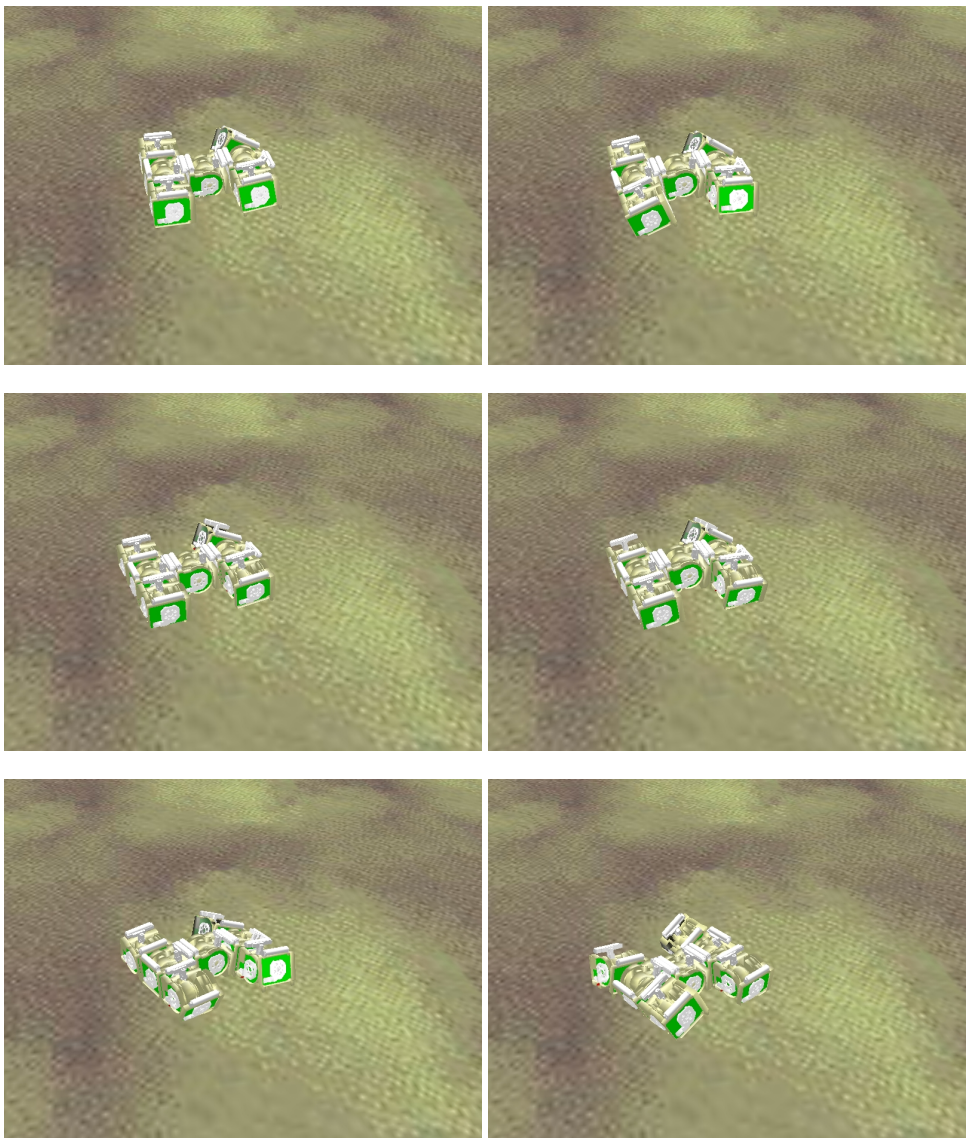


Figure 4.33: Sixth Generation: Evolving Locomotion for a Quadruped Organism.

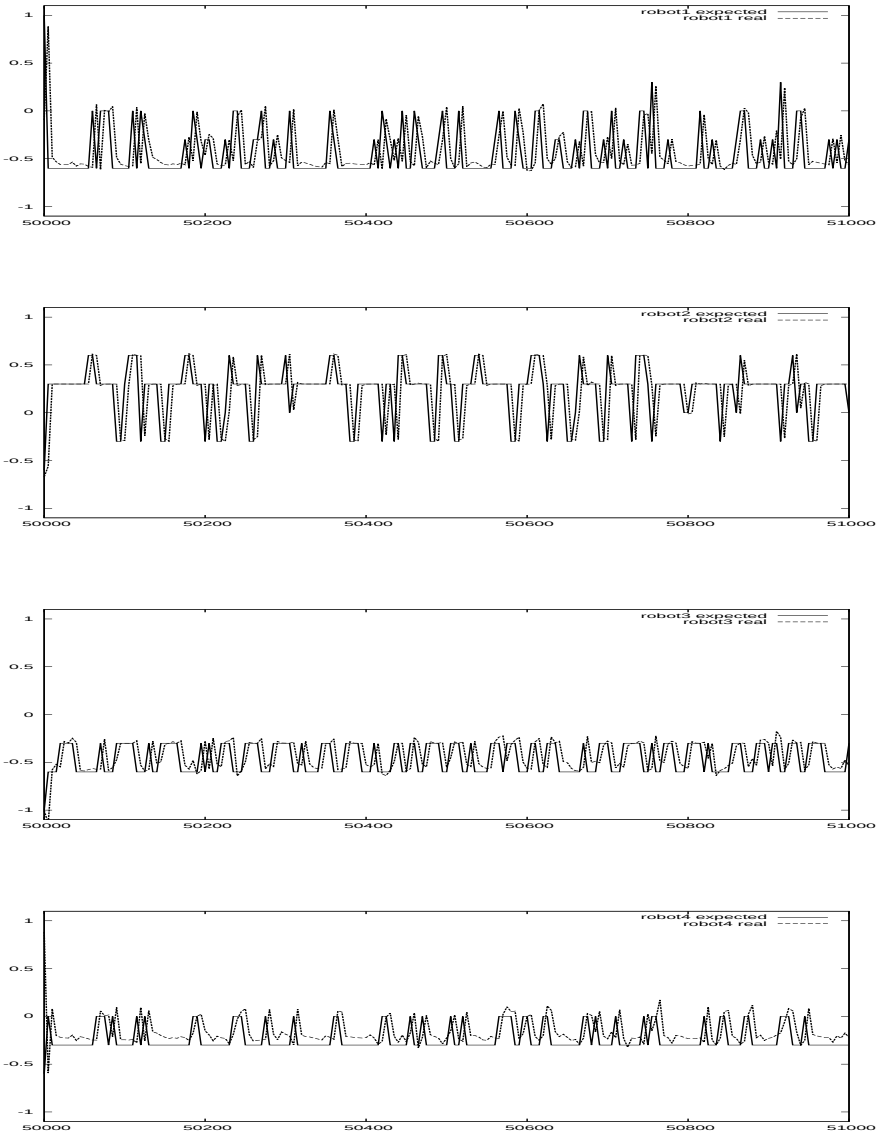


Figure 4.34: Sixth Generation: Individual Actuator Positions 1 - 4 of the Quadruped Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

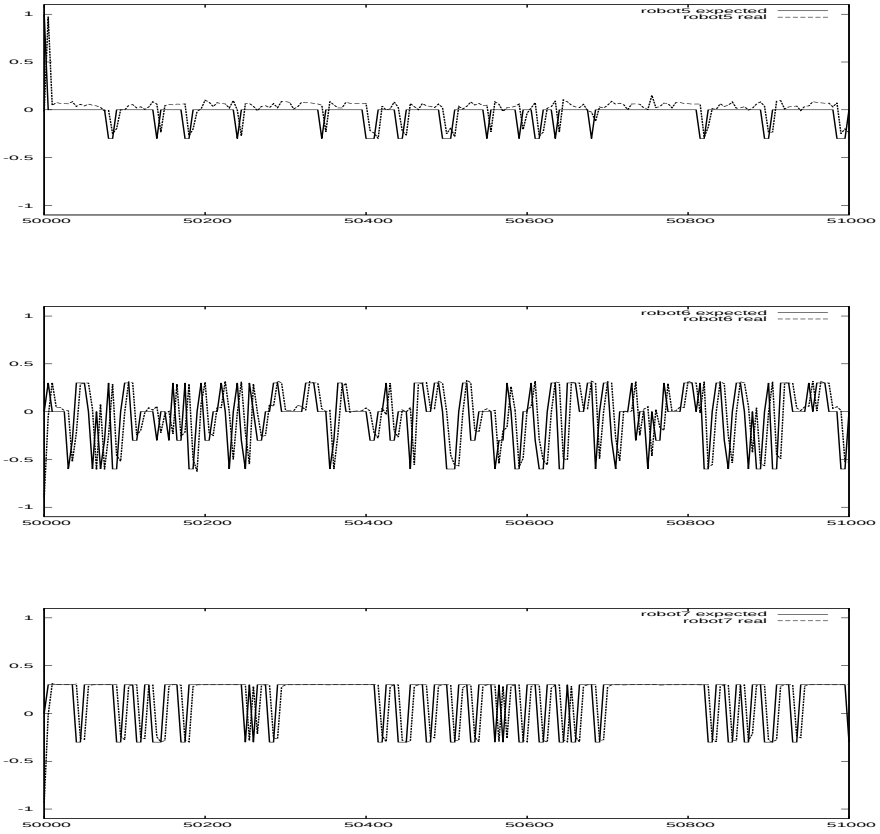


Figure 4.35: Sixth Generation: Individual Actuator Positions 5 - 7 of the Quadruped Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

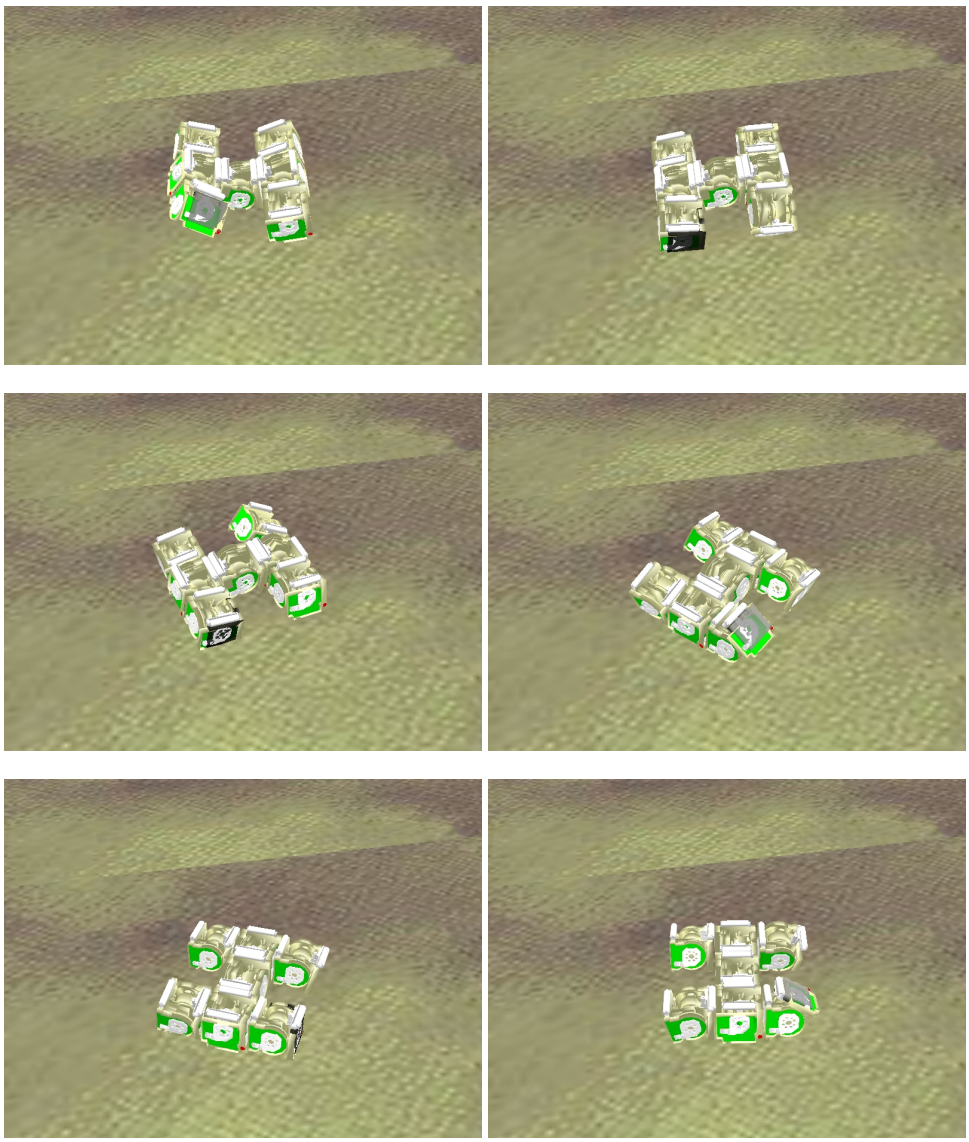


Figure 4.36: Tenth Generation: Evolving Locomotion for a Quadruped Organism.

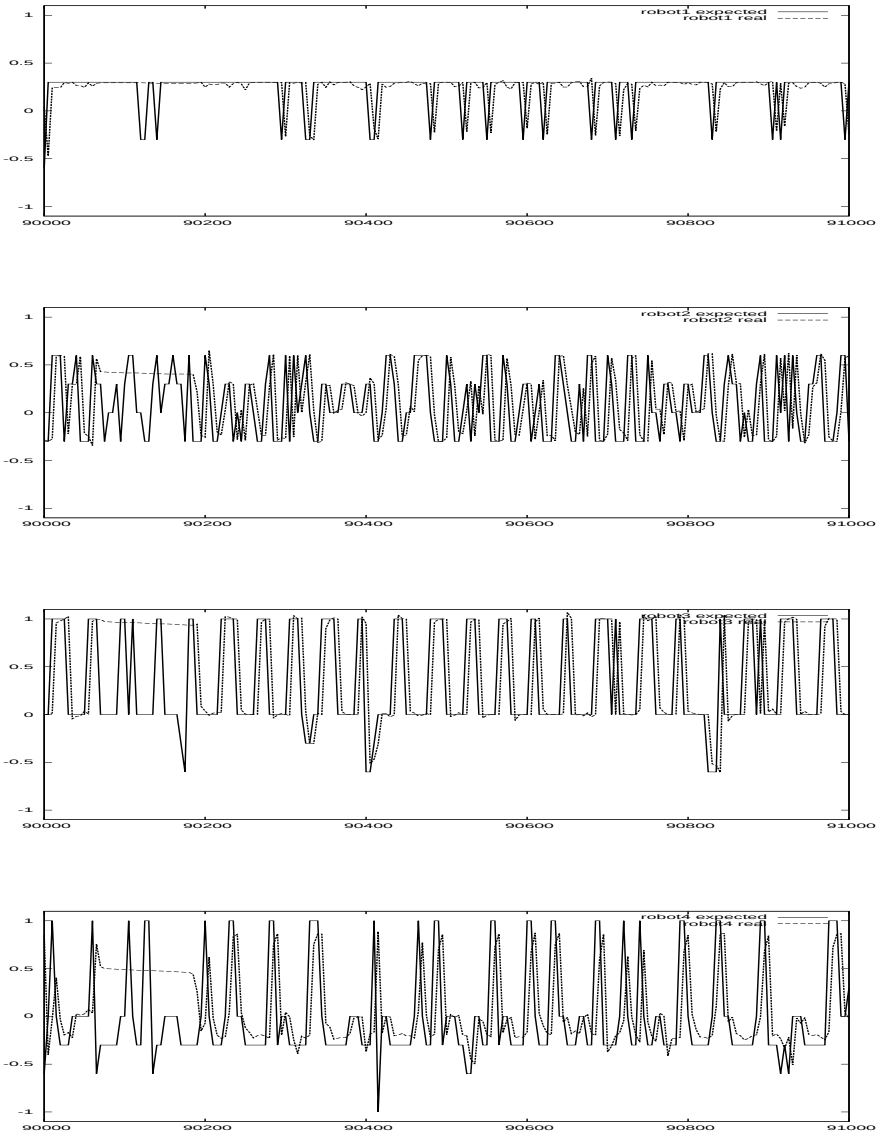


Figure 4.37: Tenth Generation: Individual Actuator Positions 1 - 4 of the Quadruped Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

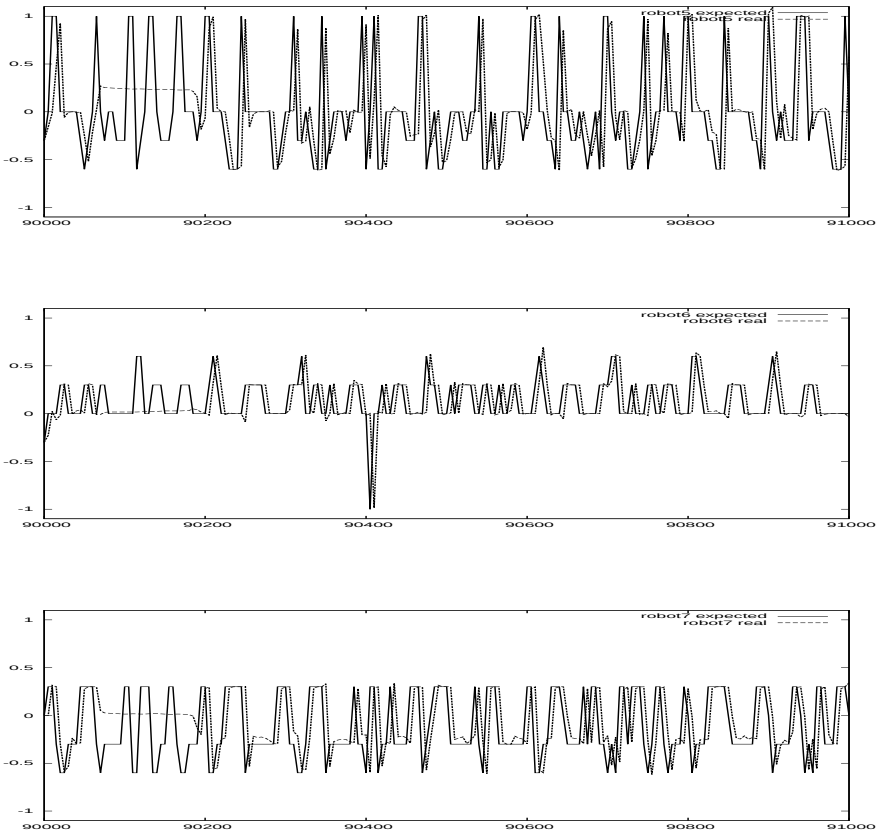


Figure 4.38: Tenth Generation: Individual Actuator Positions 5 - 7 of the Quadruped Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

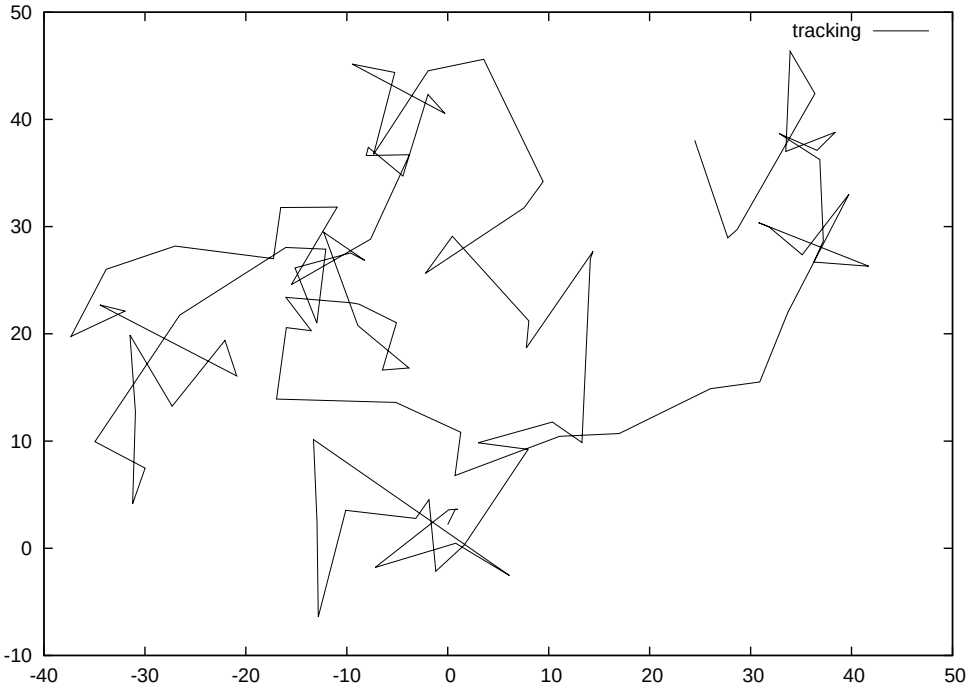


Figure 4.39: First tracking of the quadruped organism on the virtual arena.

X-axis: X direction [-∞, +∞].
Y-axis: Y direction [-∞, +∞].

Second Run: Locomotion for a Quadruped Organism

Because of the long duration of the evolutionary process this experiment could not repeated for 40 times like in the first two experiments of this thesis. For verification of the achieved result of the first run a second run with the same quadruped organism was performed.

Note that a second repetition with the same result can not be regarded as a general proof of the behavior.

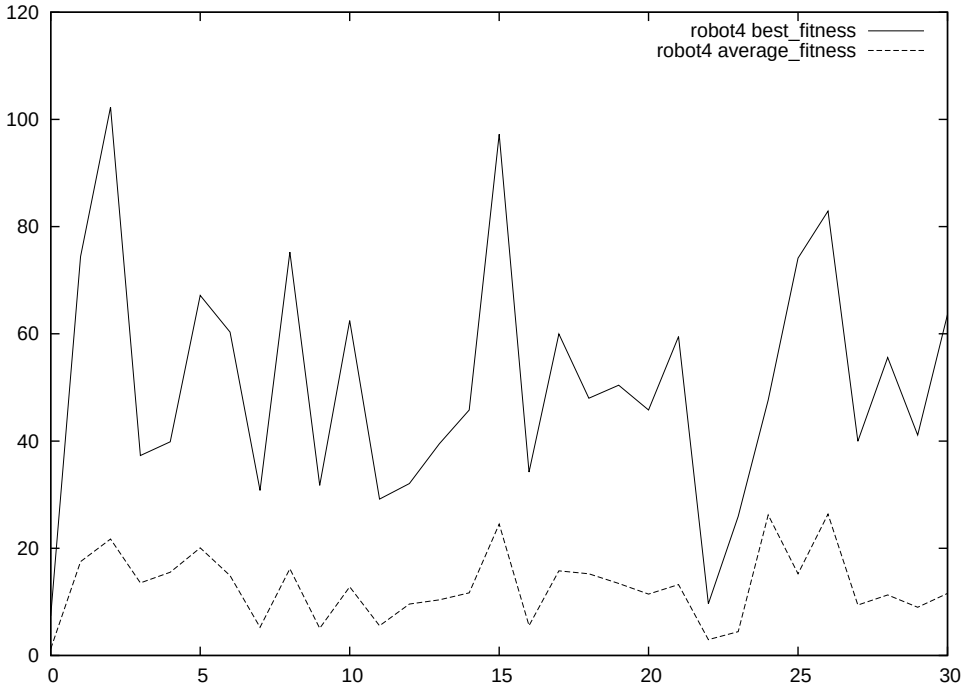


Figure 4.40: First fitness values for the quadruped organism for each generation.

X-axis: generation [0, +∞].
Y-axis: evaluated fitness value [0, +∞].

Parameters

For the second experimental run for evolving locomotion the parameters in Table 4.8 were set.

In the second run the exploitation steps were reseted to the original value of 10. This was done to see which setting is better for the evolution of locomotion.

Parameter	Value	Description
Evolutionary Algorithm Type	SNN	defines the type of network
Using Phenotype Mapping	1	activates phenotype mapping, note CGE can operate directly on genome
Population Size of Island	10	size of the start population of an island
Number of Input Neurons	5	number of input neurons
Number of Output Neurons	3	number of output neurons
Number of Initial Mutations	5	number of mutations which should be performed to a random start genome
Number of Steps for Evaluation	1000	number of steps a genome will be evaluated before switching to the next one
Number Of Generations	undefined	total number of generations
Number of Exploitation Steps	10	number of exploitation steps to perform before a exploration step follows
Default Learning Rate	0.3	Default learning rate for neuron if genome is generated randomly
Default Mutation Probability	0.3	Default mutation probability for a neuron if genome is generated randomly
Fitness Function Type	STRAIGHTON	defines the type of fitness function to use
SelectionMode	ELITISM	defines the type of selection mechanism
Selection Parameter	20.0	parameter to pass to the selection function
EnableParametricMutation	ON	enables parametric mutation
EnableStructuralMutation	ON	enables structural mutation
NewNeuronToLinkRatio	0.5	0.4 means during structural mutation, 40% probability for new neurons and 60% probability of a new link

Table 4.8: Second Run: Parameters for Evolving Locomotion with a Quadruped Organism.

Summary

Just as in the first pass the quadruped organism scrambles straight across the virtual arena floor. Documented tracking data in Figure 4.41 shows the movement behavior which is not as good as in the first pass. The area in which the organism has moved is considerably smaller than before. In Figure 4.42 the associated fitness values are illustrated.

The significant difference between the first and the second pass was in relationship between parametric and structural mutation. In the first run all fifth generation a structural instead of a parametric mutation was performed. In contrast as in the second run it was at every tenth generation. The hope was to make a statement to what relationship is more likely to be chosen.

Comparing the two recorded fitness charts in Figure 4.40 and 4.42 there are significant unexplainable changes. Unfortunately no conclusion can be made about what relationship would be better for evolution.

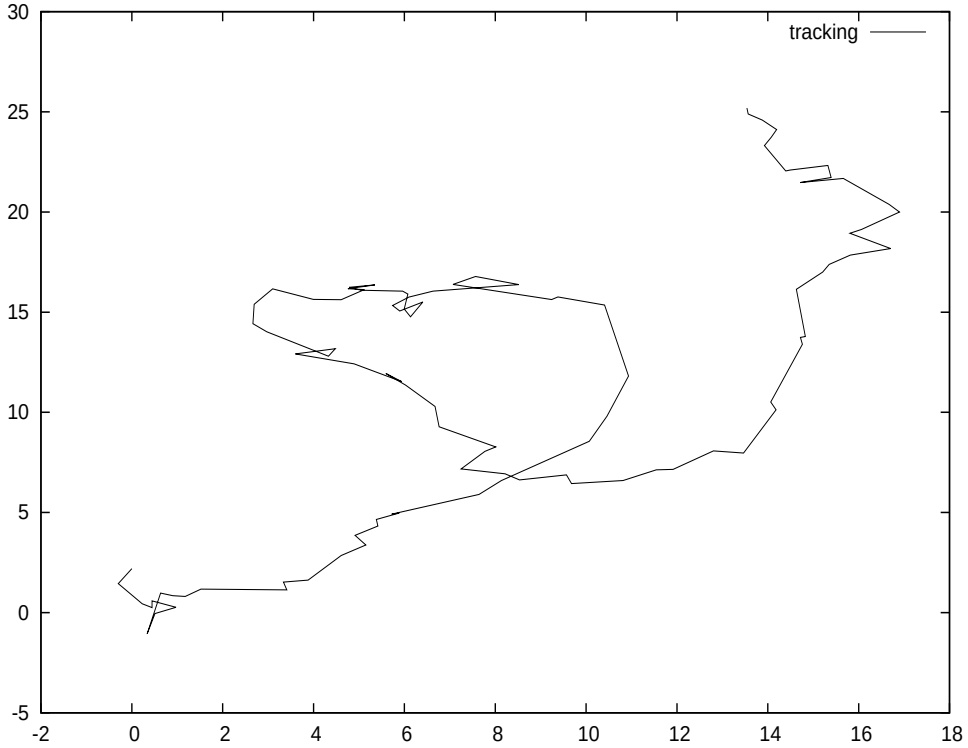


Figure 4.41: Second tracking of the quadruped organism on the virtual arena.

X-axis: X direction [-∞, +∞].
Y-axis: Y direction [-∞, +∞].

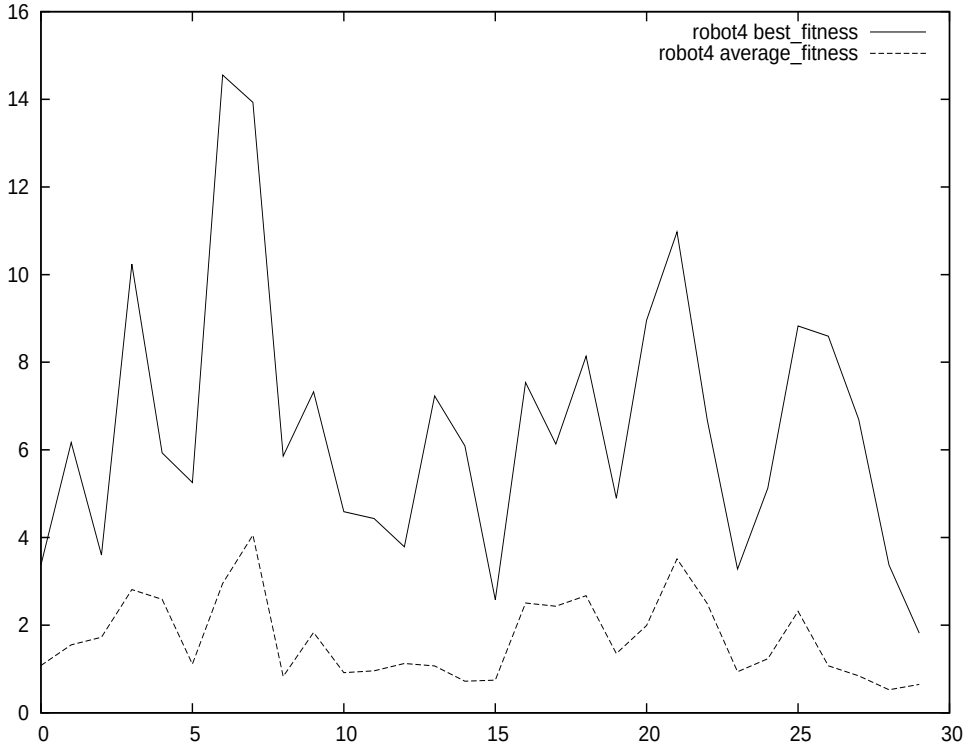


Figure 4.42: Second fitness values for the quadruped organism for each generation.

X-axis: generation [0, +∞].
Y-axis: evaluated fitness value [0, +∞].

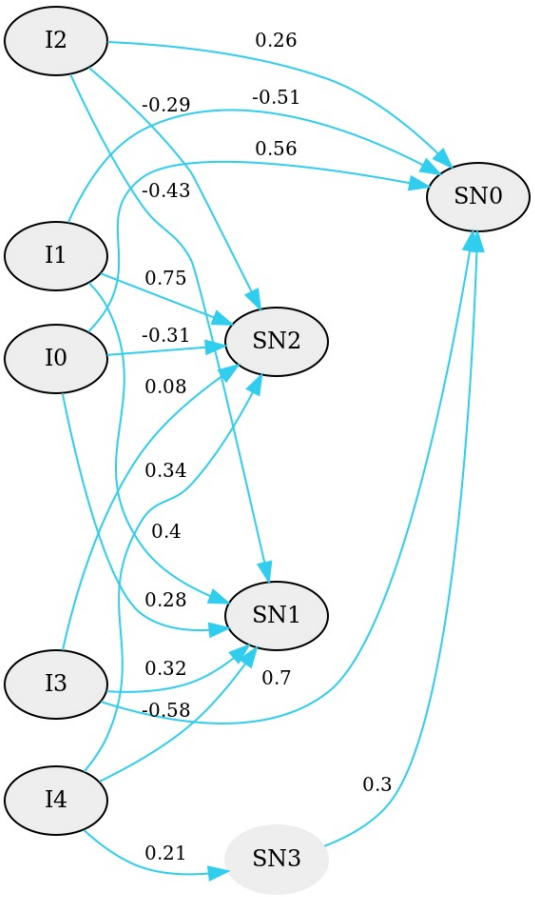


Figure 4.43: Example - Evolved Neural Network.

Example of an evaluated Artificial Neural Network

For demonstration purpose in Figure 4.43 a generated artificial neural network is presented which was made by the evolutionary engine during the development of the quadruped locomotion.

The network consists of the five input neurons “I” marked as I0, I1, I2, I3 and I4. Additional there are the three output spiking neurons SN0, SN1 and SN2. The “SN” output neurons are identifiable with their black border. Neuron SN3 is a special spiking neuron that was added by a structural mutation. It was not included right from the start but was inserted only during the evolutionary process.

4.3.5 Results of the Random Organism

With the random organism pictured in Figure 4.44 were also two documented runs performed and analyzed in the following section.

The used robot organism consists of seven individual autonomous robots from the University of Karlsruhe. Each robot is fixed connected to its neighbor with the help of specific docking elements.

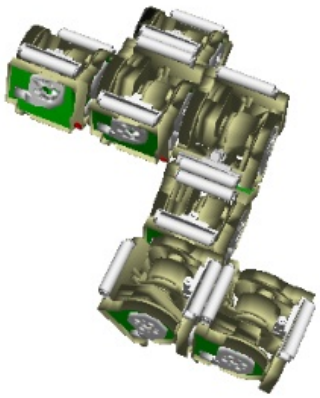


Figure 4.44: A random robot organism consisting of seven individual autonomous robots from the University of Karlsruhe. This kind of robot is one of the three available robot platforms developed during the Symbrion project. The remaining platforms can be seen in Figure 4.7.

		3		
1	2	4		
		5		
		6	7	

Table 4.9: The distributed identification numbers for each robot of the organism.

First Run: Locomotion for a Random Organism

For sake of completeness, each run the list of parameters were added because only little changes would lead to different results. The results were not reproducible without this parameter listing.

Parameters

For the first experimental run for evolving locomotion the parameters in Table 4.10 were set.

Most of the parameters are chosen as in the previous experiment with the same arguments as already mentioned in Chapter 4.3.3.

The exploitation steps was set again to 5 for verifying the consolidated findings of the preceding experiments. In addition the mutation probability was slightly increased with the hope to receive more divergent networks.

Discussion

No changes in the initial conditions leads to no amendments in the starting behavior. The organism begins with lying flat on the arena floor as in the other locomotion experiments. Descriptive images of that behavior of the first generation are shown in Figure 4.45 and the corresponding joint angles in Figure 4.46 and 4.47. In the subsequent pictures the rest of the documented generations can be seen.

In contrast to the already performed experiments in this one the parameter of the "initial mutation" which is set to value five is of relevance. In combination with the higher "mutation probability" the first generation of artificial neural networks produce already output signals. This can be seen in Figure 4.46 and 4.47. Analyzing these seven diagrams for every robot shows that there is still no synchronization between neighboring elements.

Looking to the 14th generation in Figure 4.57 with the joint angles 4.58 and 4.59 the following behavior can be interpreted with caution. The artificial neural network of the robot with identification one produces only rarely peaks. The most time the output is close to zero which means that the robot only passively participating in the movement. This provides a kind of rudimentary appendix.

Most of the labor movement is on the axis with the robots two to six. They move rhythmically with a certain phase shift similar to the snake-shaped organism wherein the robot with identification 6 plays a special role. Its artificial neural network pro-

Parameter	Value	Description
Evolutionary Algorithm Type	SNN	defines the type of network
Using Phenotype Mapping	1	activates phenotype mapping, note CGE can operate directly on genome
Population Size of Island	10	size of the start population of an island
Number of Input Neurons	5	number of input neurons
Number of Output Neurons	3	number of output neurons
Number of Initial Mutations	5	number of mutations which should be performed to a random start genome
Number of Steps for Evaluation	1000	number of steps a genome will be evaluated before switching to the next one
Number Of Generations	undefined	total number of generations
Number of Exploitation Steps	5	number of exploitation steps to perform before a exploration step follows
Default Learning Rate	0.3	Default learning rate for neuron if genome is generated randomly
Default Mutation Probability	0.5	Default mutation probability for a neuron if genome is generated randomly
Fitness Function Type	STRAIGHTON	defines the type of fitness function to use
SelectionMode	ELITISM	defines the type of selection mechanism
Selection Parameter	20.0	parameter to pass to the selection function
EnableParametricMutation	ON	enables parametric mutation
EnableStructuralMutation	ON	enables structural mutation
NewNeuronToLinkRatio	0.5	0.4 means during structural mutation, 40% probability for new neurons and 60% probability of a new link

Table 4.10: First Run: Parameters for Evolving Locomotion with a Random Organism.

duces no more output signal and the robot holds its position. This robot is no longer required to propel active the organism and serves now as a stiffened backbone.

Summary

In summary it could be shown that the random organism is partly able to move across the arena floor (Figure 4.60). It starts at point [0,0] like in the whole experiments before. The shape of the random organism is unsuitable to produce adequate movements. Thus, the calculated result is not as good as in the experiments before in which symmetrical organisms were elected.

The area in which the robot moves in the arena is very limited. But the evolutionary process has managed to partially get around a straight movement which is a remarkable achievement after 14th generations for the type of organism. These successes can be recognized also in Figure 4.61 where the fitness values are documented.

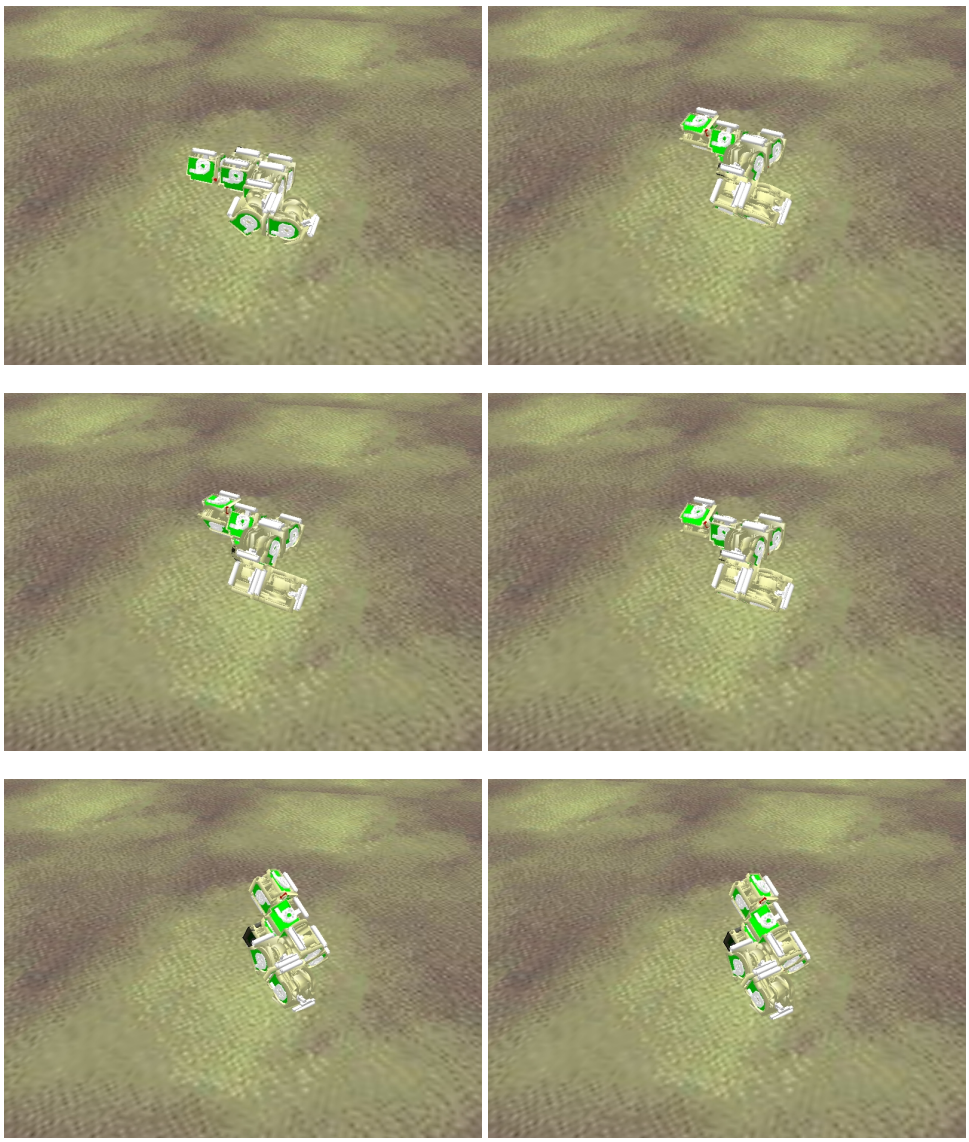


Figure 4.45: First Generation: Evolving Locomotion for a Random Organism.

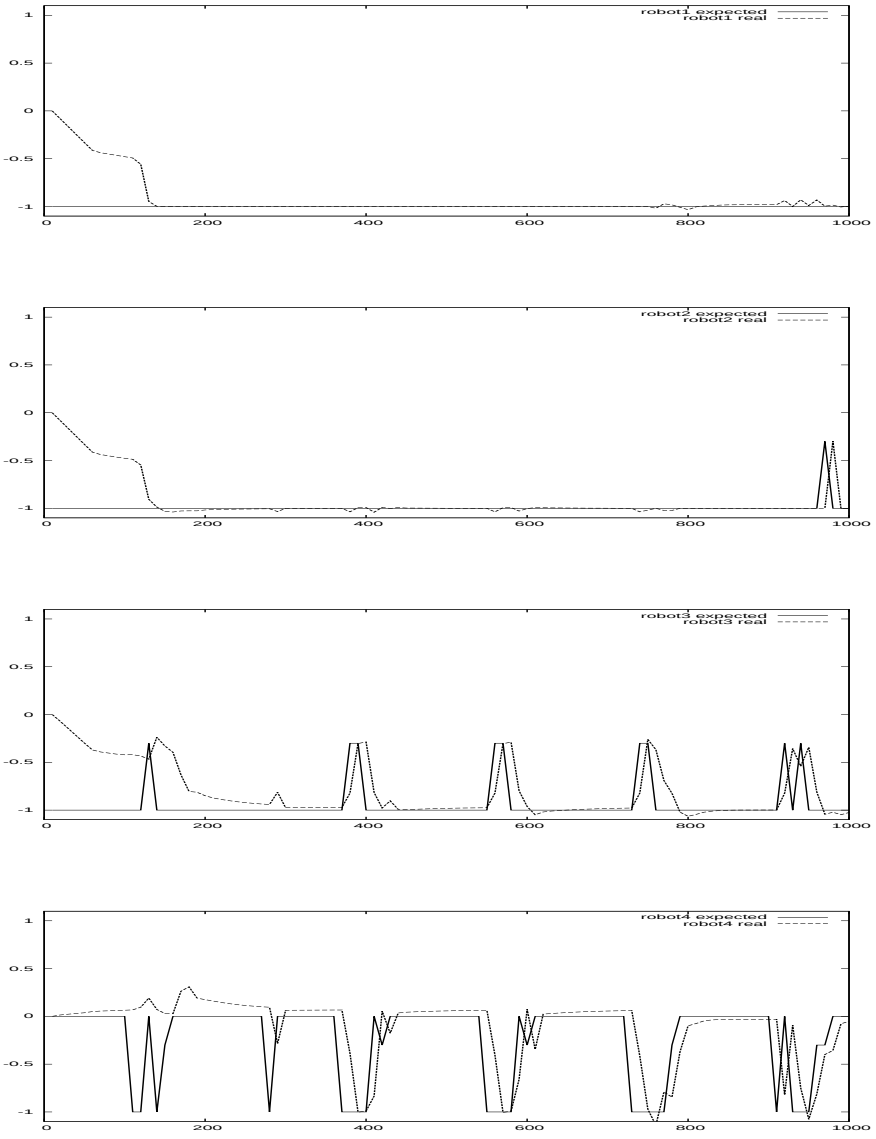


Figure 4.46: First Generation: Individual Actuator Positions 1 - 4 of the Random Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

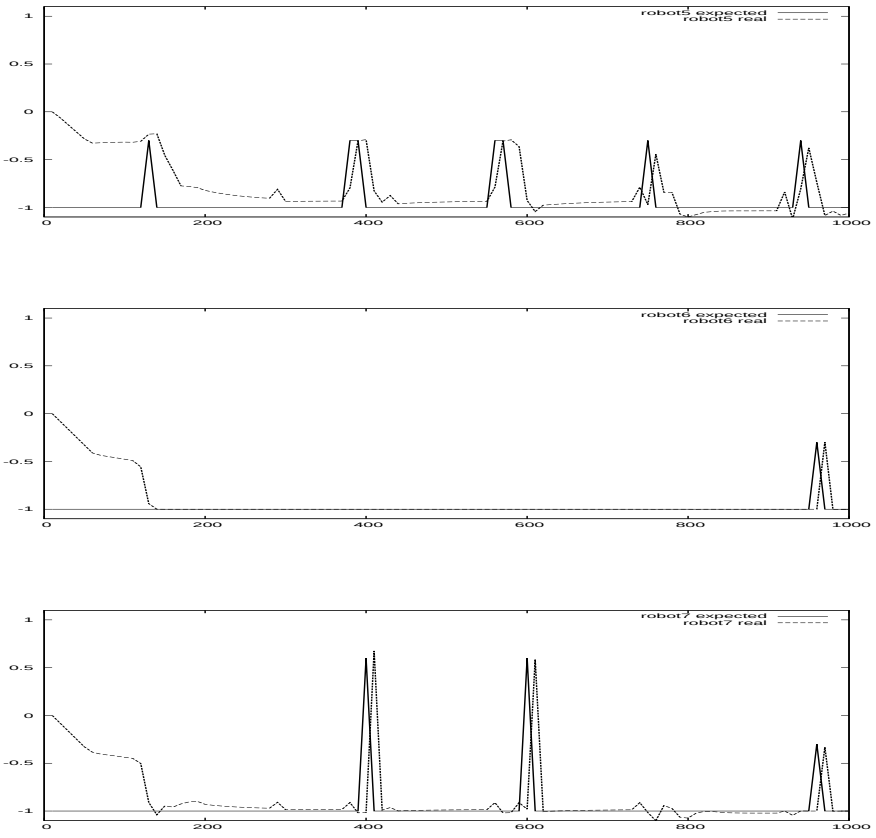


Figure 4.47: First Generation: Individual Actuator Positions 5 - 7 of the Random Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

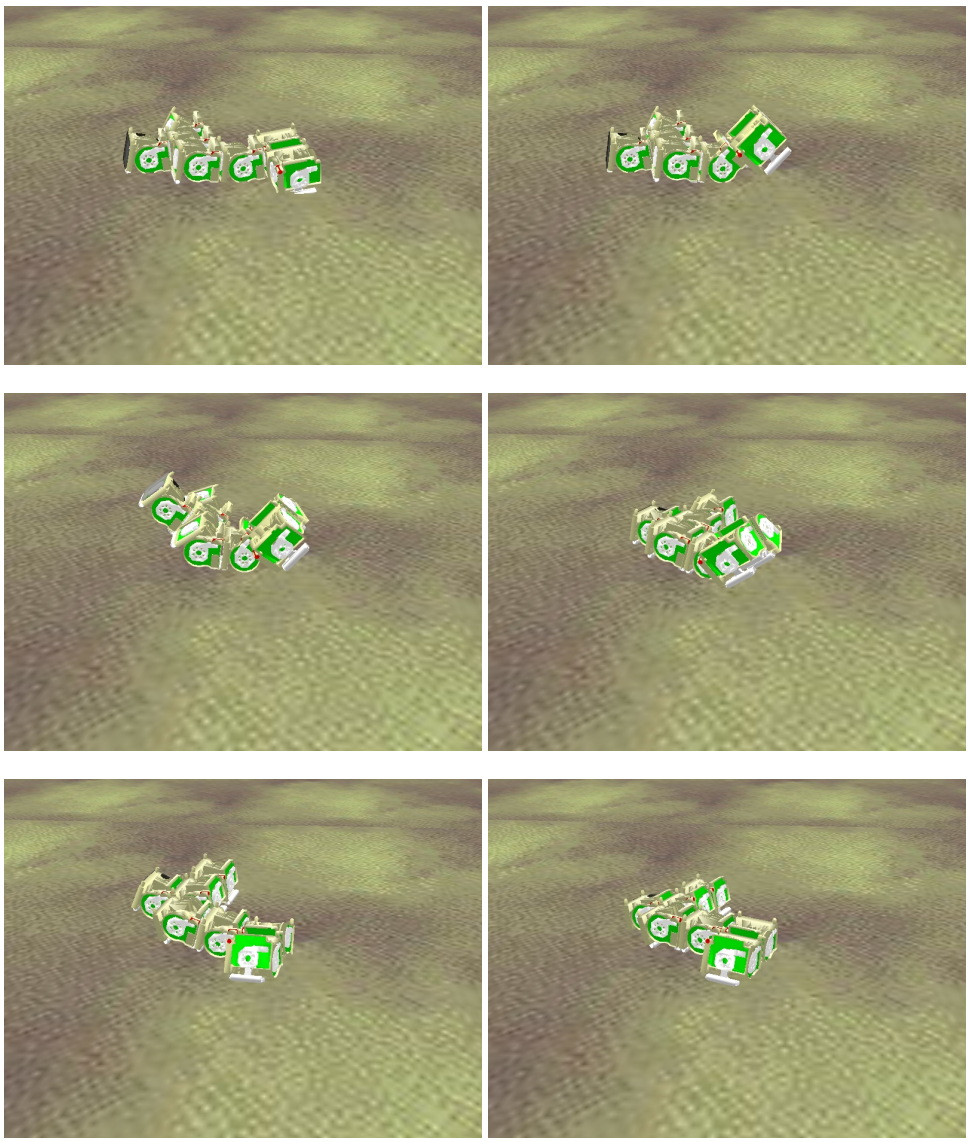


Figure 4.48: Fifth Generation: Evolving Locomotion for a Random Organism.

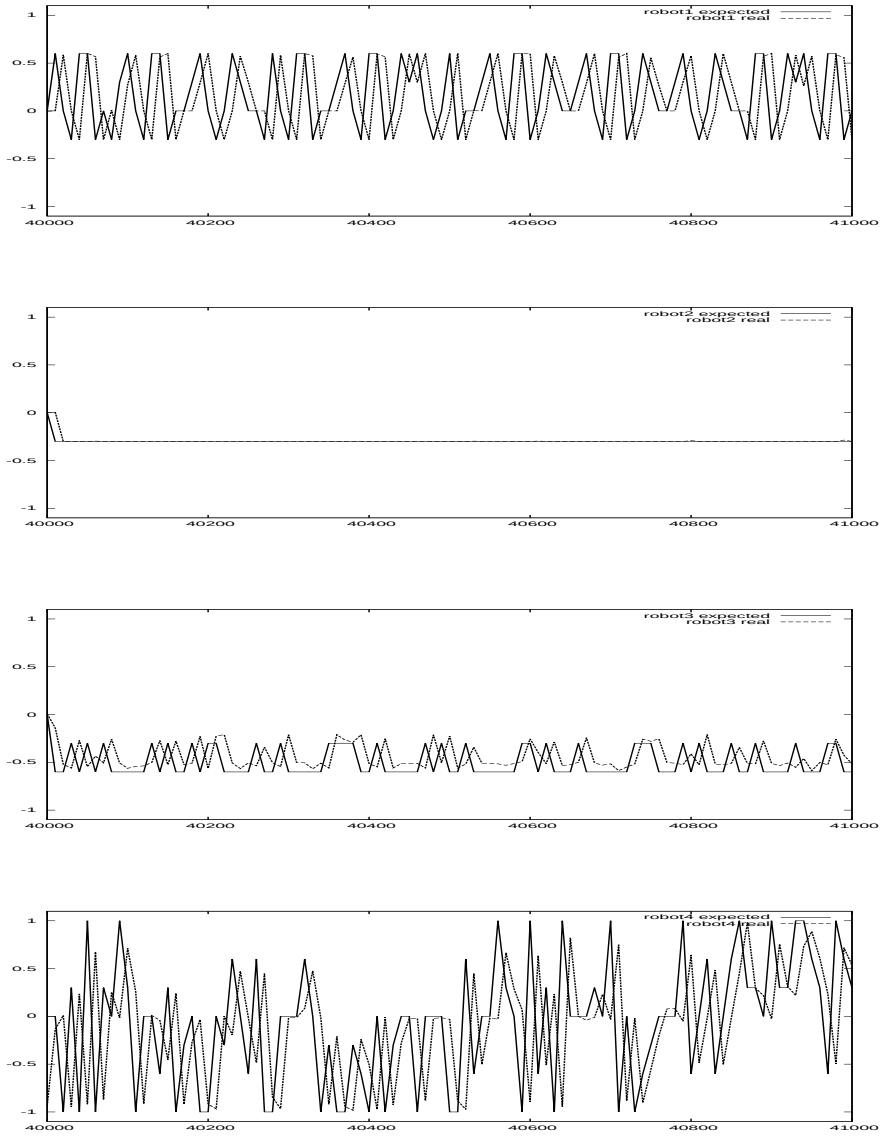


Figure 4.49: Fifth Generation: Individual Actuator Positions 1 - 4 of the Random Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

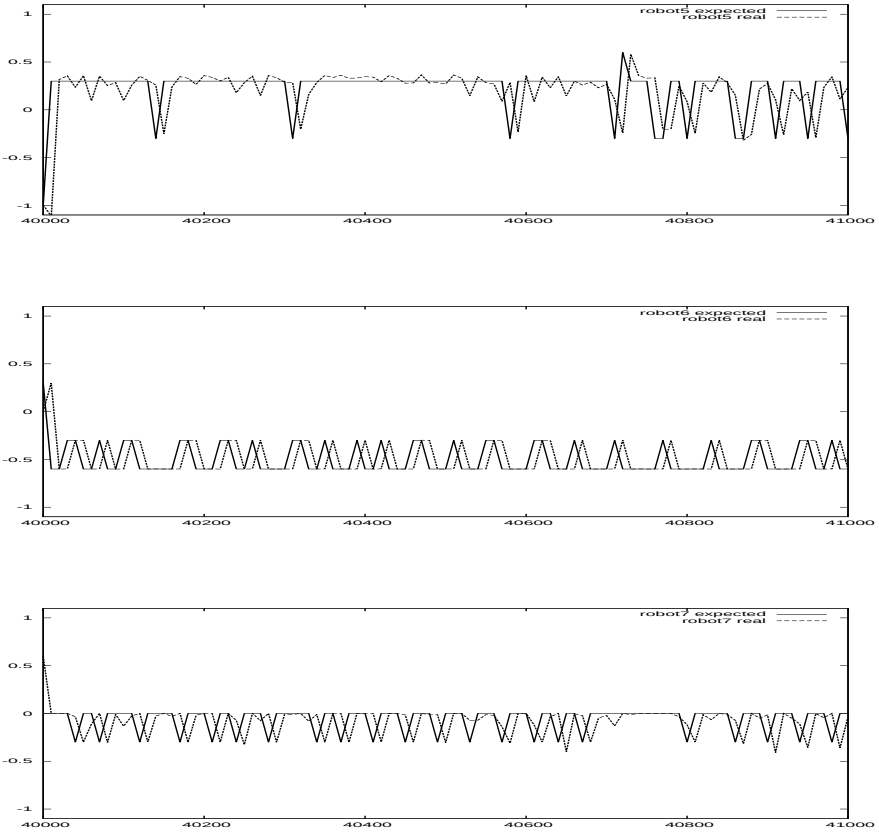


Figure 4.50: Fifth Generation: Individual Actuator Positions 5 - 7 of the Random Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

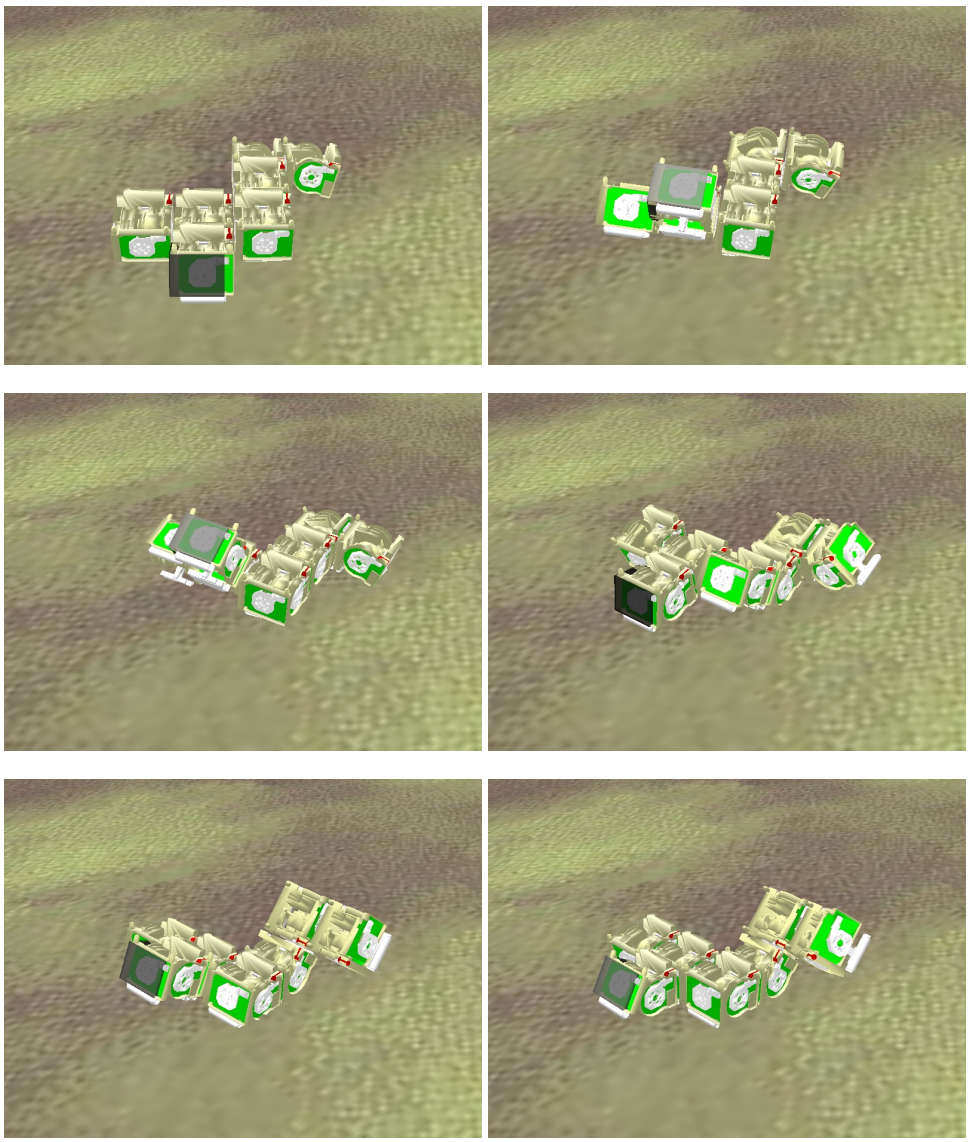


Figure 4.51: Eighth Generation: Evolving Locomotion for a Random Organism.

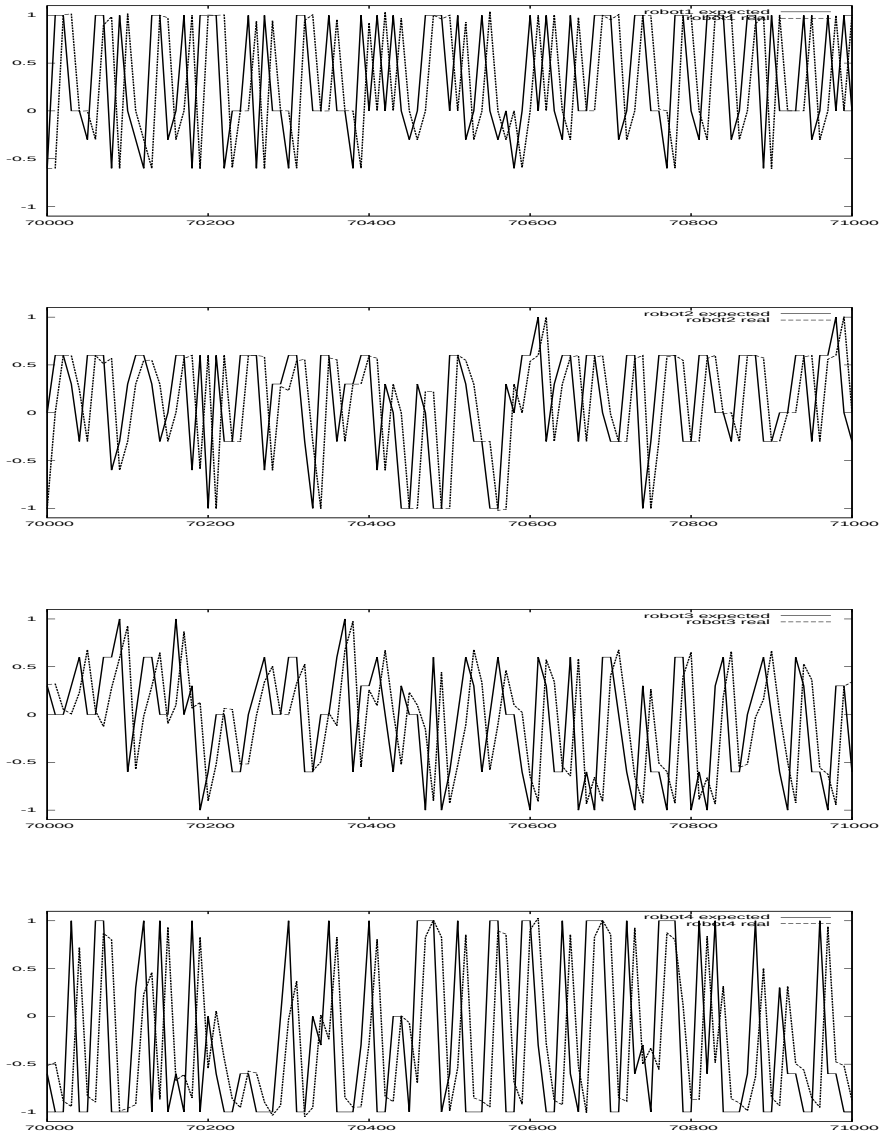


Figure 4.52: Eighth Generation: Individual Actuator Positions 1 - 4 of the Random Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

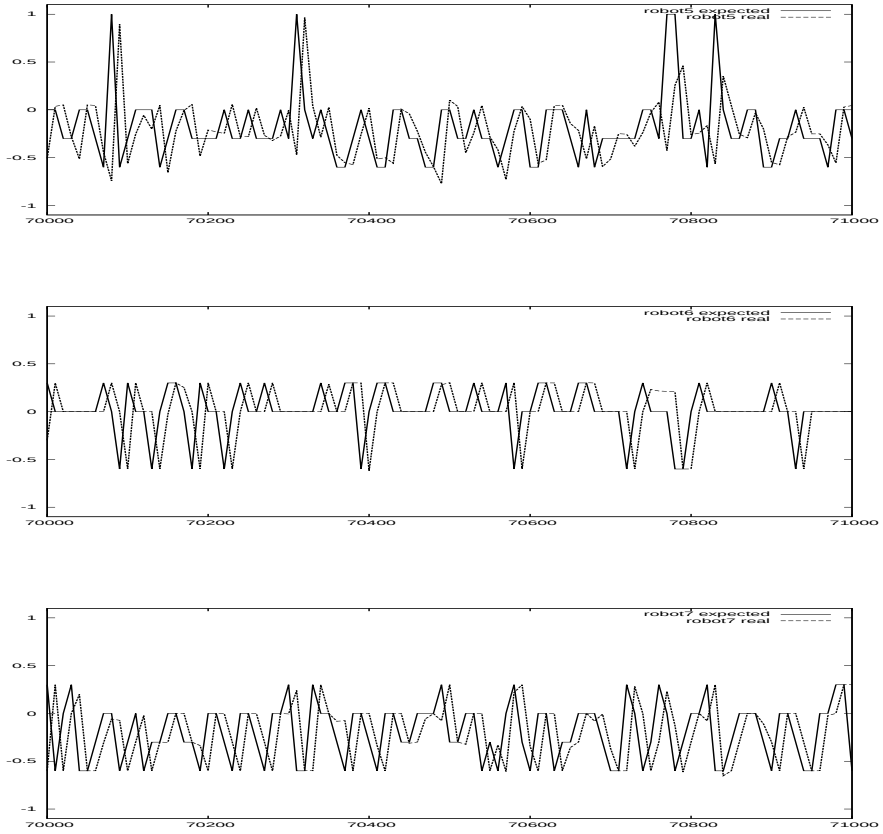


Figure 4.53: Eighth Generation: Individual Actuator Positions 5 - 7 of the Random Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

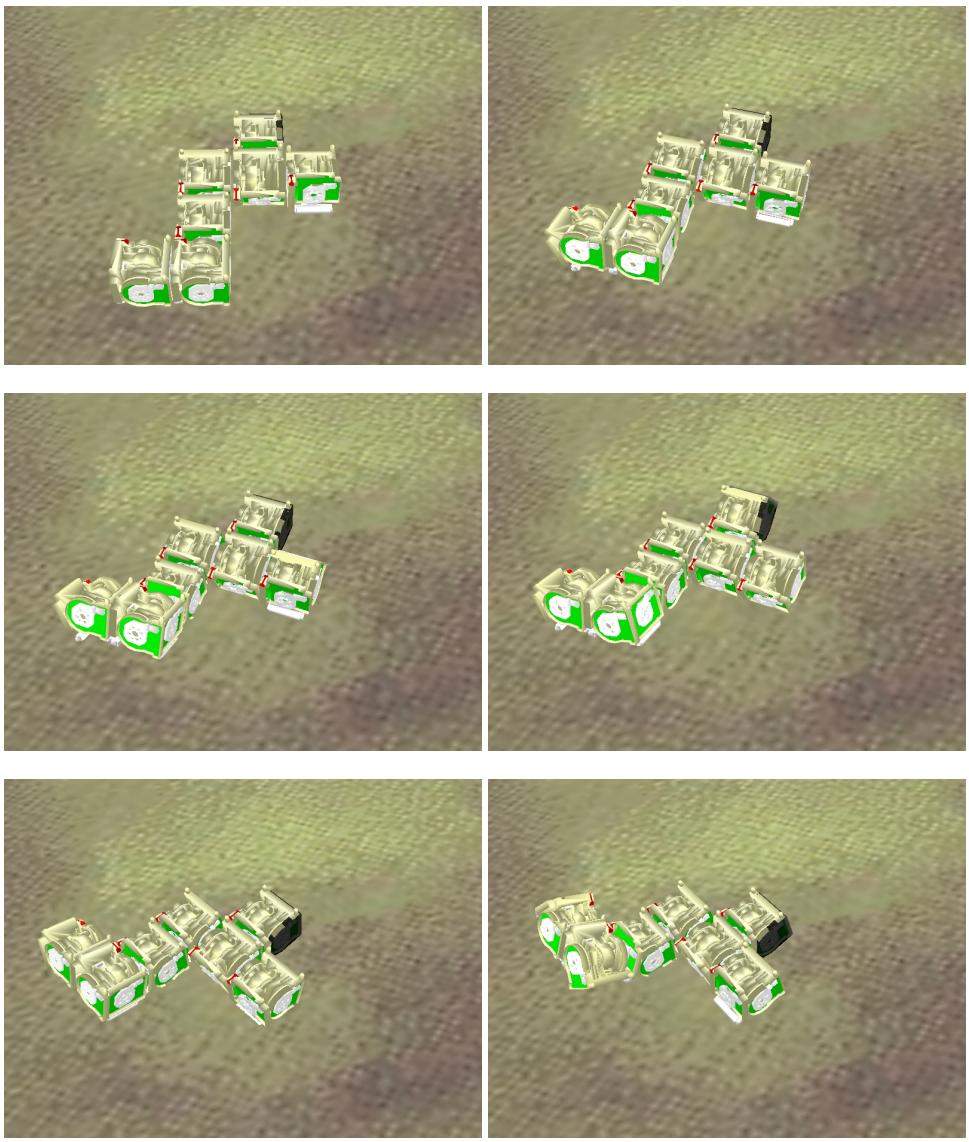


Figure 4.54: 11th Generation: Evolving Locomotion for a Random Organism.

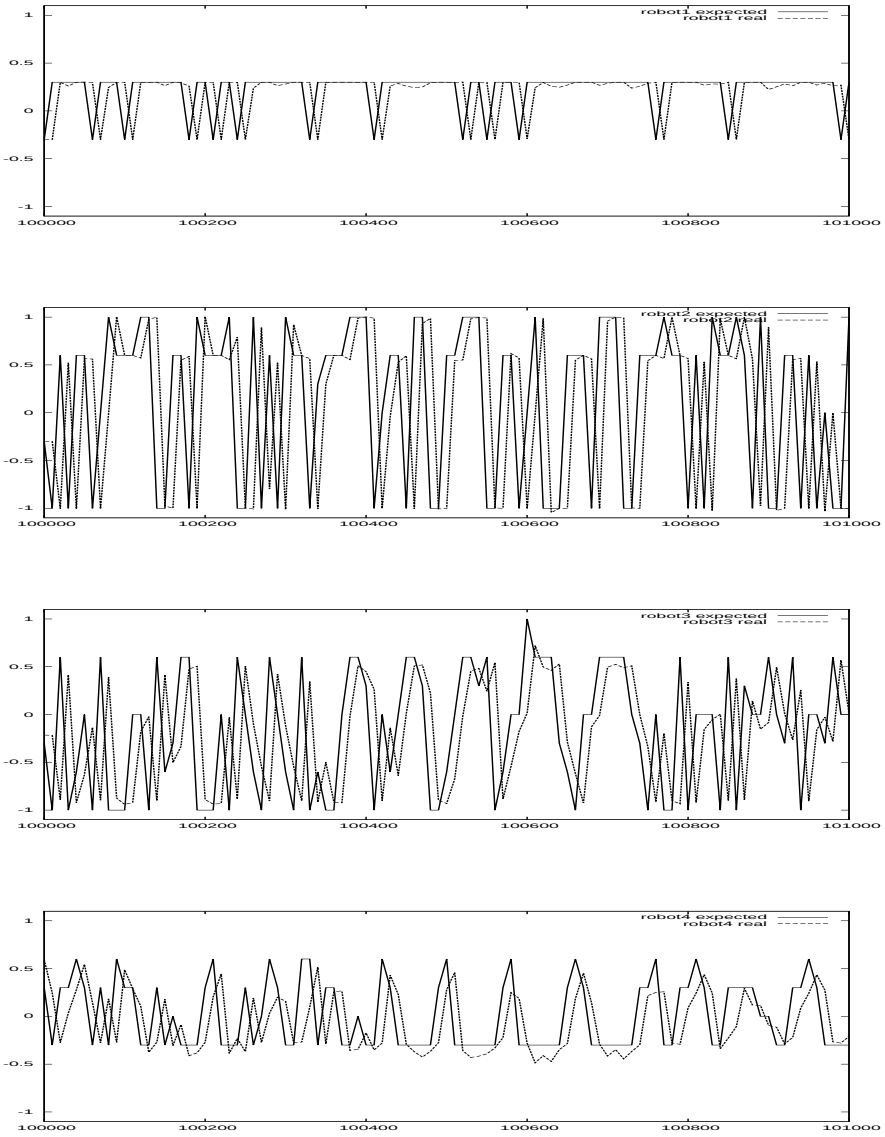


Figure 4.55: 11th Generation: Individual Actuator Positions 1 - 4 of the Random Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

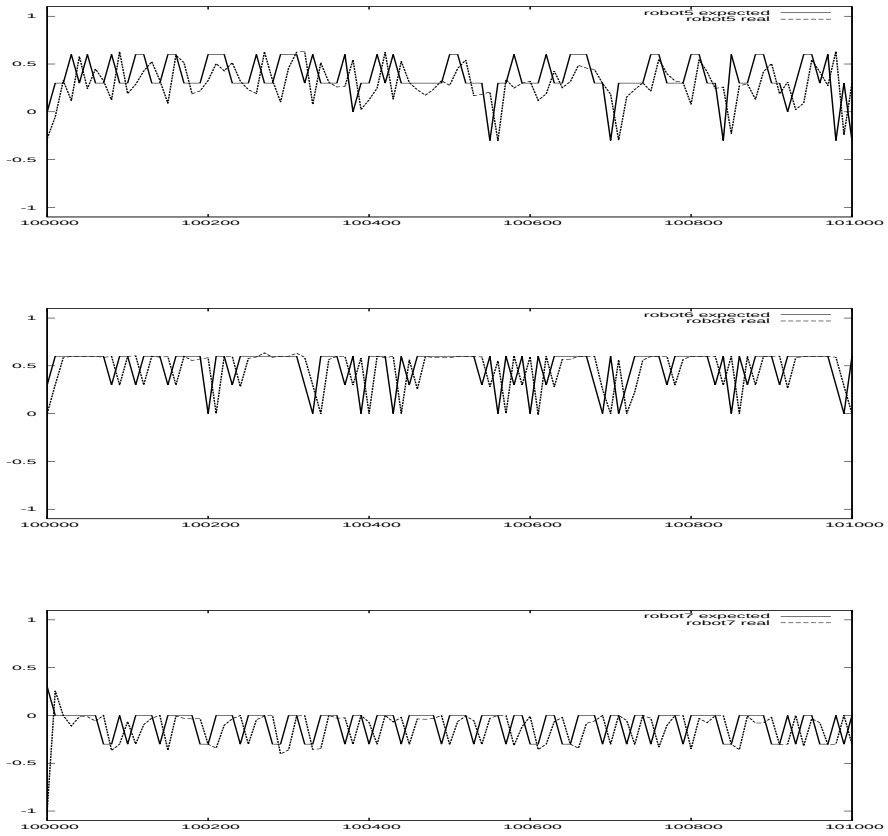


Figure 4.56: 11th Generation: Individual Actuator Positions 5 - 7 of the Random Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

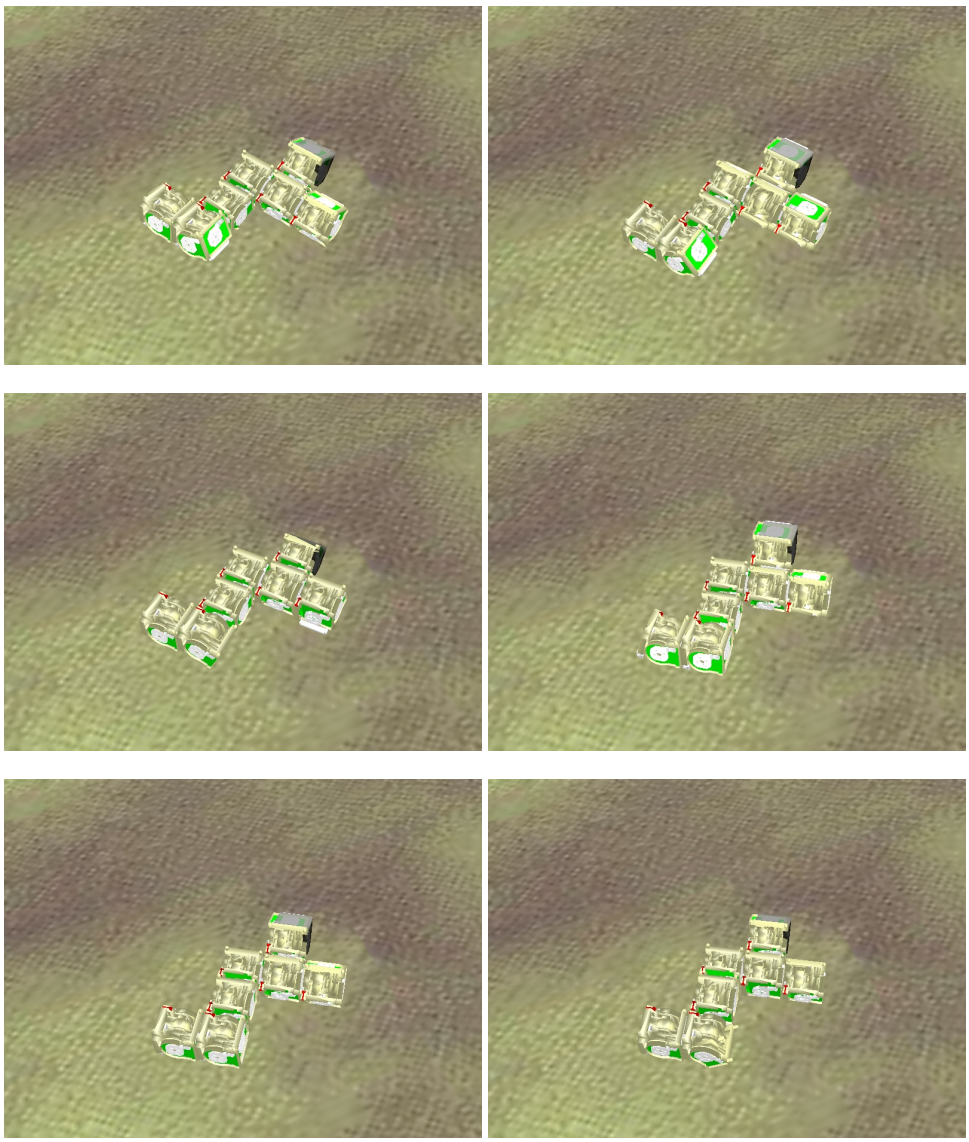


Figure 4.57: 14th Generation: Evolving Locomotion for a Random Organism.

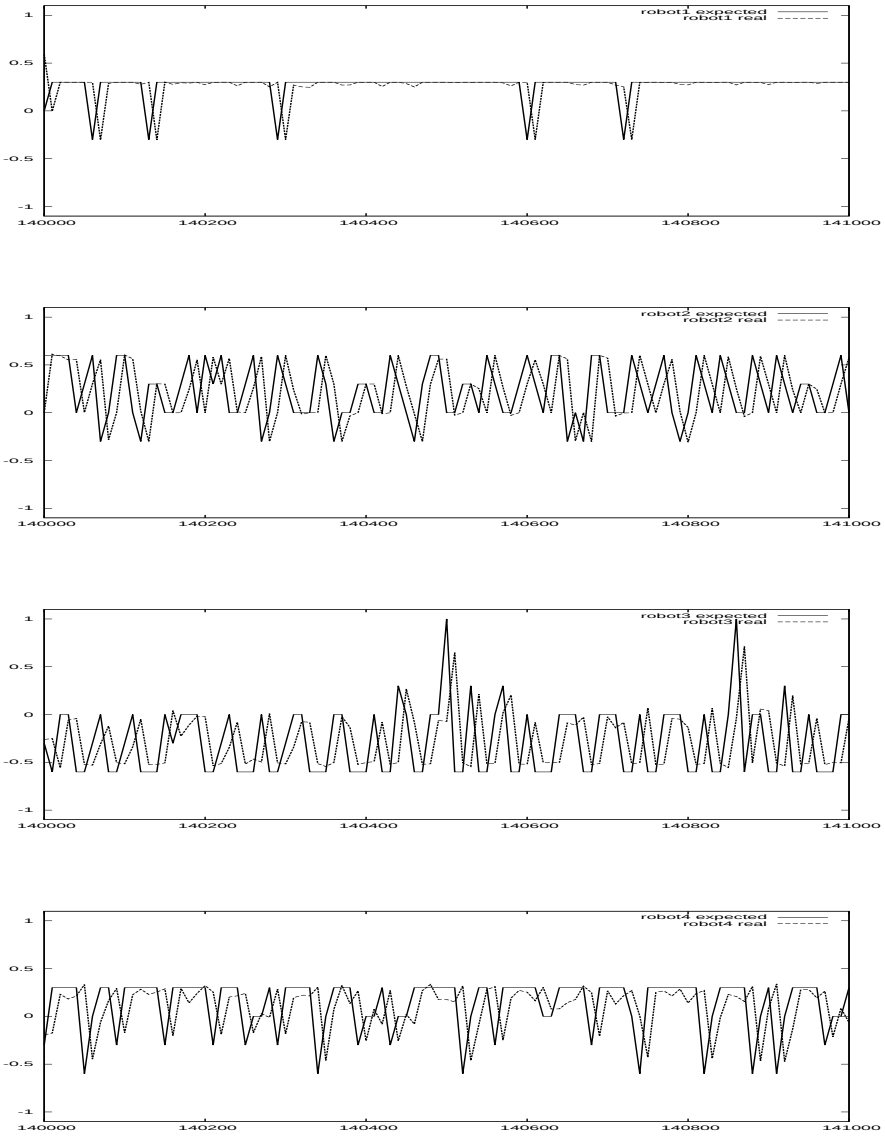


Figure 4.58: 14th Generation: Individual Actuator Positions 1 - 4 of the Random Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

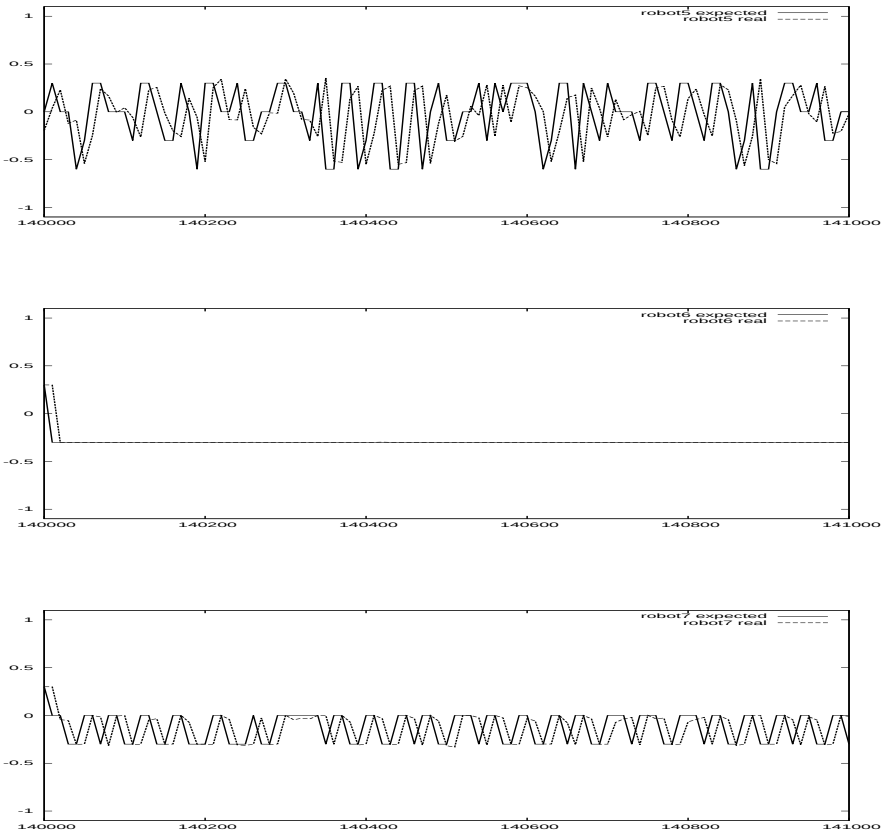


Figure 4.59: 14th Generation: Individual Actuator Positions 5 - 7 of the Random Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

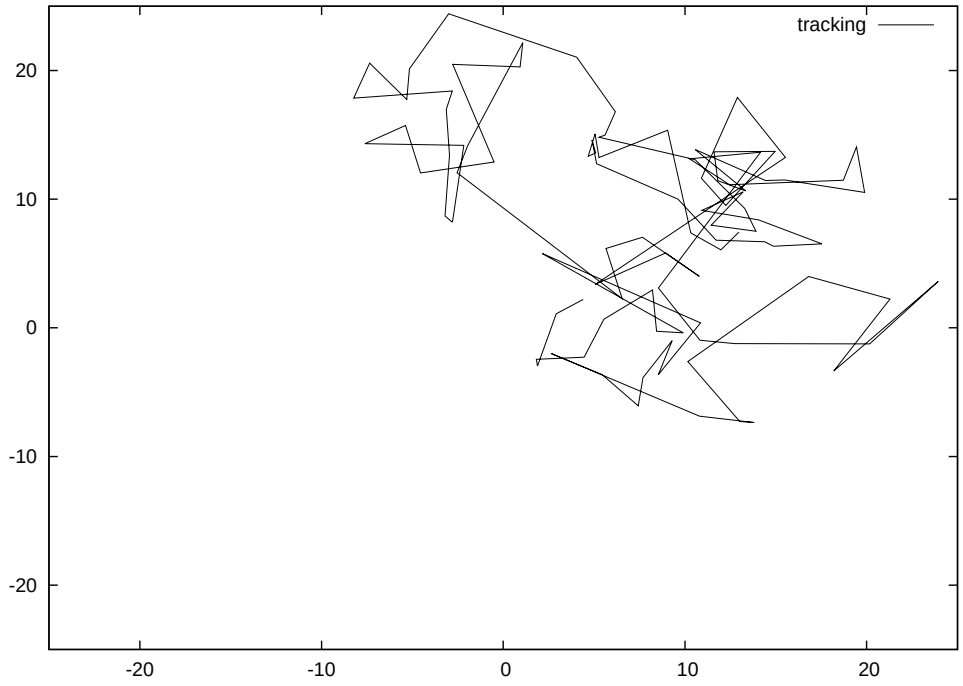


Figure 4.60: First tracking of the random organism on the virtual arena.

X-axis: X direction [-∞, +∞].
Y-axis: Y direction [-∞, +∞].

Second Run: Locomotion for a Random Organism

Because of the long duration of the evolutionary process this experiment could not be repeated for 40 times like the first two experiments of this thesis. For verification of the achieved result of the first run a second run with the same random organism was performed.

Note that a second repetition with the same result can not be regarded as a general proof of the behavior.

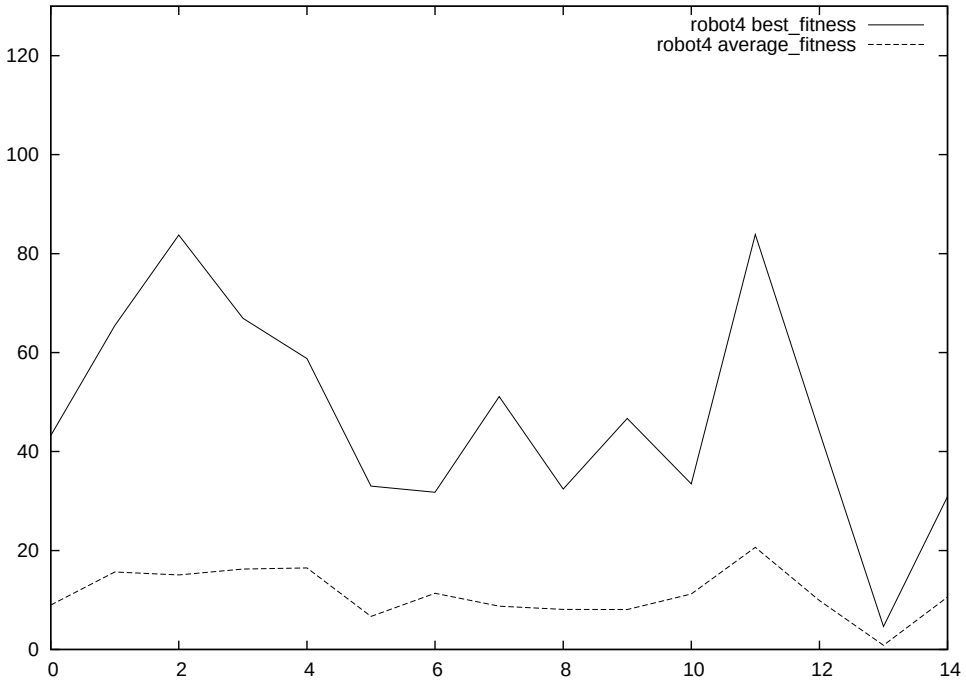


Figure 4.61: First fitness values for the random organism for each generation.

X-axis: generation [0, +∞].
Y-axis: evaluated fitness value [0, +∞].

Parameters

For the second experimental run for evolving locomotion the parameters in Table 4.11 were set.

In the second run the parameters were not changed. This was done with the hope that the evolution under these circumstances leads to similar results as before.

Parameter	Value	Description
Evolutionary Algorithm Type	SNN	defines the type of network
Using Phenotype Mapping	1	activates phenotype mapping, note CGE can operate directly on genome
Population Size of Island	10	size of the start population of an island
Number of Input Neurons	5	number of input neurons
Number of Output Neurons	3	number of output neurons
Number of Initial Mutations	5	number of mutations which should be performed to a random start genome
Number of Steps for Evaluation	1000	number of steps a genome will be evaluated before switching to the next one
Number Of Generations	undefined	total number of generations
Number of Exploitation Steps	5	number of exploitation steps to perform before a exploration step follows
Default Learning Rate	0.3	Default learning rate for neuron if genome is generated randomly
Default Mutation Probability	0.5	Default mutation probability for a neuron if genome is generated randomly
Fitness Function Type	STRAIGHTON	defines the type of fitness function to use
SelectionMode	ELITISM	defines the type of selection mechanism
Selection Parameter	20.0	parameter to pass to the selection function
EnableParametricMutation	ON	enables parametric mutation
EnableStructuralMutation	ON	enables structural mutation
NewNeuronToLinkRatio	0.5	0.4 means during structural mutation, 40% probability for new neurons and 60% probability of a new link

Table 4.11: Second Run: Parameters for Evolving Locomotion with a Random Organism.

Summary

In summary the result of the second run looks much better in contrast to the first pass. As already mentioned the generation of movement for this kind of random organism is very difficult. But in the second pass the tracking (Figure 4.62) of the organism resembles the results of the experiments with snake-shaped and quadruped organisms.

The random body moves in a fairly large area around the arena. Despite unchanged initial conditions the two performed experiments leads to significantly different results. This confirm the finding that only contingency events during evolutionary process determines where the evolution ends.

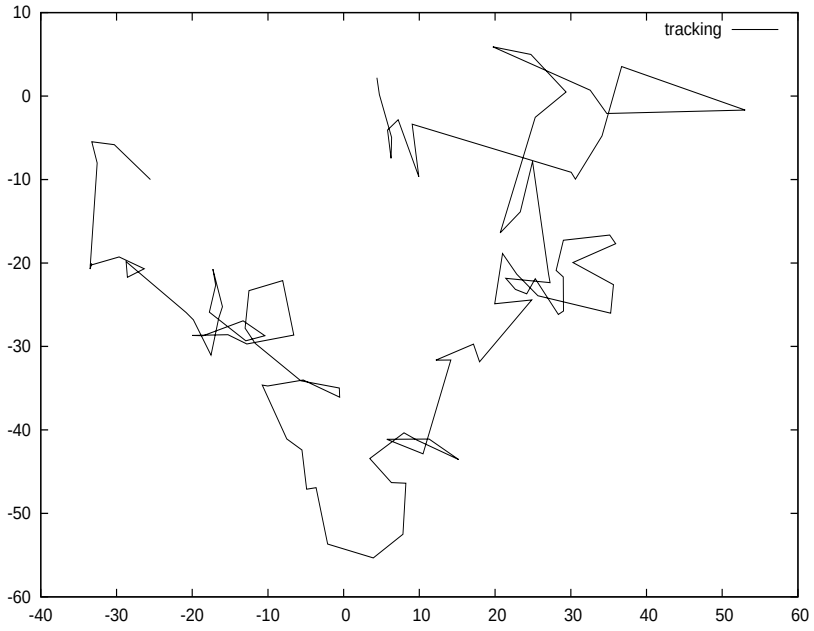


Figure 4.62: Second tracking of the random organism on the virtual arena.

X-axis: X direction [-∞, +∞].
Y-axis: Y direction [-∞, +∞].

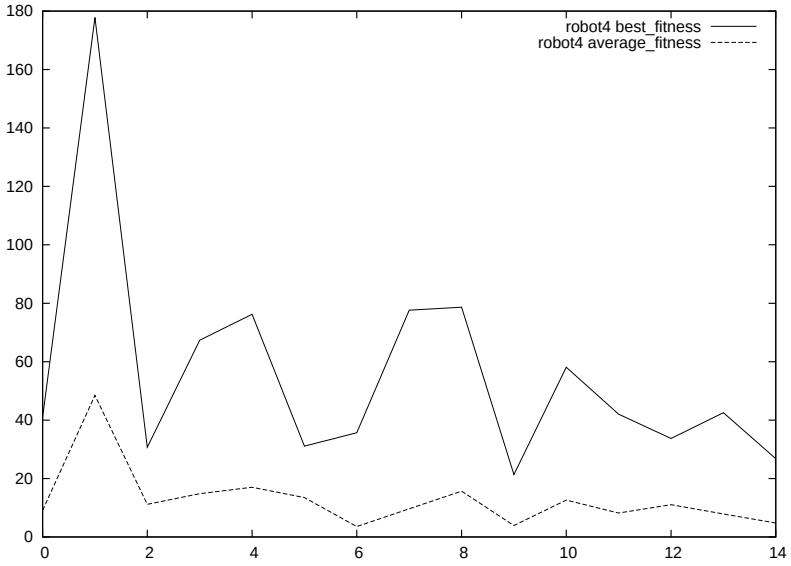


Figure 4.63: Second fitness values for the random organism for each generation.

X-axis: generation [0, +∞].
Y-axis: evaluated fitness value [0, +∞].

Example of an evaluated Artificial Neural Network

For demonstration purpose in Figure 4.64 a generated artificial neural network is presented which was made by the evolutionary engine during the development of the random locomotion.

The network consists of the five input neurons "I" marked as I0, I1, I2, I3 and I4. Additional there are the three output spiking neurons SN0, SN1 and SN2. The "SN" output neurons are identifiable with their black border. Neuron SN3 is a special spiking neuron that was added by a structural mutation. It was not included right from the start but was inserted only during the evolutionary process.

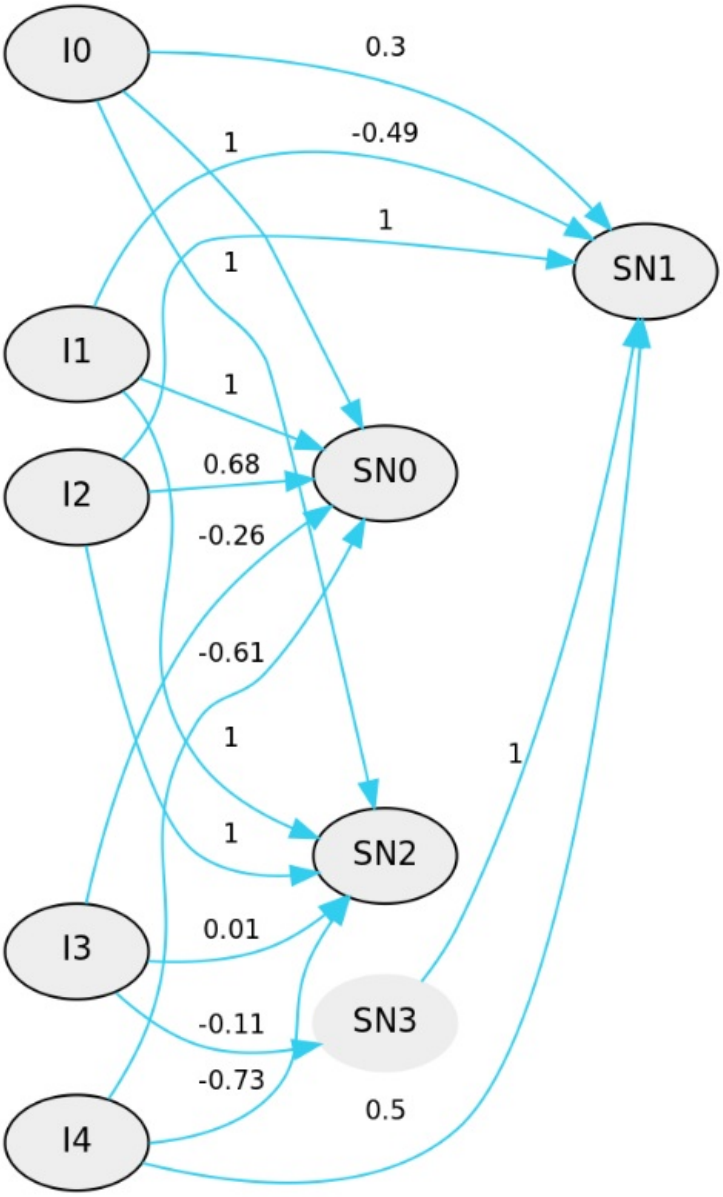


Figure 4.64: Example - Evolved Neural Network.

4.3.6 Results of the Disrupted Organism

This experiment was performed to show how mighty the on-board evolution is. The used organism consists of two separate organisms that are connected to each other.

First a locomotion pattern for this structure should be evolved for ten generations. After that phase the docking element between these "two" parts will be opened and the individual organisms will be separated. This simulates for example a defect or a conscious discarding of one leg of a quadruped if it gets stuck in a hole.

Back then a single robot would be no longer functional but in this approach will be shown that it is possible to generate a new matching pattern of movement. Thus, the robot can still be used and it is not lost.

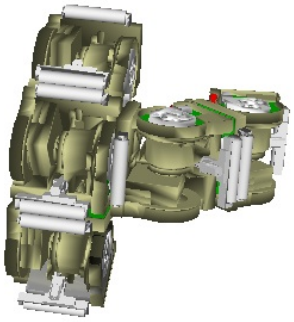


Figure 4.65: A defective robot organism consisting of five individual autonomous robots from the University of Karlsruhe. This kind of robot is one of three available robot platforms developed during the Symbrion project. The remaining platforms can be seen in Figure 4.7.

	5	
2	3	4
	1	

Table 4.12: The distributed identification numbers for each robot of the organism.

First Run: Locomotion for a Disrupting Organism

For sake of completeness, each run the list of parameters were added because only little changes would lead to different results. The results were not reproducible without this parameter listing.

Parameters

For the experimental run the parameters in Table 4.13 were set.

All robots of the entire organism are "equipped" with a population of 10 individuals. Each individual is evaluated and has the chance to show how well the current used artificial neural network is adapted to the task of locomotion during this set time period of 1000 time steps. Because the evolutionary process is not terminating the "Number Of Generations" parameter is set to undefined but in real the simulation was aborted after 20 generations.

The individuals which are rated high by the already mentioned "Straight On" fitness function were selected by the "Elitism" selection mechanism. For more information about that kind of selection see Chapter 3.1.5 - Implemented Selection Functions.

Discussion

The first generations leads to comparable results like in the other experiments. It is possible to get the whole organisms into motion with the help of the evolutionary process.

Pictures of the movement in the second generation are in Figure 4.66 illustrated. Associated angular positions are demonstrated in the Picture 4.67 and 4.68. The development up to the ninth generation can be reconstructed in Figure 4.69 and 4.70.

Note that the order of the shown angular positions have been changed. Because after separation one organism arises that consists of the three robots with identification number one, two and five - see angular positions in Figure 4.67. The other out-formed organisms consists of the two robots three and four- see in Figure 4.68. This is the reason why the respective plots were mapped together.

After separation in the tenth generation the evolutionary process was continued for another ten generations. Looking to Figure 4.71 regrettably one part of the undocked organism tilts to the side. What is a fact, however, that might happen in

Parameter	Value	Description
Evolutionary Algorithm Type	SNN	defines the type of network
Using Phenotype Mapping	1	activates phenotype mapping, note CGE can operate directly on genome
Population Size of Island	10	size of the start population of an island
Number of Input Neurons	5	number of input neurons
Number of Output Neurons	3	number of output neurons
Number of Initial Mutations	5	number of mutations which should be performed to a random start genome
Number of Steps for Evaluation	1000	number of steps a genome will be evaluated before switching to the next one
Number Of Generations	undefined	total number of generations
Number of Exploitation Steps	5	number of exploitation steps to perform before a exploration step follows
Default Learning Rate	0.3	Default learning rate for neuron if genome is generated randomly
Default Mutation Probability	0.3	Default mutation probability for a neuron if genome is generated randomly
Fitness Function Type	STRAIGHTON	defines the type of fitness function to use
SelectionMode	ELITISM	defines the type of selection mechanism
Selection Parameter	20.0	parameter to pass to the selection function
EnableParametricMutation	ON	enables parametric mutation
EnableStructuralMutation	ON	enables structural mutation
NewNeuronToLinkRatio	0.5	0.4 means during structural mutation, 40% probability for new neurons and 60% probability of a new link

Table 4.13: First Run: Parameters for Evolving Locomotion with a Disrupting Organism.

reality. Figure 4.75 (a) indicates that this leads to a complete collapse of the fitness values because it is an absolute different motion pattern. It is comparable to the actuality that a fish which is thrown on land can walk now.

However, in generation 20 the output signal of the artificial neural network of this overbalanced organism is positive rated by the fitness function. An indication that this network is beginning to adjust to the new situation already after ten generations.

On the other hand is the organism consisting of the two remaining robots. This leftover produces already in the 14th generation very good fitness results, see Figure 4.75 (b).

Figure 4.74 shows the movement course during the whole experiment. It is shown how the complete organism moves through the arena (solid line) referred as "organism". After the division the big organism is called "single A" (large dashed line) and the small one "single B" (small dashed line). It could be shown that the small body moves very well which argues for a quick good fitting.

Summary

In total could be shown that it is possible for the implemented approach to amend the motion patterns if there is an unplanned condition or a defective part in the organism. An incredible advantage that was made possible by on-board evolution. Unaffectedly it is not possible to prepare the organism or the single robot for such failures for example with offline evolution on a desktop computer.

Figure 4.66: Second Generation: Evolving Locomotion for a Disrupting Organism.

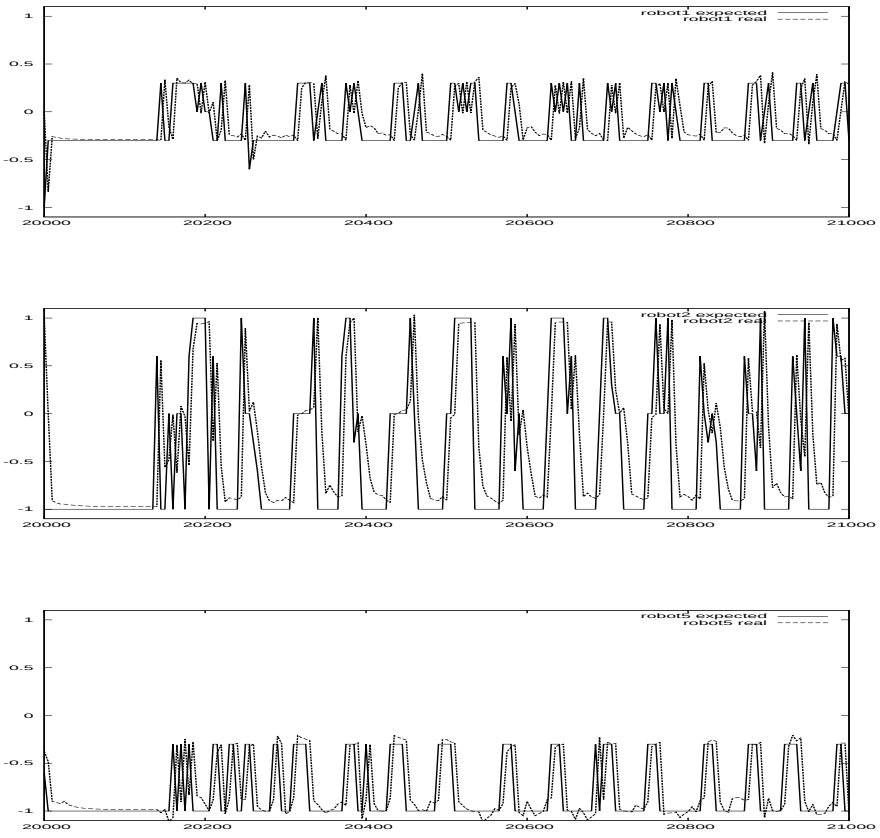


Figure 4.67: Second Generation: Individual Actuator Positions 1 - 3 of the Disrupting Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

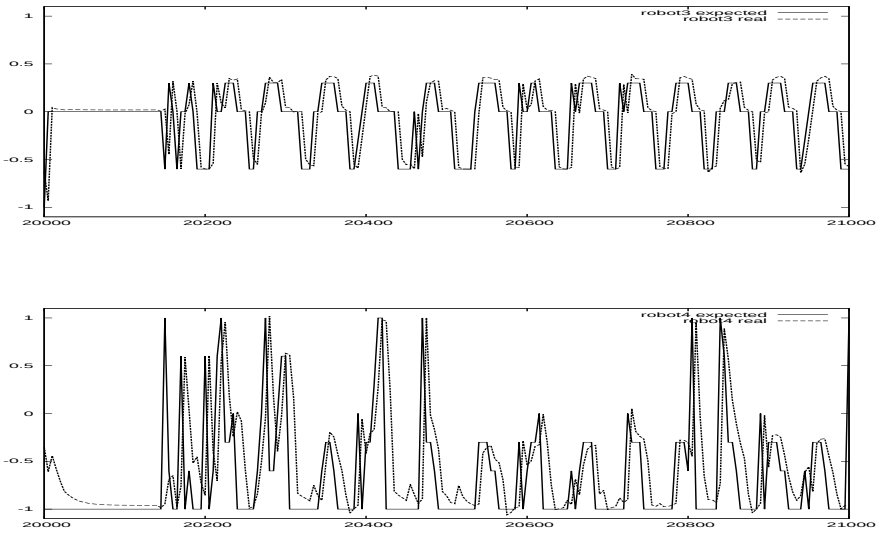


Figure 4.68: Second Generation: Individual Actuator Positions 4 - 5 of the Disrupting Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

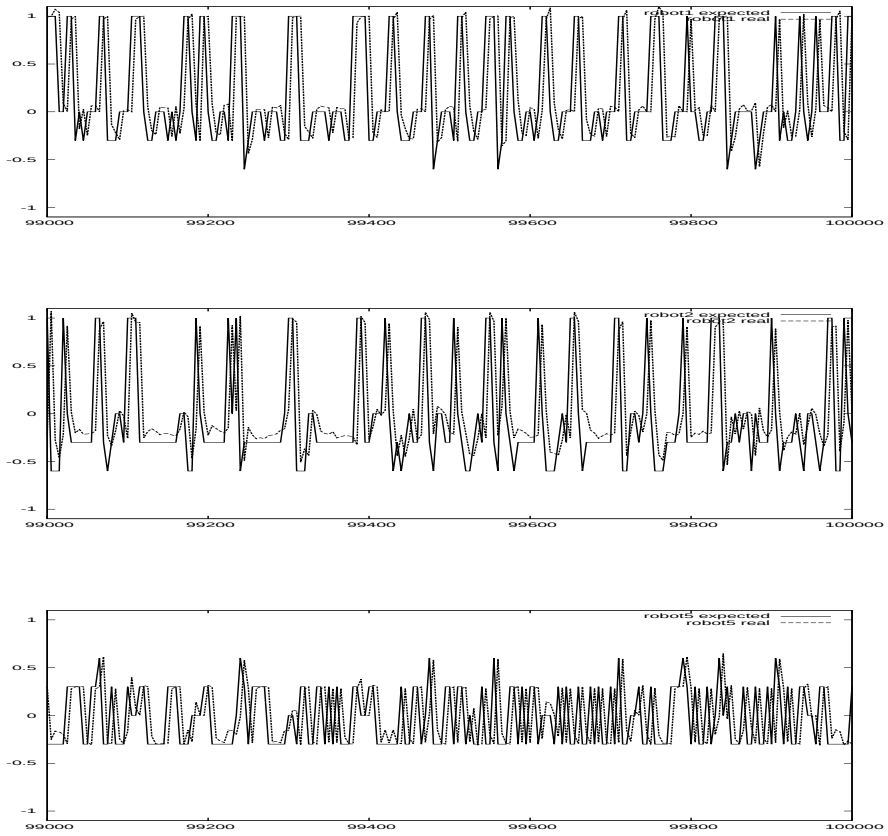


Figure 4.69: Ninth Generation: Individual Actuator Positions 1 - 3 of the Disrupting Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

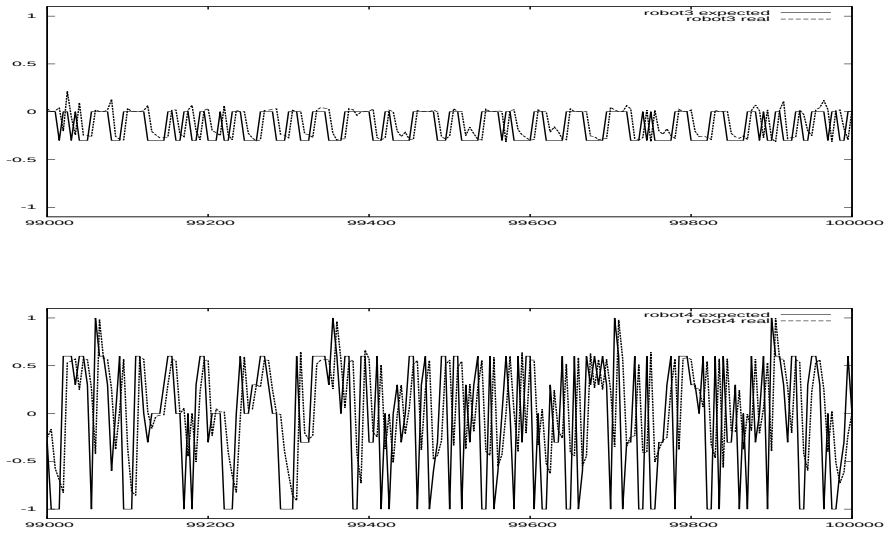


Figure 4.70: Ninth Generation: Individual Actuator Positions 4 - 5 of the Disrupting Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

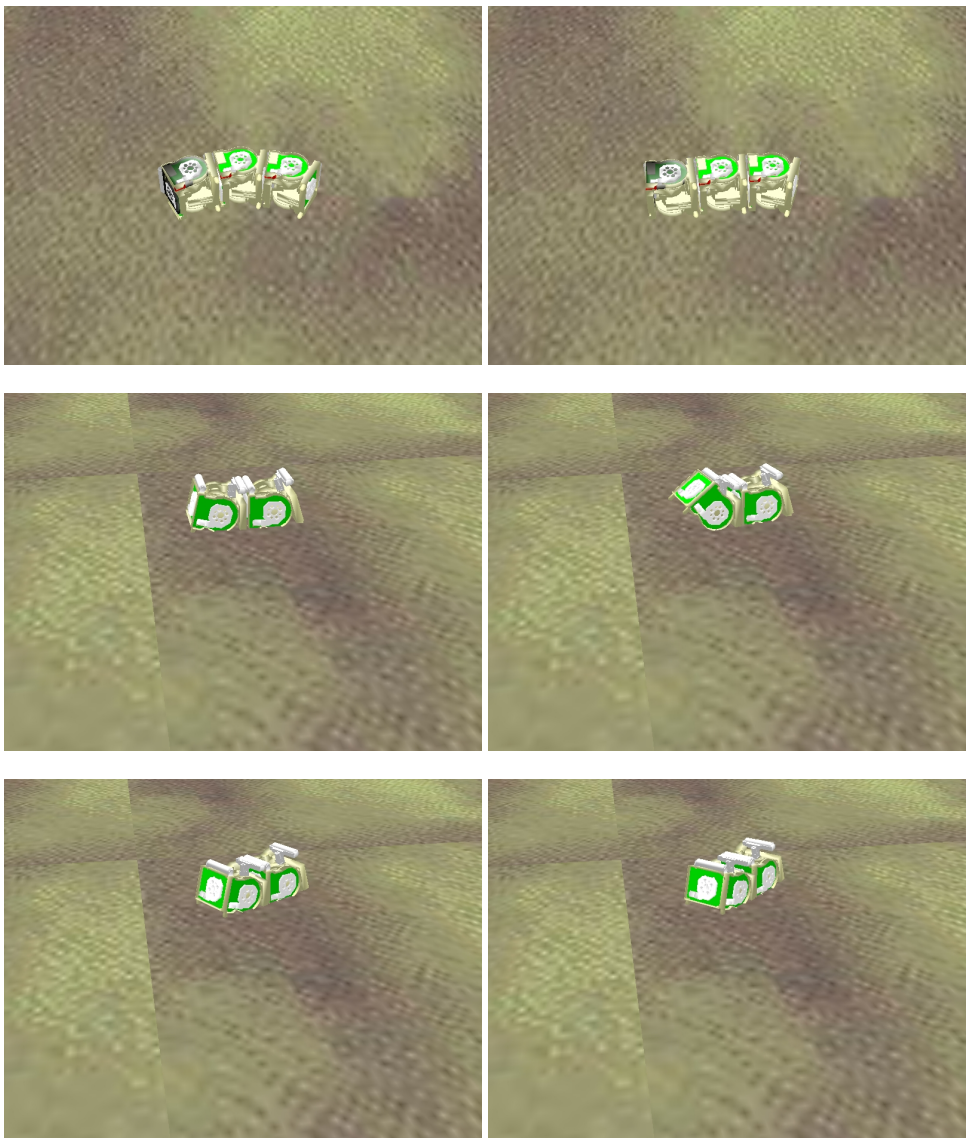


Figure 4.71: 20th Generation: Evolving Locomotion for a Disrupting Organism.

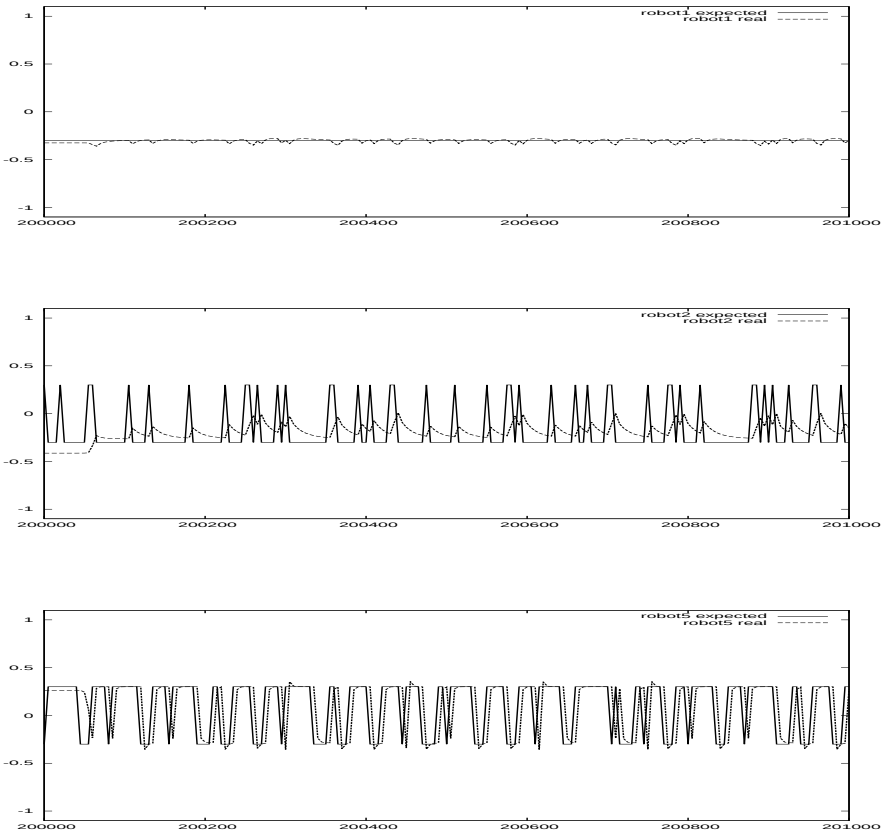


Figure 4.72: 20th Generation: Individual Actuator Positions 1 - 3 of the Disrupting Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

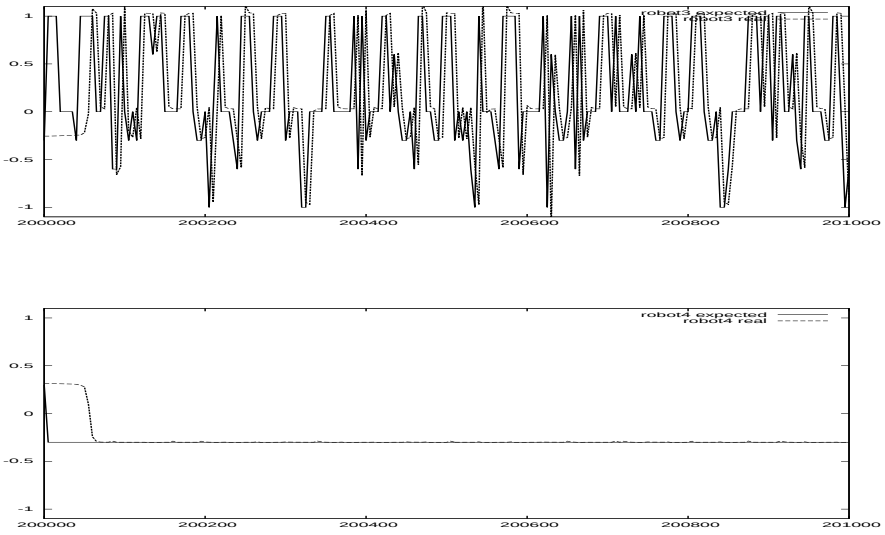


Figure 4.73: 20th Generation: Individual Actuator Positions 4 - 5 of the Disrupting Organism.

X-axis: axis of time with the time ticks [0, +∞].
Y-axis: angle of the hinge element of the individual robot [-1, +1].

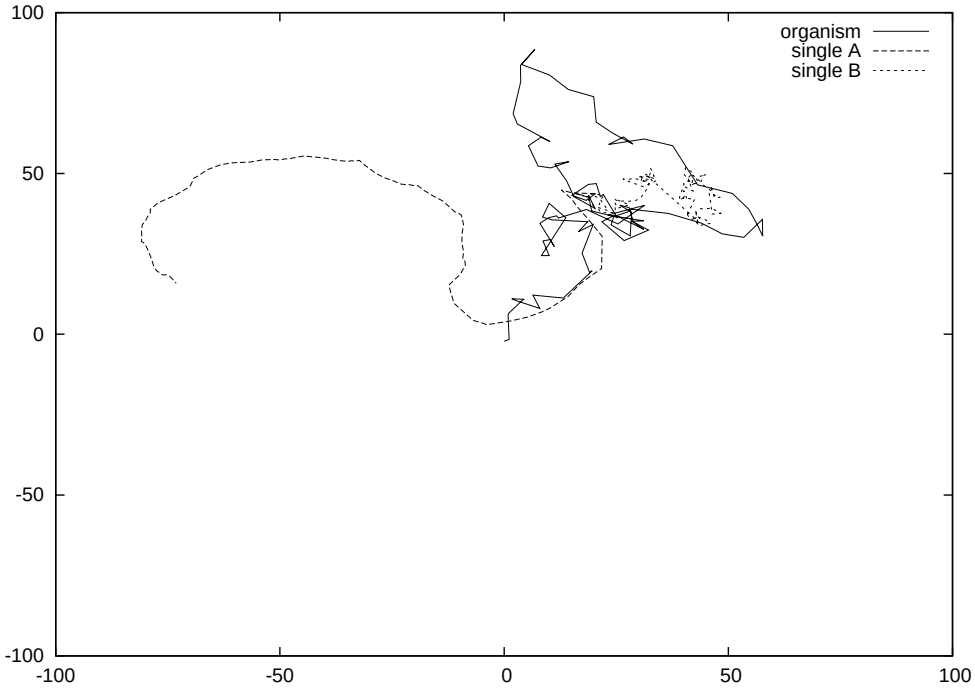
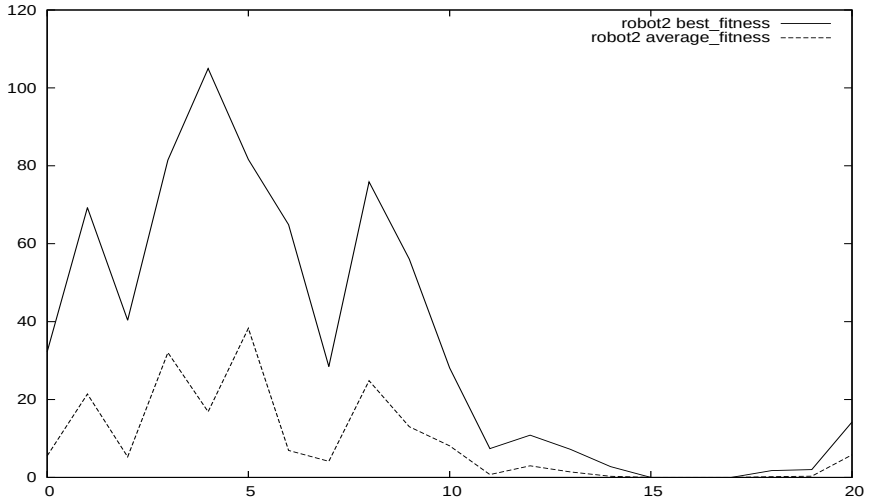
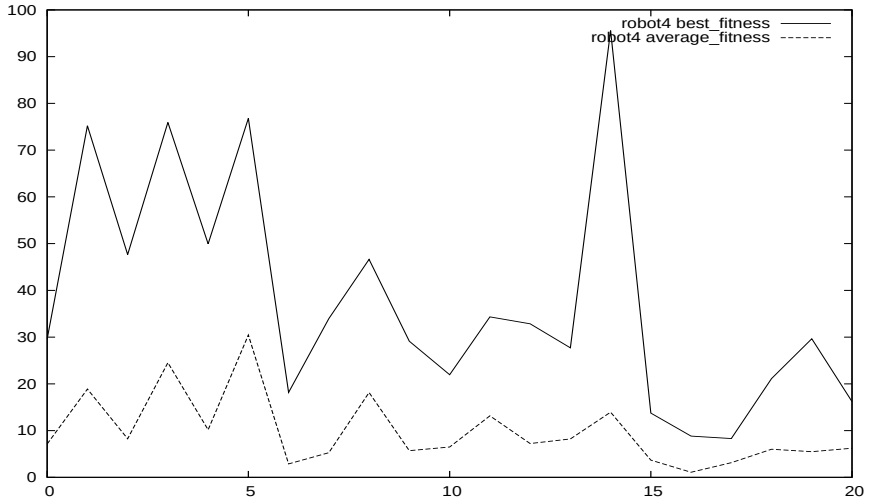


Figure 4.74: Tracking of the disrupting organism on the virtual arena.

X-axis: X direction [-∞, +∞].
Y-axis: Y direction [-∞, +∞].



(a) Organism part A.



(b) Organism part B.

Figure 4.75: Fitness values for the disrupting organism for each generation.

X-axis: generation [0, +∞].
Y-axis: evaluated fitness value [0, +∞].

5 Summary

5.1 Conclusion

In conclusion it can be said that is was possible to generate different output patterns with the help of the implemented artificial neural networks. At the beginning of the experimental part of this thesis could be demonstrated that the described approach is able to produce sinusoidal signals like a sine wave generator. Additional it could be shown that it is also possible to change the frequencies of the signal by varying the input value.

Thus it was shown that this approach is at least as good and is comparable to the approaches that work with sine wave generators.

In a second step this thesis showed that it is possible to generate motion patterns for organism with different shape. It was possible to evolve locomotion already after a short number of generations. This was done with the help of on-board evolution. The artificial neural networks were directly evolved on the robots and it was shown that it is no pre-calculated network required.

It was also shown that none of the robots must take over the control and that a distributed approach is possible. Each of the robot evaluates its own motion pattern with the help of its own set of artificial neural networks. Only neighboring elements receives the angular position of its neighbor via the internal messaging system.

It could be also achieved that individual elements in the whole robot organism take over specific functions for example that of a knee. The implemented fitness function acts thereby only on the individual element. However, each element benefits from the progress of the whole organism.

A big advantage was shown in the last experiment. An organism that comes in trouble because one of its legs is stuck in hole has the opportunity to abandon the affected limb. Old approaches were not able to deal with such danger. In most cases the complete robot was lost. This thesis showed that it is possible to change the shape of the organism and the evolutionary process adapts the motion pattern to the new design.

Thus it can be said in conclusion, that all pre-defined goals have been achieved in this thesis with the implemented approach:

- There is no restriction to the organism shape and no restriction in motion pattern for example only to sinusoidal waves.
- The Artificial Neural Networks are evolutionary adapted with the help of fitness function.
- The Artificial Neural Networks are distributed over the whole organism in each individual robot.
- The Spiking Neural Networks are able to solve different kind of problems with the help of EANT.

5.2 Outlook

On the whole this thesis represents a way to create motion patterns with the help of artificial neural networks consisting of Spiking Neurons.

Furthermore, during the period in which this thesis was written the real robots were under construction but no hardware was available for the experiments. In future work it must be gained certainty about the implemented approach on the real robots.

However it should be noted that in all the experiments the evolution has been terminated prematurely. This means that a continuously ongoing real-time environment like on a real robot has to be tested.

In Addition it was not very easy to compare and evaluate the generated signals. Perhaps a frequency analysis would be appropriate at this point in order to obtain evaluable results.

The question of the proper choice of the parameters could not be clarified in this thesis. This leaves some questions open for future work.

Bibliography

[Bar10] A. Barth. Evolutionary design of central pattern generators for multi-robot. *Institute of Parallel and Distributed Systems - Universität Stuttgart*, 2010.

[BKv09] Zdeněk Buk, Jan Koutník, and Miroslav Šnorek. *NEAT in HyperNEAT Substituted with Genetic Programming*, volume 5495, pages 243–252. Springer, 2009.

[BR] Z. Butler and A. Rizzi. Distributed and cellular robots. pages 911–920.

[Bra86] V. Braitenberg. *Vehicles: experiments in synthetic psychology*. Bradford Books. MIT Press, 1986.

[DS04] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. Bradford Company, Scituate, MA, USA, 2004.

[Eke93] Örjan Ekeberg. A combined neuronal and mechanical model of fish swimming. *Biological Cybernetics*, 69:363–374, 1993. 10.1007/BF00199436.

[ES08] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing (Natural Computing Series)*. Springer, October 2008.

[FM08] Dario Floreano and Claudio Mattiussi. *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. The MIT Press, 2008.

[HRE10] Evert Haasdijk, Andrei A. Rusu, and A. E. Eiben. Hyperneat for locomotion control in modular robots. In *ICES'10*, pages 169–180, 2010.

[HSSC10] H. Hamann, J. Stradner, T. Schmickl, and K. Crailsheim. A hormone-based controller for evolutionary multi-modular robotics: From single modules to gait learning. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1 –8, july 2010.

[IHW99] Auke Jan Ijspeert, John Hallam, and David Willshaw. Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology. *ADAPTIVE BEHAVIOR*, 7(2):151–172, 1999.

[Ijs01] Auke Jan Ijspeert. A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander, 2001.

[Ijs08] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21(4):642–653, 2008.

[Izh03] E.M. Izhikevich. Simple model of spiking neurons. *Neural Networks, IEEE Transactions on*, 14(6):1569 – 1572, nov. 2003.

[Izh04] E.M. Izhikevich. Which model to use for cortical spiking neurons? *Neural Networks, IEEE Transactions on*, 15(5):1063 –1070, sept. 2004.

[KEM+07] Yohannes Kassahun, Mark Edgington, Jan Hendrik Metzen, Gerald Sommer, and Frank Kirchner. A common genetic encoding for both direct and indirect encodings of networks. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 1029–1036, New York, NY, USA, 2007. ACM.

[KKY+03] H Kurokawa, A Kamimura, E Yoshida, K Tomita, S Kokaji, and S Murata. *M-TRAN II: metamorphosis from a four-legged walker to a caterpillar*, volume 3, pages 2454–2459. 2003.

[KKY+05] A Kamimura, H Kurokawa, E Yoshida, S Murata, K Tomita, and S Kokaji. Automatic locomotion design and experiments for a modular robotic system. *IEEEASME Transactions on Mechatronics*, 10(3):314–325, 2005.

[KLM+11] Serge Kernbach, Jens Liedke, Rene Matthias, Florian Schlachter, Christopher Schwarzer, Benjamin Girault, and Patrick Alschbach. Heterogeneity of Autonomous Robotic Modules for the Reliability of a Self-Reconfigurable Multi-Robot Organisms. 2011.

[KMEK09] Yohannes Kassahun, Jan Metzen, Mark Edgington, and Frank Kirchner. Incremental acquisition of neural structures through evolution. In *Design and Control of Intelligent Robotic Systems*, Studies in Computational Intelligence, pages 187–208. Springer, 2009.

[KS05] Yohannes Kassahun and Gerald Sommer. Evolution of neural networks through incremental acquisition of neural structures. *Institut für Informatik und Praktische Mathematik der Christian-Albrechts-Universität zu Kiel*, 2005.

[LFS92] M A Lewis, A H Fagg, and A Solidum. *Genetic programming approach to the construction of a neural network for control of a walking robot*, volume 3, pages 2618–2623. IEEE Comput. Soc. Press, 1992.

[Maa97] W. Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659 – 1671, 1997.

[MI05] D. Marbach and A.J. Ijspeert. Online Optimization of Modular Robot Locomotion. In *Proceedings of the IEEE Int. Conference on Mechatronics and Automation (ICMA 2005)*, pages 248–253, 2005.

[MNSI04] Takeshi Mori, Yutaka Nakamura, Masa-Aki Sato, and Shin Ishii. Reinforcement learning for a cpg-driven biped robot. In *Proceedings of the 19th national conference on Artifical intelligence*, AAAI'04, pages 623–630. AAAI Press, 2004.

[PMG98] D.L. Poole, A.K. Mackworth, and R. Goebel. *Computational intelligence: a logical approach*. Oxford University Press, 1998.

[Rob10] Robot3d, open source modular swarm robot simulation engine. Website, 2010. Available online at `https://launchpad.net/robot3d`; visited on January 4th 2012.

[Sab11] Renato M.E. Sabbatini. Imitation of life: A history of the first robots. `http://www.cerebromente.org.br/n09/historia/turtles_i.htm`, december 2011.

[SAP94] Saunders, Peter J. Angeline, and Jordan B. Pollack. Structural and behavioral evolution of recurrent networks, 1994.

[Zun11] Dominik Zunt. Who did actually invent the word "robot" and what does it mean? `http://capek.misto.cz/english/robot.html`, december 2011.

Declaration

All the work contained within this thesis, except where otherwise acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

(Patrick Alschbach)