AG ROBOTERSYSTEME
FACHBEREICH INFORMATIK
AN DER UNIVERSITÄT KAISERSLAUTERN

Diplomarbeit

Locomotion control of a quadruped robot based on motor primitives

Verena Hamburger

18.04.2005

Locomotion control of a quadruped robot based on motor primitives

Diplomarbeit

Arbeitsgruppe Robotersysteme
Fachbereich Informatik
Universität Kaiserslautern

Verena Hamburger

Tag der Ausgabe : 01.10.2005
Tag der Abgabe : 18.05.2005

Externer Betreuer : Prof. Dr. Rolf Pfeifer
Referent : Prof. Dr. Karsten Berns

Ich erkläre hiermit, die vorliegende Diplomarbeit selbständig verfasst zu haben. Die verwendeten Quellen und Hilfsmittel sind im Text kenntlich gemacht und im Literaturverzeichnis vollständig aufgeführt.

Kaiserslautern, den 18.04.2005

Verena Hamburger

THANKS A LOT

In no particular order:

- My parents for everything,
- Jochen Schäfer for great assistence,
- Fumiya Iida for inspiring talks,
- Rolf Pfeifer for inviting me to Zürich,
- Karsten Berns for letting me go,
- Gerhard Neumann for support with RLT and
- Andreas Bösche for linguistic guidance.

Verena Hamburger

Kaiserslautern, April 2005

Contents

Abstract

One of mankind's prevalent interests is to understand the principles of nature, above all the enormous abilities of learning, adaptation and the wide spectrum of design principles of musculo-skeletal systems. Apparently, the solution cannot be found in just one component, but is distributed over the whole system and arises from their dynamic interplay.

In many cases, humans and animals gain higher behaviours by combining well known lower level components instead of learning from scratch. Additionally, we can find evidence that even some animals entire locomotion repertoire is designed according to this very principle, namely as sequential or linear combination of low level motor primitives. Recent advances also take for granted that all movements are intended as cyclic motion. Needless to say that the use of oscillations makes linear combination straightforward. The interesting issue considering artificial agents is how to generate a good basis of motor primitives. Moreover, we hope that this approach will offer us a little more insight into animal and human behaviour. In this context, we present the first in a series of experiments that serve the development of a quality criterion for the design and analysis of meaningful motor primitives. The first step towards such guidelines elaborates the impact of different vocabularies on behavioural diversity, robustness of pre-learned behaviours and learning process. The case study is a locomotion task of running and having the robot stand up from a lying position. To investigate these ideas, the quadruped robot MiniDog6M is controlled by a simple sinusoidal function for each of its motors. Further, we apply reinforcement learning to train a linear approximator estimating a Q-Function for a variety of motor primitives, each consisting of a set of frequencies.

Keywords: Motor primitive, morphology, behavioural diversity, linear approximator, reinforcement learning, sinusoidal control

1. Introduction

In the course of this introduction, we first motivate the basic principles underlying our research. On that basis we explain about our project and finally give a description of the structure of this thesis.

1.1. Motivation

The overall goal of embodied AI and behaviour-based robotics is to provide insight into animal/human behaviours and facilitate the improvement and development of new skills in robotic systems.

While many animals reach enormous speeds, most of today's mobile robots are capable of slow locomotion only. The reason for this is quite simple: Each movement is static and fully controlled. In contrast, if we have a look at nature (ourselves in the first row), locomotion is anything but fully controlled. Biped walking, for instance, more or less resembles forward tumbling. Gravity takes a major role in walking. Many more examples can be found where physical interaction either within our body or between our body and the environment substitutes active control. One example is the existence of natural rest positions in pairs of counteracting muscles. This posture is especially energy efficient and it can effortlessly be reached from every other posture just by letting go. Further, we notice that the elastic properties of our hands and feet help us to passively adapt to uneven surfaces. These are just two examples that everybody can experience in his/her own body. This principle of exploiting the givens is applied in the modern approaches to artificial intelligence, the so called new AI, embodied AI or behaviour-based AI.

Operating in natural environments, it is absolutely essential for mobile robots to enhance adaptability. Flexibility is crucial. Drawing inspiration from nature as a first class designer of adaptive beings, we find that humans often gather new abilities not from scratch, which means acquiring new motions (or ideas) without a priori knowledge, but rather we gain highly sophisticated abilities by combining well known lower level components. Therefore, it is reasonable to assume a hierarchical behaviour representation where abstract

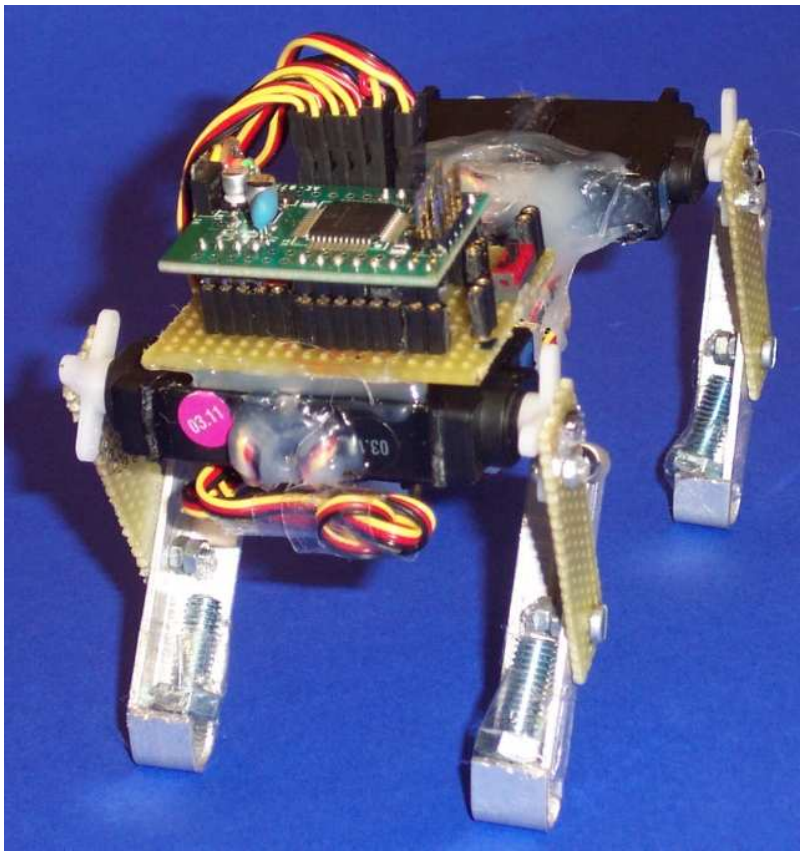


Figure 1.1.: MiniDog6M

behaviours are represented as a sequence of lower level movement primitives. While higher levels plan complex behaviours such as standing up, the detailed motor scheme is accomplished in lower levels. Evidence for this distribution can be found in various animals, but as well in the human spinal cord which takes a major role in complex movement generation [Tani 02, Bizzi 84, Feldman 80].

One of the best explored examples of movement primitives can be found in the nervous system of the frog [Giszter 93, Mussa-Ivaldi 94]. When special points of frog's spine are stimulated with electrodes, its legs automatically perform a fixed behaviour e.g. wiping. Careful studies found only few of these point, which lead to the assumption that about a dozen of those primitives are enough to produce a frog's entire movement repertoire by means of sequence and superposition.

Applying this principle to robotic environment, these movement primitives represent modules that are repeatedly found in complex sequences of motor patterns. In contrast to other approaches that rely on adaptive basic behaviours, the example of the frog suggests fixed primitives. They ease learning in more than one way. On one hand, the learning process is accelerated, since the agent acquires new behaviours as composite temporal ordered combinations of low level primitives. This significantly reduces the search space of possible postures and trajectories. On the other hand, relying on a set of basic behaviours not only simplifies the generation of movement, but also facilitates its perception. The existence of sensory-motor-integration in mirror neurons, which are equally active when-

ever a motion is observed or executed, proves the theory that behaviour primitives are simultaneously used to recognise and plan motions. Thus learning through imitation is eased [Schaal 99, Kuniyoshi 94]because the robot identifies well known patterns in new temporal order instead of a complete series of angles for each motor. Moreover, when confronted with an incomplete sensory input, prediction is motivated through constant classification of observed movements into its known repertoire. The latter is approved by developmental psychology providing evidence for goal prediction in infants while observing incomplete or incorrect actions.

In addition, motor primitives are a simple and effective approach to solving the degree-of-freedom problem. Since the Russian scientist Nicolai Bernstein was the first who formalised the difficulties arising from a physical body having more degrees of freedom than the actual task, this is also known as Bernstein problem [Rosenbaum 96]. A plain example to illustrate this issue is the position and orientation of an object in space compared to the degrees of freedom of a human operator trying to grasp it. Whereas we need only six descriptors to determine the position and orientation of an 3D object, there are infinite possibilities for us to approach our target. The reason for that is that the number of degrees of freedom of the human body is larger than the number of task descriptors. Moreover, there is an exponential increase in the state space and thus in the number of actions that can be generated in a movement system with many degrees of freedom. Bernstein also emphasised that the state space is often smaller than suggested from the bare number of joints or effectors, since morphology restricts the actual number of possible postures.

Regardless of the method used to acquire the skill, it is widely believed that all sorts of movements seem to be designed as cyclic motions. Analysis of animal locomotion suggests that these motions are generated by neural networks which are capable of generating basic rhythmic motor activity. Recent advances found in a variety of legged locomotor systems, for example, in the mud puppy (Heterocephalus glaber), the turtle (Testudinata), the cat (Felix felix) and the stick insect (Carausius morosus), that the central network can be decomposed into multiple lower level generators, each controlling a subunit such as joint, segment, or muscle of the locomotor system [Büschges 05]. Thus complex joint angle trajectories are generated by composing oscillatory movements with lower complexity. To simplify matters locomotion is often the result of superposition of different cycle frequencies of the different muscles.

All these ideas and concepts motivated us to investigate the impact of cyclic motor primitives and morphology on locomotion control for a quadruped robot. The assignments of this project are described in the next section.

1.2. Project description

This project aims at investigating the impact of motor primitives and morphology on locomotion control for a quadruped robot.

The theory of having a basic set of motor primitives which can be composed into a broad and general movement repertoire has served as inspiration for behaviour-based control and robotic models. Their development is one of the most recent trends in the field. While it

is obvious that the introduction of powerful, adaptive motor primitives (and thus avoiding online trajectory planning) is an appealing organizational principle, three questions remain open:

1. What should the primitives in such a basic set be like?
2. How can those discrete entities be derived from morphological properties?
3. How can generic primitives obtain enough information for complex adaptive behaviours?

The latter can be answered quickly pointing out that the representation of a complex function as a linear combination of much simpler functions is a well established theory in mathematics and physics. Parts of the other questions will be elaborated in the course of this thesis by investigating the influence of morphological constraints to motor control. Combining the issues mentioned in section 1.1, we create a methodology that helps to derive several design principles for meaningful motor primitives.

Behaving in natural environments with all its' disturbances mobile robots may tumble and fall from time to time. In order to fulfil its task nevertheless, the robot needs to first recognise the mere fact that it fell and consequently be able to stand up. Therefore, our case study is a locomotion task involving standing up.

The hardware platform underlying my research is a quadruped robot developed by Fumiya Iida at the AILab of the University of Zürich. Following the principles of embodied AI, the so-called MiniDog6M, which can be seen in Figure 1.1 on page 4, is able to move quickly by employing plain hopping as found in nature. This capability is obtained by optimised design which means that most of the control is compensated by exploiting some simple physics, such as the resilient properties of a spring yielding a passive degree of freedom in each leg.

The robot dog will be toppled by a random force applied to its head while running. Then MiniDog6M shall get up and carry on its way. Contrary to approaches that rely on human interference, this thesis tries to enable MiniDog6M to free itself out of this situation. This assignment is accomplished only by few robots - especially not among quadrupeds.

The first step in a row of experiments leading towards such guidelines examines the behavioural diversity of different vocabularies and morphologies and the validity of those behaviours in environments with slopes. The second investigates the impact on the learning progress.

The next section give a short description of contents of our work.

1.3. Structure and description of contents

In this section, we roughly characterise the main points of each chapter of this thesis.

After this introduction, in chapter two, the concepts underlying our research will be elaborated. First, we will give a few examples where physical interaction between the body and the environment substitutes active control. Aiming at a cost-efficient fast locomotion, researchers engaged in passive running. We will shortly recapitulate the quadruped model. Then, we will introduce the morphological and control concepts of the "running dog project" which MiniDog6M is part of. Subsequently, we introduce a framework for the investigation of morphological implications on evolved behaviour.

After that we will give a quick overview of the state of the art. First, we will introduce Kenken and Scout 2 that serve to investigate compliant running with springy legs. Then we explain two successful strategies for oscillatory movement that are embodied in Tekken II and BISAM. For the latter, oscillatory movement is only partially fitting, since, its control strategy later changed towards a reactive architecture. The reasons for this change will also be presented. Finally, the terminology and representation of an assortment of projects using motor primitives is overviewed.

At the end of this chapter, we will elaborate the entitlements to this thesis which result from the concepts and projects presented here.

In chapter three, we will introduce the morphological and control concepts of our research platform and subsequently provide a description of our experiments in the real world which build the basis of our further studies. In these primary experiments with the physical robot dog, several gaits and a pre-programmed standing up motions are captured.

In chapter four, the first set of motor primitives will be extracted out of these pre-programmed sequences. Each primitive involves one or more motors. Further we will establish a general means to evaluate them in regard of the frequencies they assign to the motors. On that basis, we will work out several vocabularies, each of which being an assortment of motor primitives that will be investigated throughout this thesis.

In the second stage, starting with chapter five, a virtual model of MiniDog6M will be created to enable further experiments with different morphologies which cannot be changed effortlessly in the real world. Its implementation as well as the simulation platform will be described in this chapter. In doing so, we will introduce the public library Open Dynamics Engine (ODE), which is used for physically realistic simulation. The model of MiniDog6M and its controller will be specified afterwards. Finally, the morphologies that will be investigated in the course of this thesis will be selected.

In order to quickly overview the behavioural diversity, the standing up sequences in chapter six will be generated as trial and error combination of the underlying motor primitives. This chapter describes the first row of experiments for the evaluation of the vocabularies selected above. First, we establish a general means to compare the behavioural diversity of different tasks or vocabularies. To get an intuition for the variety of legal standing up sequences and to investigate how the shape of the head affects the behavioural diversity, we will run a full search simulation. Performing the same full search in inclined environment we will be able to overview how such changes affect the nature and amount of solutions. This second set of experiments addresses an important characteristic of robot control: robustness.

Then, in chapter seven, we engage in learning, since trial and error at random is generally not a good strategy for an agent to behave in natural environment. To get along in new situation, structured search and learning is much more suitable. Therefore the support of the learning progress is an important feature of a good basic vocabulary. In this chapter, the design of the learning environment and the experimental setup is described. We accommodate an artificial Radial Basis Function network for Q-learning with the different combinations of head and vocabulary. The Reinforcement Learning Toolbox is used to perform the learning process of this linear approximator. Finally, the learning progress of each vocabulary in dependency of the shape of the head will be evaluated.

At the end, chapter eight will sum up the main points of our work and the results gained from our experiments. Together with these concluding remarks, we will further give an outlook on future assignments that directly hook up on the framework presented in this thesis.

In the appendix we provide a glossary, some additional data on the behavioural diversity and transfer capabilities gained from our experiments with the simulated MiniDog6M. Moreover we provide a set of training patterns for supervised learning derived from the latter mentioned data set.

2. Challenges in legged locomotion

In this chapter, modern principles of robot design, legged locomotion, energy efficient fast running and a framework to systematically investigate the influence of morphology on evolved behaviours are presented. Further, an assortment of related projects is summerised. Finally, the resultant entitlement to this thesis is outlined.

2.1. Importance of morphology

In this section, the reasons why it is important to stress morphology in a robot's design process are given. Usually, morphology is used in the context of animal physiology. In the broader sense, we copy this term into technical environments meaning the shape, material, choice and placement of sensors and actuators of a robot [Pfeifer 99].

In the traditional approach to robotics, designers usually predetermine the morphology of the desired robot and afterwards design a controller according to the given mechanical design. Further performance enhancements must thus be achieved through improvements in control. This methodology has on one hand led to many successful and famous examples of biped walking such as the Honda humanoid series, but on the other hand left unexplored the numerous fields that take advantage of optimised morphology and its intrinsic dynamics. Recent advances in embodied AI proved the great influence of mechanical structure, sensor and actuator placements on the performance of a robot.

Talking about the importance of morphology in locomotion, we come across the passive dynamic walker [Pfeifer 03b, McGeer 90a, McGeer 90b] which can be seen in Figure 2.1 on the next page. This extreme example of parsimonious robot design is capable of walking down a slope without active control. Hence, the walker is rather a mechanical device than a robot (at least not in the common sense of a robot), since it relies purely on the body dynamics of swinging legs and arms. As a result it is tied to the small ecological niche of medium inclines with flat surfaces. Though there is no actuation but gravity, its walking looks very humanlike. However, for the robot to be applicable in natural environment, actuation and consequently control would be a necessity. Nevertheless, it is a great example of cheap design. Cheap in this connotation refers to parsimonious robot design

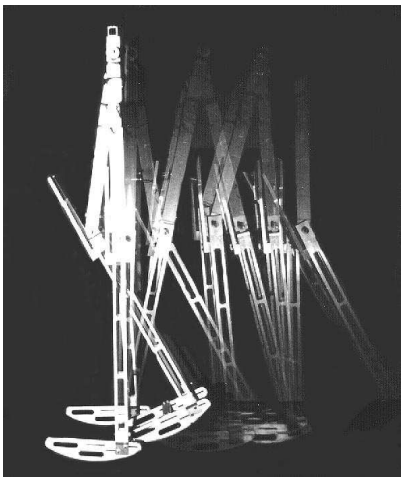


Figure 2.1.: Passive dynamic walker
A mechanical device that is capable of walking down a slope with no actuation but gravity.

which exploits the physics of system-environment-interaction as well as the constraints of the ecological niche. In doing so, the system's inherent dynamics achieved by proper design partly substitute active control.

Paul and Bongard [Paul 01] for the first time optimised morphology simultaneously with a closed loop controller in a single process to achieve stable biped walking. Being one of the first mechanical design decisions, they addressed the problem of mass distribution along the biped skeleton in terms of positioning motors and gears, which usually supply the heaviest components in a robot.

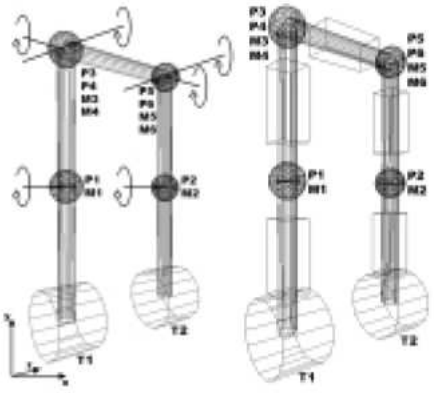


Figure 2.2.: Simulated biped construction
This skeleton with six degrees of freedom is shown with (right) and without (left) mass blocks attached to it. The position and the geometrical dimensions of the discrete blocks is evolved together with a closed loop controller to investigate the simultaneous optimisation of morphology and control.

The simulated robot, which can be seen in Figure 2.2 on the facing page, has a waist and two legs consisting of an upper and a lower part. The model has jointly six degrees of freedom. It contains two haptic sensors in the feet and a proprioceptive sensor in each joint. The joints are driven by torsional actuators limited with ranges of motion closely resembling those of human walking. Because of its intrinsic capability of producing cyclic dynamics, the controller was desinged as recurrent neural network. The aim was the development of stable gaits by evolving an optimal controller together with the position and the geometrical dimensions of discrete blocks and herewith study the effect of changing mass distributions on the robot's dynamics.

Three rows of experiments were investigated dealing with minor, mid range and major reallocation of weights. Interestingly, the more stable gaits were achieved, the higher the allowed degree of weight shifting. These results suggest that morphological changes can indeed optimise performance.

Another interesting example is the Eyebot [Lichtensteiger 00, Pfeifer 05], which can be seen in Figure 2.3 on the next page. This project supplies evidence that the computational effort can be drastically reduced by optimised sensor placement. An evolved sensor arrangement of artificial eye facets (here: light sensitive cells) automatically brings forward a design where the optical sensors are more dense towards the front. Thus a navigation task is significantly eased by compensating the phenomenon of motion parallax[1]. This phenomenon can also be found in the compound eye of the housefly.

Further, we already mentioned the natural rest positions in muscles and the self-adaptation of our hand and feet.

The robustness, ease and flexibility of these solutions give an idea about the advantage of exploiting body dynamics. Many different solutions can emerge naturally. Emergence here indicates that behaviours which are not explicitly specified in the robot program, come up as a result of agent-environment-interaction [Pfeifer 99, Pfeifer 03b]. This distribution of work is visualised in Figure 2.4 on page 13.

One of the advantages of the emerging of behaviours is the evasion of the symbol-grounding problem. Traditionally AI works with internal symbol and their relations, but these symbols are not grounded in the system's understanding and interaction with its environment. A human user or developer automatically maps the symbols to the representing objects. The relations between internal representations and their physical correspondents as well as the possible resulting interactions are founded in our experience. We see a car, we know that it's the object's name is "car" and that we can drive away with it if we own the keys. If the concept "car" is already known, the mapping between the concept and its properties can easily be achieved by a computer, whereas the first problem, namely the identification of an object on the basis of given sensor input, is a really difficult problem. Neural networks by themselves are not capable of resolving it and thus start from designer-defined high-level ontologies. Consequently, gaining suitable sensor data through embedding the sensor(s) appropriately in the agent's architecture is of

[1] The phenomenon of motion parallax can easily be experienced by looking out of a moving vehicle. Objects that are closer to the observer seem to move much faster than the ones that are far away.

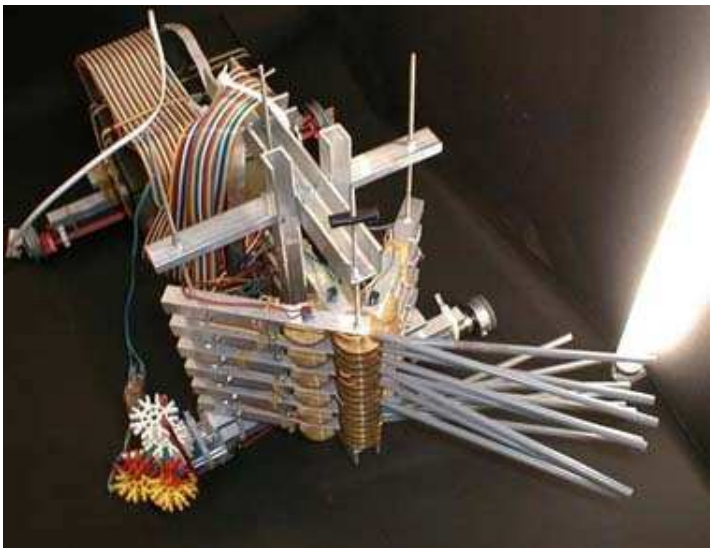


Figure 2.3.: Eyebot
Adaptive arrangements of light sensitive significantly reduces computational effort by compensating the phenomenon of motion parallax

great importance. Being useless in classical computer science because it lacks system-environment interaction, it is a central challenge to robotics [Pfeifer 99, Pfeifer 03a].

Talking about symbol grounding, yet another difficult problem has to be taken into account. The object constancy problem addresses the fact that one object leads to set of very unlike sensory patterns depending on viewing point and surroundings. Categorisation on the basis of this almost infinite search area is truly a lot of hard work. The approach of embodied cognitive science reduces the complexity of this task by designing an autonomous agent that learns its concepts through active interaction with its environment - not only through observing the world passively. The goal of these advances is the active creation of well directed sensory inputs using sensory-motor coordination and thereby reducing the input space. This attempt resembles natural human behaviour. It can be compared with a human child that turns an object in front of its eyes at a fixed distance and then bites into it. This behaviour structures the input and by this means induces regularities that significantly simplify category learning, an important precondition for intelligence. Apparently finding the proper morphology and above all sensor position is fundamental for generating stable input with sensory-motor coordination [Pfeifer 99, Pfeifer 03a].

The lesson that can be learned out of these examples is that not everything must be controlled by the brain respectively the robot program. Physical interaction either within our body or between our body and the environment can often ease or even substitute active control. The consideration of a morphological change can often be much more efficient than improvement of the controller. This distribution of computational and control functions between controller and morphology and environment is called morphological computation.

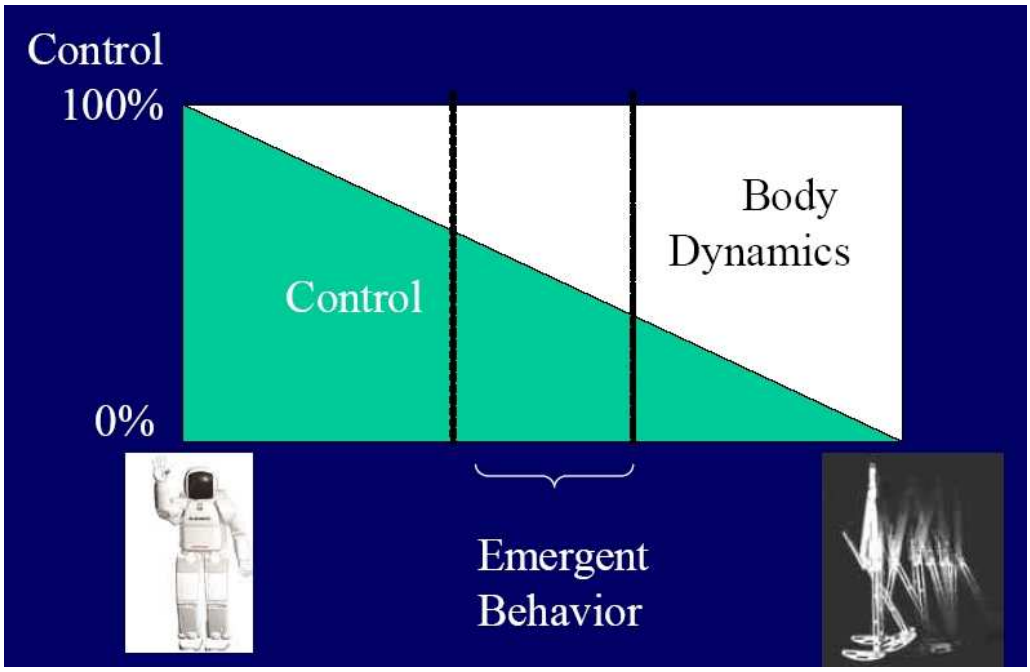


Figure 2.4.: Division of control in emergent behaviours
Emergence means that behaviours are not explicitly specified in the robot's program, but come up as a result of agent-environment-interaction instead.

The next section is on general and particular approaches to legged locomotion and passive running.

2.2. Approaches to legged locomotion

In this section, general concepts of legged locomotion are dealt with. Further a passive quadruped runner is presented and the "running dog project", which our experimental platform is part of, is introduced.

2.2.1. General concepts

Recent entitlements to increasing range of robotic applications have led researchers to the limits of wheeled and tracked robots. For application on irregular terrain, adaptable locomotion machines with many degrees of freedom have raised attention. Unfortunately, adaptability comes at the cost of complex control and low energy efficiency. As it is essential for (power) autonomous mobile robots to reduce their power consumption and at the same time maximise the utilisation of their operational time, a lot of effort has to be spent on energy efficient control of fast running. One of the biggest problems in fast locomotion is the extremely short response time for the sensory feedback control loops that are usually employed in walking robots.

Breaking up locomotion into a series of steps, each step passes through the same phases. Statically stable walking on one hand needs first to release a leg, then to swing it towards the desired position and finally to stabilise it again. Dynamically stable running, on the

other hand, self-stabilisation achieves through proper body dynamics and thus lacks the stabilisation phase. First, in the stance phase, all legs support the body weight, whereas in the flight phase, the legs are significantly released or even lifted off the ground. Examples in which these phases can clearly be discriminated will be presented in subsequent sections.

Regarding the biological background (as already introduced in chapter 1), the ongoing debate whether periodic movement is based on either reflexes (as in the frog) or neuro-oscillators (as in the stick insect) seems to conclude with both in collaboration [Luksch 02].

Here 'reflex' means a goal-oriented behaviour tightly coupled to the strength and type of sensor stimuli. This sort of behaviour is initiated by vegetative or motor processes. A neuro-oscillator is a neural network that, irrespective of the current sensor state, produces rhythmic impulses each of which kicks off a motor action. As a consequence, reflexes are much more situated and adaptive than neuro-oscillators. Robotic experiments [Ferrell 95] compared three different control strategies found in insects by implementing them on a hexapod robot. The results showed that CPG performs much better than purely reflexive approaches. Recent advances revealed mutual influence on both schemes via sensory-motor-coordination. While CPG dominate in general, reflexes take over when dealing with disturbances. In doing so, the activity of single muscles or groups of muscles are modulated and coordinated e.g. at spinal cord and brain stem,

Technically speaking, methods for legged locomotion control can be classified into zero moment point based control (ZMP) and limit-cycle-based control [Luksch 02].

ZMP is the "extension" of the centre of gravity considering inertia force which means that the centre of mass must always be above the bearing area of the body. Consequently, each motion is statically stable. As the whole body motion must be considered, ZMP is mainly controlled by an upper neural system. From the standpoint of energy consumption this approach is effective only for posture control and slow walking, since with every step the large body mass must be accelerated and decelerated by actuators. Moreover, this statically stable pace can lead to deadlock situations, where it is impossible to lift a leg because this would lead to an unstable situation. These problems must be considered in advance and therefore require complex planning mechanisms.

Superior energy efficiency and dynamic stability is achieved by limit-cycle-based control. The term limit- cycle refers to the fact that the motions in time plane form a stable limit cycle on the phase plane. This stability is achieved by alternating support of the legs. A typical example for limit cycle based control is the passive dynamic walker, which was already mentioned aforement. This category can be divided further into control by lower neural systems (CPG and reflexes) and mechanic control by a spring-damper-system. Regrettably, the first subcategory is appropriate for not more than medium-speed. The second concept of compliant legs is realised in the "running dog project" and accomplishes high-speed running through self-stabilisation.

Contradictory to statically stable machines, dynamic locomotion with compliant legs supports higher speed and drastically improves mobility enmeshed in simplified mechanics.

As a penalty, discontinuous storage and release of energy in the passive leg compliance is needed and, as a consequence, it is not longer possible to control the body motion directly. Pioneering work in the field of effective uncontrolled running behaviour was accomplished at MIT's LegLab in Boston. Raibert et al [Poulskakis 05, Raibert 85] engaged in springy legs of telescopic form and realised monoped, biped, and quadruped robots capable of various gaits. They found that such quadrupeds, do not need active posture control in bounding gait as long as their body's momentum of inertia is smaller than its mass times the square of the hip spacing. Moreover, They revealed passive trot, gallop or bound is possible in both stance and flight phase if the system is provided with the proper initial conditions.

Meanwhile, even legs with adjustable stiffness were suggested in order to adapt to different surfaces [Ferris 98].

2.2.2. Passive running in quadrupeds

Aiming at a cost-efficient way of fast locomotion, passive running on robots with one, two or four legs were investigated after the paradigm of the passive dynamic walker. Let us briefly look at the quadruped version [PassiveRunning 05].

The assembly can be seen in Figure 2.5. In place of active actuation, three springs are attached at each leg. Two of them symetrically connect the leg to the body, one on the left and one on the right. The third attaches the foot to the lower end of the leg. This design saves the leg swinging energy. In doing so, each leg has two passive degrees of freedom: a rotational and a linear one.

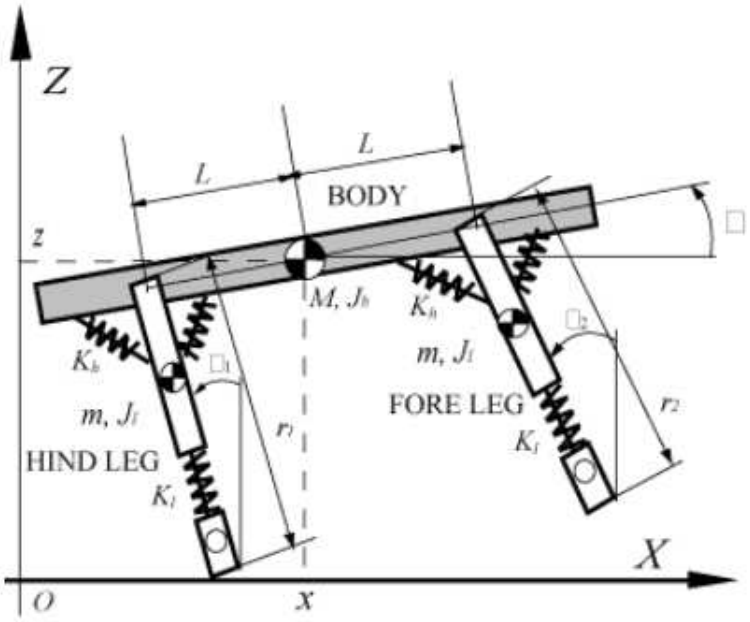


Figure 2.5.: Schematic of the passive quadruped runner
This design saves the leg swinging energy. Each leg has two passive degrees of freedom: a rotational and a linear one.

Creating a simulation, the leg mass is taken into account. Thus this approach covers not only the body's oscillatory pitch motion, but also reveals the swinging properties of the leg. Since there is no specific running sequence designated, the propagation of learning follows no specific order. Each step can be divided into four phases triggered by touchdown or lift-off of front respectively hind legs which move in parallel. The transition between these phases can be seen in Figure 2.6. In double support, which is always the initial phase when a robot starts to move, all legs stand on the ground and support the body weight. Performing a regular hopping behaviour, the forelegs should be released and lifted off the ground, which would bring the robot in hind leg stance phase. After pushing to robot forwards, the hind legs are automatically released which brings the robot in flight phase. Next, in fore leg stance, the forelegs have reached the ground and start take on the body weight. The initial phase is reached as soon as the weight is once again supported by all legs. Depending on the particular gait, the phases may come up in alternating order.

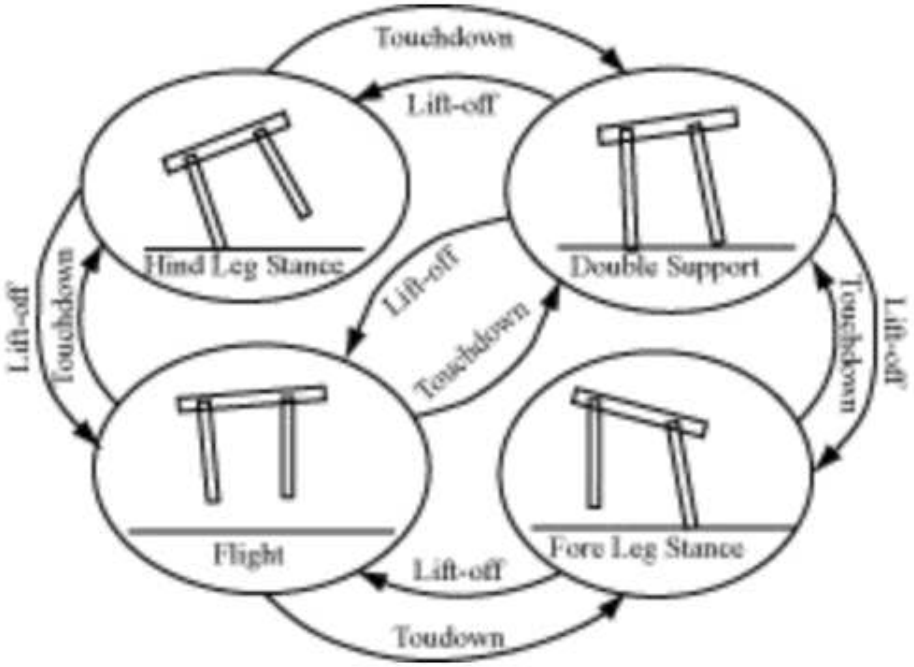


Figure 2.6.: Phases of passive running gaits
All resulting gaits have proven to be symmetrical, but unstable.

All passive running gaits found during the analysis of the simulation are symmetrical. Unfortunately, uncontrolled running is secure for not more than 25 steps. By the 26th step, the quadruped is no longer upright, but has either canted over the forelegs (during fore leg stance) or toppled over the hind (during hind leg stance). Thus all resulting gaits have proven to be unstable. Actuation is needed to keep a quadruped upright.

2.2.3. The "running dog project"

The challenge to biomimetic design aims at reproducing the number of passive joints, dimensions of limbs, weight, properties and locations of muscles of natural systems.

Aiming at a minimalistic, but still biologically plausible design for fast locomotion, the "running dog project" was founded by Fumiya Iida in order to investigate a big variety of morphologies. All its members, amongst others our research platform MiniDog6M, comply to the same morphological and control principles [Iida 03, RunningDog 05].

2.2.3.1. Morphological concepts

Looking at nature, we notice that limbs comprise pairs of counteracting muscles (antagonistic principle), but that they also have natural rest positions. As a substitute, many technical systems contain springs as artificial muscles. Springs qualitatively approximate several natural properties of the muscle-tendon system, for example unidirectional actuation, multiple passive joints moved by a single motor and nonlinear torque depending on the angle of the passive joints. In addition, they are cheap, widespread and available in many different variants such as size, spring constant and material.

One of those biologically inspired projects using springs is the quadruped Geoff, which can be seen in Figure 2.7 and which was designed after anatomical studies of a canines musculoskeletal system.

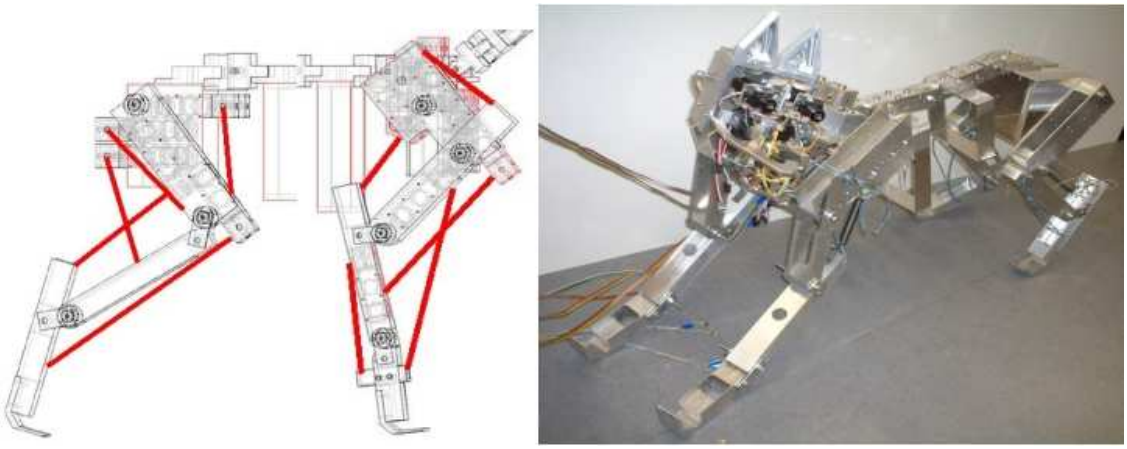


Figure 2.7.: Geoff in photo (right) and schematic (left).
Geoff was designed after anatomical studies of a canine's musculoskeletal system and is the first member of the "running dog project"

Its skeleton is approximately 750 mm long, 300 mm wide, 600 mm tall and made of aluminium. Geoff contains 28 rotational joints with one passive degree of freedom. The rotation angles are mechanically restricted and capable of small translational displacement. A binocular active vision system with four servo-motors and two miniature CMOS cameras are installed in the robot's head. The red lines in Figure 2.7 depict the position of the springs.

Geoff is the first in a series of quadrupeds articulating the "running dog project". Thus Geoff had many descendents that concentrate mainly on the design of adequate legs. The underlying canine leg design is qualitatively shown in Figure 2.8 on the next page. *Gastrocnemius Planataris* and *Soleus* are muscles connecting the heel to thigh and shank. The corresponding antagonists are not shown.

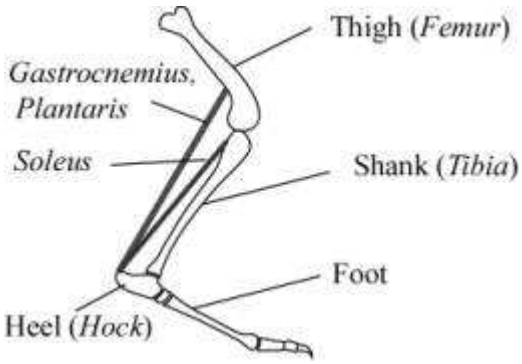


Figure 2.8.: Schematic of a dog leg
Gastrocnemius Planataris and *Soleus* are pairs of counteracting muscles (antagonistic principle).

The one robot, that embodies this leg model most closely, is Puppy which is shown in Figure 2.9. Each of its legs is designed identically with independent servomotors in hip respectively shoulder and knee. The motors are marked with an encircled cross in the schematic. Two springs in each leg connect the lower part of the leg (heel) to the upper (thigh) and the middle part (shank) and thus substituting *Gastrocnemius Planataris* and *Soleus* in Figure 2.8. So far the motors in the knees are fixed. Further, a flexible spine is used in order to support far jumps and thus fast running. As we can see, this is a great example of cheap design.

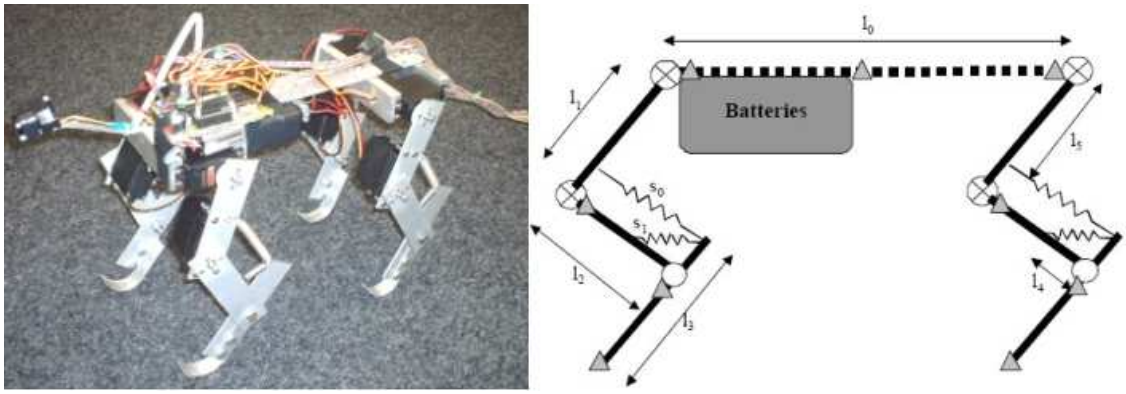


Figure 2.9.: Puppy in photo (left) and schematic (right).
Its legs are designed after the schematic of a canine's leg in Figure 2.8. Its passive flexible spine supports far jumps and fast running.

After Puppy, many other quadruped were developed. Each is unique in its design, size and material, but constructed according to the same principles. We show just a few example in Figure 2.10 on the facing page without going into detail.

The most remarkable point is that the one on the left hand side gets even along without any spring at all. The springy property is provided by the elastic material of the legs. The robot in the middle is MiniDog, the closest relative of MiniDog6M. The new aspect in

MiniDog6M is the active spine that enables the agent to twist and bend. The exemplar on the right has the identical leg design as MiniDog, but is realised with different material, leg diameter and relatively sharp ends.

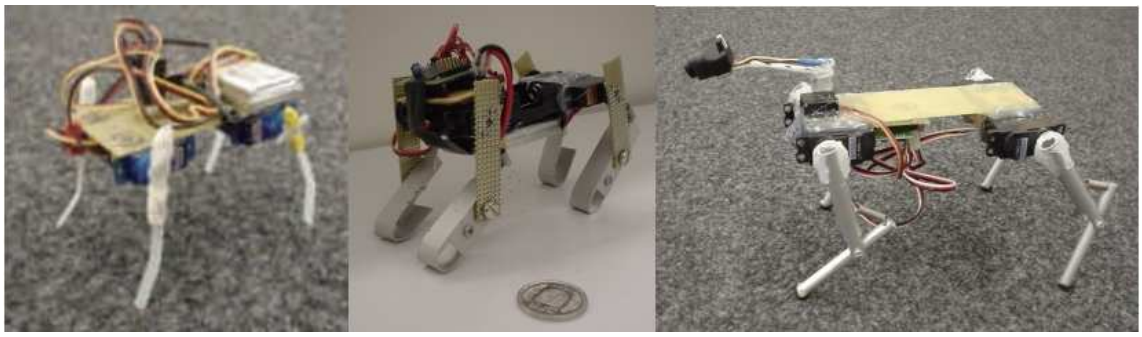


Figure 2.10.: Members of the "running dog project"
Left: The springy property is provided by the elastic material of the legs
Middle: MiniDog, a close relative of MiniDog6M, but without actuation in the spine
Right: Identical leg design as MiniDog, but different material

2.2.3.2. Locomotion concept

The outstanding characteristic throughout the running dog project, is that all the different morphologies are controlled with the same simple controller. The approach combines high speed, energy efficiency and robust yet simple control in a unique way. This is especially remarkable in face of the great variety of morphologies, the few degrees of freedom of the legs and the lack of sensory feedback.

While most locomotion approaches need to distinguish stance and flight phase, the springy legs need neither control for jumping nor for landing. This type of springy locomotion requires neither task-level or body state feedback nor active control over the leg length, because it relies to a large extent on the self-stabilising property of the spring-mass-system. Hence MiniDog6M's (as well as the other members') locomotion can be controlled via a simple sinusoidal control law [Iida 04a, Iida 04b].

Though the resulting hopping behaviour is quite irregular, the average speed can be controlled by means of frequency α, amplitude A and phase delay β between front and hind legs.

The resulting controller equations being in command of the servos are as follows:

$$\text{Front/right hand legs: } Motorvalue = A * \sin(w * \alpha) + offsetFront \tag{2.1}$$

$$\text{Hind/left hand legs: } Motorvalue = A * \sin(w * \beta + \phi) + offsetHind \tag{2.2}$$

Here w is an integer variable increased in each controller step, initialised with zero when the dog robot starts moving. The decision on sinusoidal control was not at all arbitrary, but biologically inspired [Seyfarth 02].

Figure 2.11 gives an impression of the motion sequence of the running robot's legs. Though captured from Puppy, a close relative of MiniDog6M, the (ir-)regularity is representative for all members of the "running dog project". Puppy served to investigate speed control and the influence of friction.

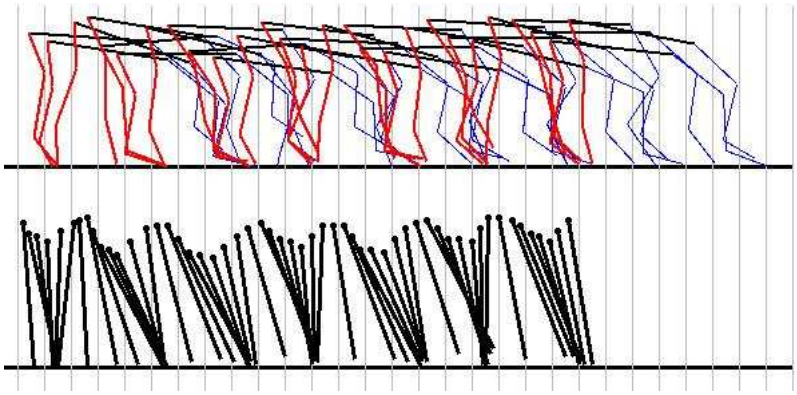


Figure 2.11.: Motion sequence of the Puppy's legs
Its (ir-)regularity is representative for all members of the "running dog project".

As we can see the running dog project is a great example of ecological balance, cheap design and morphological computation. Ecological balance means equal complexity of a robot's task, morphology and controller. The overall concept of sinusoidal control is implemented in all members of the "running dog project" and even in an artificial fish [Ziegler 05, Pfeifer 05]. For detailed information on self-stabilisation, speed control and the influence of friction, please refer to Fumiya Iida's work [Pfeifer 03b, Iida 04a, Pfeifer 05].

In the subsequent section, a methodology to examine how morphological properties influence evolved behaviours is introduced.

2.3. Isolating morphological effects on evolved behaviour

As already mentioned, Paul and Bongard investigated the optimisation of biped weight distribution simultaneously with a closed loop controller to achieve stable walking, but, generally speaking, little effort was spent to identify the morphological properties that make an agent suitable for a given task or controller design. An outstanding example for such investigations in legged locomotion will be presented in this section.

One of the sparse systematic investigations was performed by Lund et al who showed a correlation between body size, wheel base and sensor range and the performance of wheeled locomotion [Lund 97]. Having biologically motivated systems and thus legged locomotion in mind, the work of Bongard and Pfeifer is much more relevant for us [Pfeifer 03b].

They engaged in comparing evolved locomotion for ten legged agents which are shown in Figure 2.12. Irrespective of the particular body structure, each of the robots has an identical sensory system, actuation and controller design.

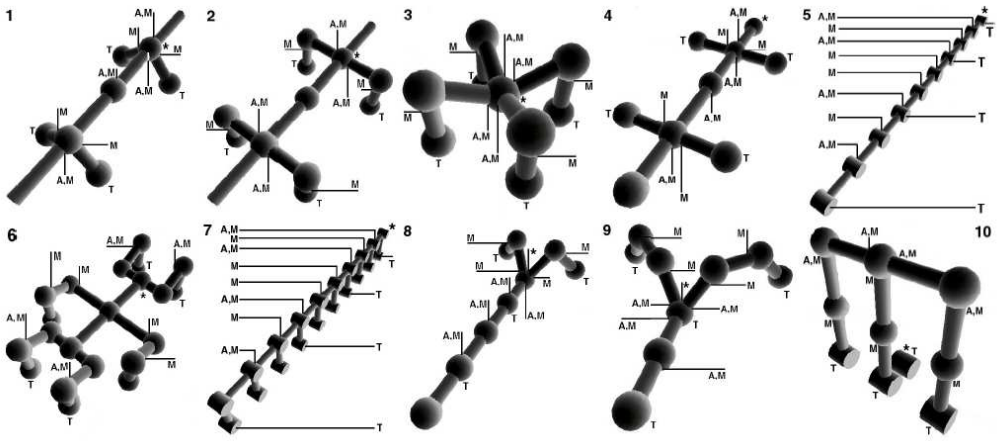


Figure 2.12.: Evolved locomotion for ten different morphologies
Quadruped and hexapod locomotion is especially stable, fast and easy to evolve. Regarding evolved control with neural networks, the performance of other agents can eventually be predicted comparing them to the ones above.

All of the agents are controlled by a partially recurrent neural network. The input layer receives input from four touch and four angle sensors. The output layer sends commands to eight one degree of freedom torsional motors. Input, output and hidden layer are fully connected. In addition, the hidden layer is fully recurrently connected. All weights are evolved using a fixed length, generational genetic algorithm. Each evolutionary run was conducted using a population size of 300, and was run for 200 generations. At the end of each generation, strong elitism was employed with an average of three point mutations using random replacement. On that basis they tried to derive a general policy how morphology can ease or complicate the evolution of behaviours for simulated agents.

Comparing the respective locomotion stability and speed, they found that quadrupeds and, above all, hexapods are particularly good candidates for which to evolve locomotion based on neural networks. Since the number and type of sensors as well as actuators are constant across all models, the reasons for that must be found in the specific shape and mass. One part of the explanation is a partial negative correlation between mass and performance. Unfortunately, this is not a sufficient explanation, since the three best agents (agent 2, 3 and 6 in Figure 2.12) offend against this apparent principle. Furthermore, additional hidden neurons improved the performance especially of segmented agents with many similar parts. Most of the agents achieved a relatively rhythmic gait during evolution.

Such a systematic investigation of many different morphologies could help to predict the performance of new robots within this evolutionary framework, if the unexplored agent shares one or more of the morphological characteristics with an agent that was examined earlier. Consequently, it is important to explore further how the morphology and control can be designed simultaneously in the process of evolutionary or ontogenetic develop-

mental processes [2].

By now, methodologies were proposed that evolve complete agents [Bongard 03, Pfeifer 03b], which means morphology and controller at a time, but these approaches are still quite restricted to single tasks and non-arbitrary morphologies. Typically, those agents can be composed of only few basic shapes, sensors and materials. Some of them still require quite a lot interference from human designers. So far all agents designed by such an automated design process are tested only in simulation.

In the next section, the so called “state of the art” is presented.

2.4. Related work

To provide an overview over the achievements in the field of legged locomotion, an assortment of projects is presented which are related to our project on the subject of morphological computation, compliant legs with springs, fast running, cyclic movement or use of movement primitives.

2.4.1. Morphological relevance in standing up

A whole body dynamics biped of humanlike shape, size, mass distribution, mobility range and strength was developed at AIST and later at the University of Tokyo. Aiming at generality and openness, the robot with the average height and weight of a Japanese person possesses 46 degrees of freedom [Pfeifer 03a, Kuniyoshi 04].

Through exploitation of natural physical dynamics and with sparse control at critical points, the experimental platform shown in Figure 2.13 on the facing page is able to stand up with roll and rise motion as observed in humans without assuming a particular task or posture. The control was entirely open-loop. This approach is closely related to the principle of cheap design as the programmer intuitively chooses some intermediate key postures, arranges them in appropriate temporal order and let the natural dynamics take care of the rest.

The actual movement was not self acquired, since the relevant constants are adjusted by human operators to save time. Anyway, the more interesting point is the fact that they found critical points where trajectories converge (compare Figure 2.14 on page 24) and that only these points are decisive for success or failure of the overall standing up task. As a result, the particular motions can diverge between critical points, but the completion of the goal is more tolerant of small perturbations in the postures and dimensional parameters of the robot’s body.

This is a great example of how a proper weight distribution can lead to good dynamics and thereby ease the given task.

2.4.2. High speed running with springy legs

In this subsection, two compliant leg models with springs will be presented. Though both are suitable for fast locomotion, they comply with differnt design and control concepts.

[2] This concept of Hardware-Software-Codesign has for a long time been addressed by the wide field of embedded system.

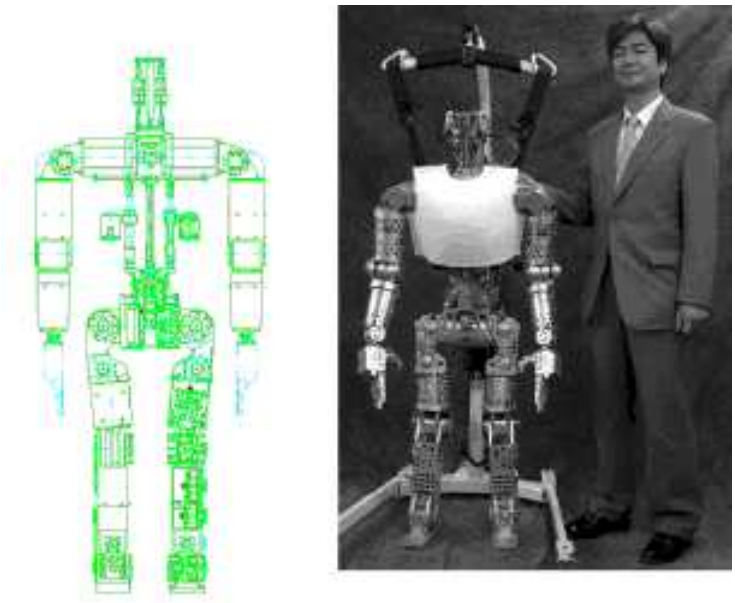


Figure 2.13.: Schematic (left) and photo (right) of Kuniyoshi and his biped
A whole body dynamics biped of humanlike shape, size, mass distribution, mobility range and strength that is able to stand up with roll and rise motion.

2.4.2.1. Empiric control of biologically inspired leg

Being inspired by biological standards, several parsimonious leg models for high speed running were developed. Focusing on the geometric alignment and function of the muscle-tendon system of canine's ankle joint, the one-legged running robot Kenken was built in 1999 [FastRunning 05]. Figure 2.15 shows Kenken in photo and schematic.

Kenken's leg design can be decomposed into three links. It uses two hydraulic actuators as muscles and a linear spring as a tendon. As a result Kenken has two active joints in hip and knee and one passive joint in the ankle that enables the toe to rotate freely on the ground during the stance phase. The leg spring between the thigh and heel is attached in parallel to the shank. This arrangement allows the robot to produce sufficient propulsion force by means of energy transfer from the knee during the stance phase. Absorbing the impulse at touchdown, the leg spring stores the available kinetic energy as potential energy for the next step. A knee extension contributes additional energy. During swing phase, it enables passive retraction and extension of the leg by releasing the stored energy.

Using foot switches and six potentiometers at each joint, an empirical controller based on the uncontrolled dynamics derived from the leg model was developed. Hence the robot has succeeded in one-legged hopping experiments proving that this leg mechanism is in fact effective for running. The induced motion sequence is depicted in Figure 2.16 on page 25. Whereas the simple controller function employed in the "running dog project" is suitable for many different morphologies, Kenken's controller was designed empirically based on analysis of the characteristic dynamics of the robot.

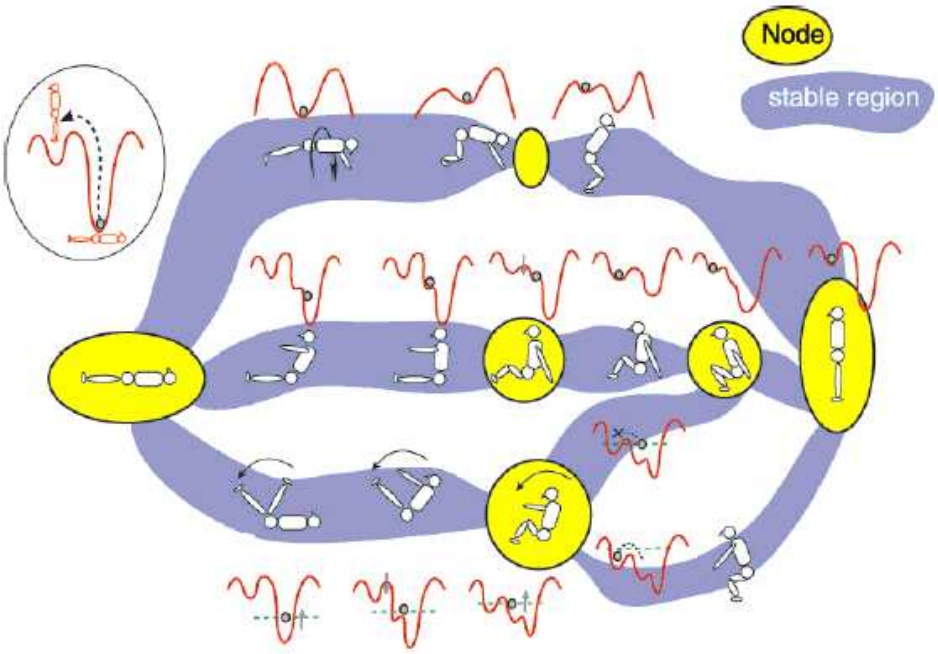


Figure 2.14.: Roll and rise trajectories
The particular motions can diverge between critical points. Only these points are decisive for success or failure of the overall standing up task.

As a direct extension of Kenken, a planar biped robot named Kenken II was developed to realise not only hopping, but also biped walking and running.

2.4.2.2. Cheap running with teleskopic joints

Another highly non-linear quadruped project utilising a compliant leg design with springs is the Scout 2 in Figure 2.17. It is the first power and computationally autonomous quadruped robot that achieves compliant running with a linear spring forming a telescopic joint and only one actuator per leg for rotation in the sagittal plane [Poulskakis 05, Buehler 00].

The torso contains the computer unit, the I/O boards and three batteries. The front and hind hip assemblies include two additional batteries, the DC motors, PWM amplifiers, gearboxes and pulleys for actuation.

Two controllers realise a PD algorithm and command the DC motors independently for front and hind legs. By simply controlling the constant desired hip torque during stance and the constant desired leg angle during flight phase, full mobility with energy efficient stable velocity control up to 1.2 $\frac{m}{s}$ is reached. Therefore the controller requires touchdown/lift-off detection and local feedback of the leg angles relative to the body provided by leg potentiometers and motor encoders. The linear potentiometers measure the displacement of the lower leg with respect to the upper leg in order to distinguish stance

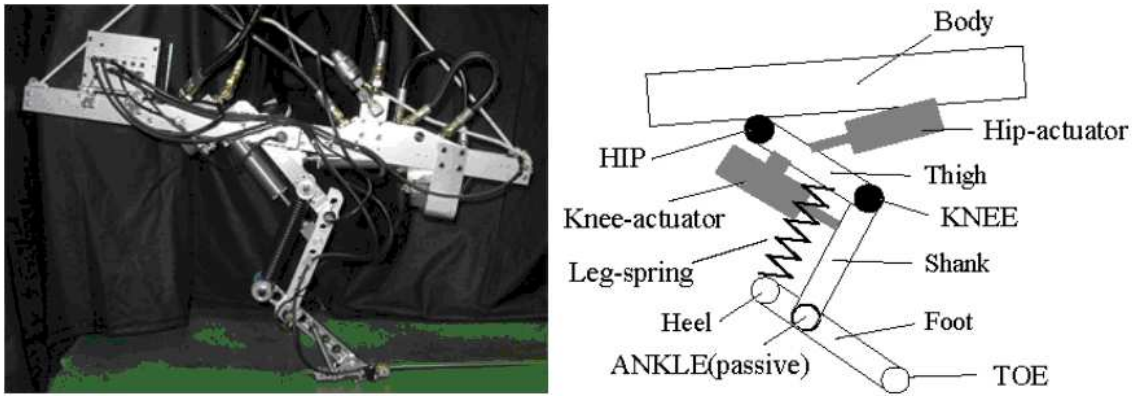


Figure 2.15.: Kenken in photo (left) and schematic (right)
The leg spring between the thigh and allows the robot to produce sufficient propulsion force by means of energy transfer from the knee during the stance phase. Absorbing the impulse at touchdown, it stores the available kinetic energy as potential energy for the next step.

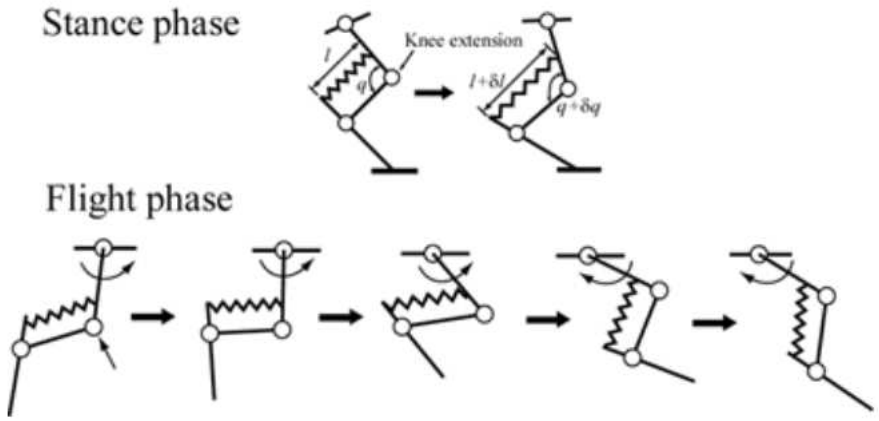


Figure 2.16.: Kenken's motion sequence

from flight phase. The angular displacement of the motor shaft is measured by incremental optical encoders. The controller is structured in two hierarchical levels and can be parameterised for each leg individually by motor torque, speed slope, maximum torque, touchdown and sweep limit angles. Here, the touchdown angles are needed to achieve cyclic motions. Each controller cycle is divided into free rotation, position control and velocity control.

In 2003, Scout 2 featured the simplest running control algorithm and the simplest mechanical design.

2.4.3. Generation of rhythmic motion sequences

Various experiments showed the suitability of central pattern generators (CPG) in combination with peripheral controllers to accomplish periodic locomotion. The CPG is often

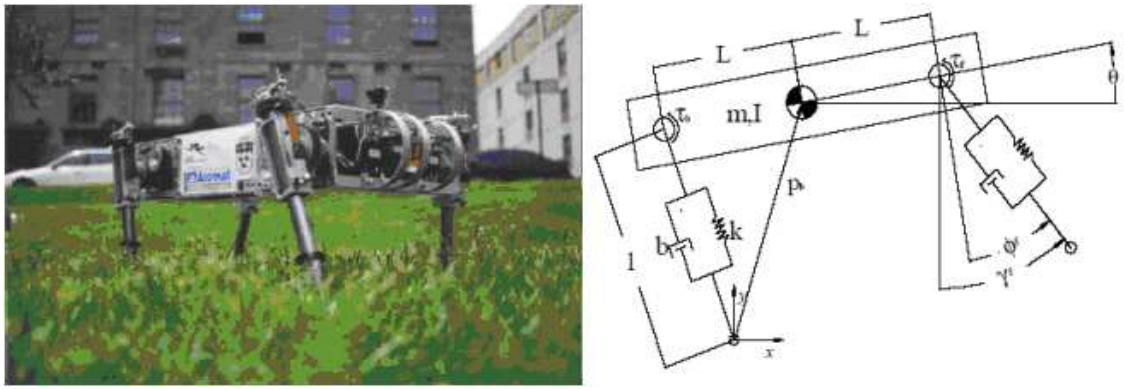


Figure 2.17.: Scout 2 (in photo and schematics)
Scout 2 featured the simplest running control algorithm and the simplest mechanical design employing a telescopic joint with a linear spring and only one actuator per leg.

modelled with a neural oscillator consisting of two mutually inhibiting neurons, which was originally designed to characterise the alternating activation of flexors and extensors in a cat limb [Kimura 00]. As will be demonstrated, rhythmic motion sequences can also be induce by a distributed control scheme.

2.4.3.1. CPG based control

The subsequent example introduces a quadruped that employs legs with springs controlled by a neural oscillator network whose frequency matches that of the spring-mass-environment-system oscillation to achieve dynamically stable motion. Tekken II [Kimura 04, Kimura 03, Kimura 01, Kimura 99], which is shown in Figure 2.18 on the next page, is a quadruped, self-contained robot that is capable of walking in outdoor natural environment. With its design based on biological concepts, Tekken II realises robust medium speed walking on irregular terrain and fast trotting on flat terrain.

Tekken II has a hip pitch joint, a hip yaw joint, a knee pitch joint and an ankle pitch joint on each leg. Whereas the latter can be passively rotated, the others are activated by DC motors. To support locomotion, a hard and a soft extensible spring were attached between the lower limb and the long foot. The hard one is needed for shock absorption and reuse of the kinetic energy. The other one keeps the angle of the ankle joint constant during the flight phase. This spring-damper system is controlled by a PID controller for each joint. The passive hip knee joint can change the mechanical stiffness of the spring-lock-mechanism between the stance phase and swing phase. In order to measure the body pitch and roll angles, two rate gyroscopes and two inclinometers are mounted on the body. The direction in which Tekken moves can be changed by using the hip yaw joints.

Tekken II's neural system is modelled by numerous levels of CPGs, reflexes and responses. Here 'reflex' is defined as joint torque generation and 'response' means modulation of the CPG's active phase, both as response on sensory feedback. Using CPG, each gait is described as stable oscillations of a robot-environment system. The transition between two gaits is achieved by modifying a few parameters in the neural oscillator network, which keeps these oscillations steady.

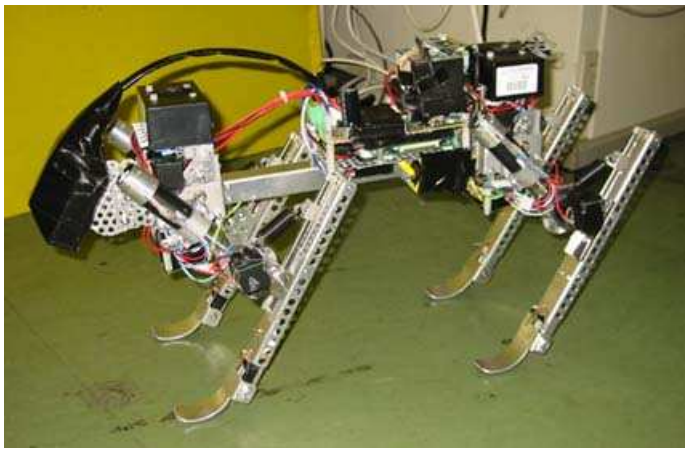


Figure 2.18.: Tekken II
Tekken II is capable of dynamically stable walking in outdoor natural environment by employing a neural oscillator network whose frequency matches that of the spring-mass environment-system oscillation.

Though Tekken II and the “running dog project” comply with different control methods, both of them use legs with springs and oscillatory control for successful dynamically stable locomotion. Tekken requires a highly complex controller and therefore reaches much more precise motions. Unfortunately, Tekken II can only stand up from a single predefined posture, namely lying back in an upright posture with the feet stretched towards the front.

2.4.3.2. From CPG to emergence of rhythmic gaits

The biologically inspired walking machine BISAM[3] [Luksch 02, Ilg 00, Ilg 99, Ilg 98] implements a control architecture for high numbers of degrees of freedom. Adaptation and optimisation of different stable mammalian locomotion behaviours are accomplished with the coupling of different control levels.

BISAM, which is shown in Figure 2.19 on the following page, is 70 cm high, weighs 23 kg and consists of a main body, four identical legs and a head. The body as well as each leg consist of four segments connected by five rotary joints driven by DC motors and ball screws gears. The sensory system is equipped with three component force sensors in the feet along with inclinometers and angle velocity sensors in the body.

In a first approach, optimised adaptation is attained by online reflex learning with an actor critic algorithm and a self-organised RBF-network for function approximation. The motor actions are generated by the output of the oscillators that directly and indirectly integrate sensor information. Depending on the current sensor state, reflexes and higher level behaviours can add offsets or even actions to the oscillator output to adapt or even dominate it in order to ensure the security of the robot. In contrast to the “running dog project”,

[3] *B*iologically *I*nspired w*A*lking *M*achine

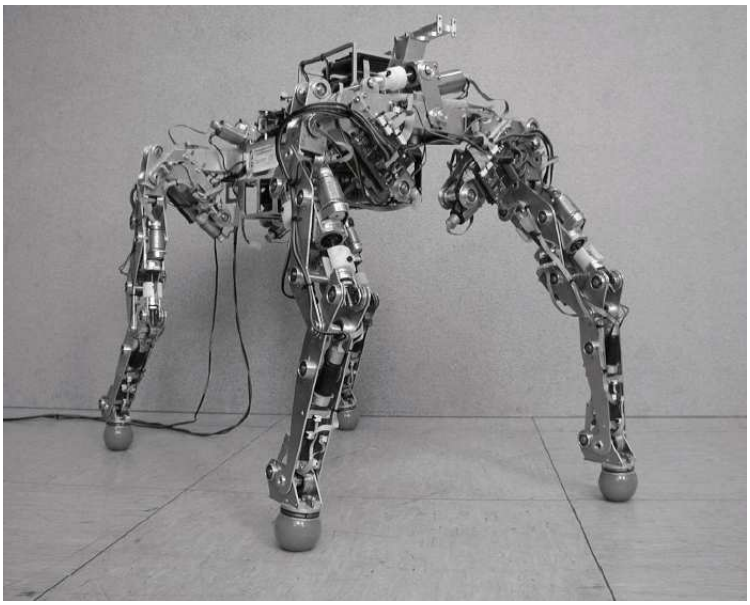


Figure 2.19.: Biologically inspired walking machine BISAM
Adaptation and optimisation of stable oscillatory mammalian locomotion is accomplished by coupling and superimposing of movement primitives selected from different behaviour based control levels

BISAM's control architecture has to distinguish stance from flight phase and as a result switch between two elementary oscillators that model the respective CPGs for either one or the other phase. Consequently, the controller can easily extend the swing phase due to delayed switching between stance and flight plane. Such a possible increase of the step length becomes necessary, e.g. when, after the regular step length, the robot's foot does not touch the ground as its bearer steps into a hole. This approach guarantees robust yet slow locomotion. Aiming at faster locomotion, the response time of the controller had to be improved.

In view of highly dynamic environments, reactive control offers early enough reaction and much more adaptability than CPG. Belonging to the general category of behaviour based control, there is a close coupling between perception and action. Inspired by psychology, neurology and ethology, the basis for behaviour based control was provided by Rodney Brooks' subsumption architecture [Luksch 02]. Here the controller tasks are not decomposed into functional blocks that stepwise extract an environment model and on that basis create plans and appropriate reactions, but into behavioural competences instead. Undemanding behaviours or reflexes are implemented as simple stimulus-reaction-pairs. More complex behaviours emerge through their hierarchical and sequential interplay. Each basic behaviour only reacts to relevant stimuli. Concurrent behaviours must be coordinated. Higher levels can change the output and variables of lower levels, while lower level know nothing about the existence of higher levels. Thus the architecture can easily be extended with further higher behaviours and the robot can even continue, if higher levels drop out As a trade-off and due to the non-existence of a suitable environment model, it is not possible for the agent to develop long-term plans. The subsumption architecture was suc-

cessfully implemented in many robotic fields e.g. for navigation tasks in autonomous agents.

For these reasons a behaviour-based control has been applied in a second approach to BISAM. The architecture is mainly reactive, but with few planning components. Rhythmic motion is no longer generated by a neuro-oscillator, but emerges from the interplay of the basic components in even terrain.

Each behaviour component has an input to determine its relevance respectively activation within [0.0; 1.0]. A value of 0.0 deactivates the behaviour, whereas a value of 1.0 leads to maximum influence on the robot. Values in between the extremes can be used to mix the motor primitives from different behaviours. In doing so, the activation expresses the relevance of the behaviour respectively the corresponding output for the current situation.

After evaluating the current sensor input and after selecting an appropriate motor action, each behaviour component gives an additional output expressing its satisfaction. Here satisfaction means to what extent the current sensor state equals the desired world state of the behaviour. This output is called activity and, again, this value lies within [0.0; 1.0]. Here, a value of 0.0 means that there is nothing to do for this behaviour, because its goal has already been reached. A value of 1.0 means that the behaviour wants to change the current state with every means it has. Note, that the activity must not exceed the behaviour's activation.

All basic behaviours are arranged in hierarchical groups within a single network. This model consists of a network of different competences, each of which generates motor output as soon as its specific goal is not met. To merge all resulting motor programs, supplementary knots are added to the network in which concurrent behaviours are combined by either superposition or prioritisation.

BISAM achieves cyclic statically stable walking for secure movement in segments of extremely rough terrain, arbitrary leg step sequence to overcome obstacles. With its reactive control structure, it is further capable dynamically stable trotting. Rhythmic motion was no longer generated by neuro-oscillators, but emerges from the interplay of the basic components in even terrain. Regrettably, BISAM cannot stand up and thus makes posture control and stable locomotion even more important.

2.4.3.3. Benefits in both approaches

Both approaches, BISAM and Tekken II, create their own non-linear dynamics exploiting the strong interaction of neural system and the environment. Hence autonomous adaptation under changes in the environment (e.g. adaptive walking on irregular terrain) and under adjustment of the neural system parameters (e.g. gait transition or change of walking speed) is induced without an internal representation of its own body or of the environment. Therefore, serious problems such as generating a body image and an environment model, autonomous planning, coping with discrepancies between planned and actual motion etc can be avoided. Unfortunately, none of them is capable of fast running. For walking on

terrain of higher degree of irregularity, e.g. with holes and large obstacles, adaptation based on vision would be a necessity.

2.4.4. Different approaches to motor primitives

Regarding the conceptual idea of motor primitives, several other project, mainly in the field of imitation learning, can be found that basically work with the same concept, but different representations and/or terminology. In this section, we provide a short overview over an assortment of interesting projects.

2.4.4.1. Constitutional research

Arbib was the first who proposed the idea of movement primitives. In literature, this concept also is referred to as movement primitives, motor schemas, motor programs, basis behaviours or action units [Paine 04, Arbib 81].

Stefan Schaal [Schaal 99] defines movement primitives as a sequence of actions that can accomplish a certain movement goal. He also stressed the importance of modular motor control in form of movement primitives in imitation learning of humanoid robots as mentioned in section 1.1 on page 3. Herein a movement primitive can be as simple as an elementary action or implement complete temporal behaviours. In that sense, primitives can be formalised as form of control policy which results in a compact state-action representation where only a few parameters need to be adjusted for a specific goal.

Later, he developed the theory of dynamic movement primitives (DMP) which are represented as non-linear differential equations in order to create smooth motions [Schaal 02]. The idea is to employ well-defined mathematical concepts such as attractor equations to implement the basic behavioural patterns. The DMP's attractor landscape can be modified using statistical learning to match the detailed needs of the current situation [Schaal 04]. The idea of using non-linear dynamic systems as policies is the most closely related to the original idea of motor pattern generators (MPG) in neurobiology. Further they implemented this system of programmable pattern generators on a complex anthropomorphic robot [Schaal 00].

2.4.4.2. Imitation learning

Combining perception and generation of motor primitives, real time 3D imitation was achieved by Yasua Kuniyoshi et al [Kuniyoshi 94]. They proposed a method where a robot learns reusable task plans by observing assembly tasks performed by a human operator. In doing so, the agent splits up a continuous task into meaningful units. The identification of those temporal segments is performed by concurrent recognition processes with active attention control. For the overall imitation task, the system consists of three units: seeing, understanding and doing. This approach is called "learning by watching". The action recogniser relies on an action model, an environment model and an attention stack. The action model itself is a collection of knots representing the temporal structure. Currently, the temporal segments, called "assembly motion", are characterised by qualitative motion features of the hand and the relative location of hand and object. Further, each partial

action is characterised by a set of "assembly operations" describing changes or invariants in the environment during the corresponding time sector. A task is then composed of several segments that together cover the whole temporal extent of an assembly task. As a basis for recognition, the design has to provide a hand-coded set of fixed templates of partial task sequences. Though they use "reusable task knowledge or plan" instead of motor primitive, it essentially means the same, except that the movement primitives can be adapted by parameterising the given templates.

A similar approach was taken by Mataric et al [Mataric 98, Mataric 02]. Other than "learning by watching", complex behaviours are perceived as not only sequences but also as superposition of basis behaviours. In doing so, they investigated alternative representations including convergent vector fields in joint and Cartesian space, impedance control, interpolated joint-space control and CPG. Furthermore, they engaged in self-acquisition of primitives on the basis of observing an exemplary set of simple movements. The concept was tested on a physics-based humanoid simulation, two humanoid avatars, AIBO and a number of wheeled robots. Talking about representation, Mataric speaks of a set of sensory-motor primitives forming a basis movement vocabulary.

Both approaches use fixed behaviours and function in the style of mirror neurons found in primates.

2.4.4.3. Self-organisation

Jun Tani et al [Tani 02] started with a localised representation of motor primitives which were self organised in a hierarchical neural network. Later, they switched towards a distributed representation scheme. Since this model is characterised by two levels of forward models, it is called "forwarding forward model". Each level was implemented as Jordan-type recurrent neural network. Here, behaviours are perceived and generated as combination of reusable pieces. Those sensory-motor spatio-temporal patterns are self-organised in the lower level forward model. The "control neurons" in the higher level are bi-directional connected to the lower level. By means of parametric bias, they are able to switch between several primitives. With that concept, they also accomplished imitation learning on the basis of those pre-learned primitives, while only the upper level neurons were allowed to adapt. Further, they observed how the interaction between the two levels enables adaptation if the target object is moved so that, while executing a grasping task, the agent has to switch between two fixed behaviours in the middle of the execution.

Later, Rainer W Paine and Jun Tani [Paine 04] deepened the studies of this dynamic adaptation process. Using a similar concept, they decided on two levels of continuous time recurrent neural networks. The two levels are connected bi-directionally. Again, the higher level was allowed to switch between pre-learned primitives represented in the lower level network. Both levels were stepwise evolved for tasks of increasing complexity. Here a navigation task to multiple goals in a maze environment were achieved, if starting from the same initial position. In both projects, the movement primitives remain unchanged, once they self-organised in the lower level.

2.5. Entitlement to this thesis

The prevalent interest of embodied AI and behaviour-based robotics is to provide insight into animal/human behaviours by building biologically inspired robots. This principle is commonly known as “learning by building”.

Animals, for instance, lie down only in a few particular postures resulting from the constraints of the body. By taking physical interaction between the body and the environment into account, the sensory state of the robot can no longer be arbitrary and thus is significantly reduced. Providing a set of meaningful, adaptive motor primitives, which can be composed into a broad and general movement repertoire, is an appealing biologically inspired strategy. Despite the restriction to a distinct set, diverse activities can emerge by means of sequential or linear combination of single primitives.

Since trial and error at random is generally not a good strategy for an agent behaving in natural environment, structured search and learning is much more suitable to acquire such higher abilities. Reinforcement learning is a widespread means for autonomous agents to get along in new situation. Making categories in the state space is a crucial issue for learning algorithms since the agent faces the problem of a enormous state respectively search space. Hence the designers must find an appropriate way to cut down the state-action space. While it is obvious, that the introduction of discrete actions alone reduces the complexity of a learning task by avoiding online trajectory planning, another important assignment is to further investigate morphological implications on robot control. The support of the learning progress is consequently an important feature of a good basic set.

Due to the fact that ecologically well balanced robots require only control e.g. for locomotion, all the remaining resources can be used to develop higher abilities, e.g. learning. Ecological balance is an important design principle for intelligent systems and means finding a balance between the complexity of the given task, robot morphology and the controller. In classical approaches to robotics, the complexity of the controller exceeds that of the others by far. Anyway, it can be reduced by encouraging emergent behaviours. This can be achieved by exploiting morphological properties such as material and mechanical design of the robot. In section 2.1 on page 9 this kind of unburdening the controller was introduced as morphological computation. Since redundancy is a necessity to the emergence of interesting new behaviours, this is naturally a trade-off to cheap design, which stresses parsimony. MiniDog6M should realise these principles as close as possible.

Motor primitives as a parsimonious means of encoding a basic movement vocabulary are very effective for structuring a robot’s movement. Forming a set of fundamental agent-environment-interaction, they build an elegant modular concept for motor control which is driven by the kinematical and dynamic constraints of the motor system and represent frequently executed moves in a robotic task.

In contrast to other approaches that put forward the developement of adaptive motor primitives, the example of the frog, which was already mentioned in the introduction, suggests the use of fixed primitives. The primitives found in its spinal chord only vary by adapting

the power with which the motion is carried out to the strength of the sensor stimulus. Nevertheless the frog is an adaptive being. Its fixed behaviours are combined into sequences or mixed by means of superposition. To follow suit, we decide against the use of adaptive primitives. Further, we decided to use a localised instead of a distributed representation for the primitives, since the primitives that are used throughout this thesis are pre-determined.

It is widely believed, that knowledge and behaviours that are acquired in one environment cannot be used if the setting changes. Adaptability is needed for the capability of standing up in different terrain, e.g. with different slopes. In order to be able to react to changing environment, the agent must constantly keep on learning. Regretably, many examples showed that this kind of adaptation takes as long as learning from scratch. We contrast this common claim of steady learning: Changing environmental conditions may prevent certain behaviours, but new behaviour may come up. Even if some of the prelearned behaviours fail, the robot might still be able to fulfil its task by employing another behaviour. Being equipped with enough behavioural diversity, the robot may still be able to perform its task after several unsuccessful trials. Therefore, it is much more interesting to investigate the feasibility of the gained knowledge in changing environment without adaptation. Discovering the suitability of a multitude of complex sequences for different morphologies, tasks and environments, this means the evasion of online adaptation and still being able to succeed in new situations by exploiting behavioural diversity.

The intention of this thesis is to systematically investigate if and how the influence of morphological constraints to motor control makes learning faster and simpler. The goal is to provide a methodology to explore how the morphological properties contribute to generating these discrete entities in the continuous sensory space. Such a methodology needs to investigate how different vocabularies affect behavioural diversity, robustness and learning progress. Robustness in this context means that the behaviours are tolerant against changes in morphology, environment and posture. Herefrom, guidelines and design principles for the developement of motor primitives as well as the morphologies can be extracted.

For this purpose, such a methodology has to settle the following claims:

- Primary experiments should reveal the potential of the robot dog's sensory and locomotion system. The result gained from these experiments will build the basis for the subsequent investigations.
- In order to get an appreciation for possible vocabularies, a first set of motor primitives should be extracted out of well known standing up sequences.
- To derive quantitative results, abstract, task and platform independent measures are needed to categorise and evaluate single motor primitives, entire vocabularies and behavioural diversity.
- Different vocabularies, as well as different morphologies and different ground configurations are to be examined.

In order to finalise this list, we will proceed as follows:

1. First, the morphological and control concepts of our research platform will be introduced and subsequently several gaits and a pre-programmed standing up motions and their feasibility of these behaviours on flattish, medium and steep slopes will be elaborated.

2. Out of these pre-programmed sequences, the first set of motor primitives will be extracted.

3. Further, we will establish a general means to evaluate them in regard of the frequencies they assign to the motors. On that basis, several vocabularies will be worked out, each of which being an assortment of motor primitives for the vocabularies that will be investigated throughout this thesis.

4. In the second stage, MiniDog6M is transferred into simulation to enable further experiments with different morphologies which cannot be changed effortlessly in the real world.

5. On that basis, the morphologies that will be investigated in the course of this thesis will be selected.

6. After that, a general means to compare the behavioural diversity of different tasks or vocabularies will be established.

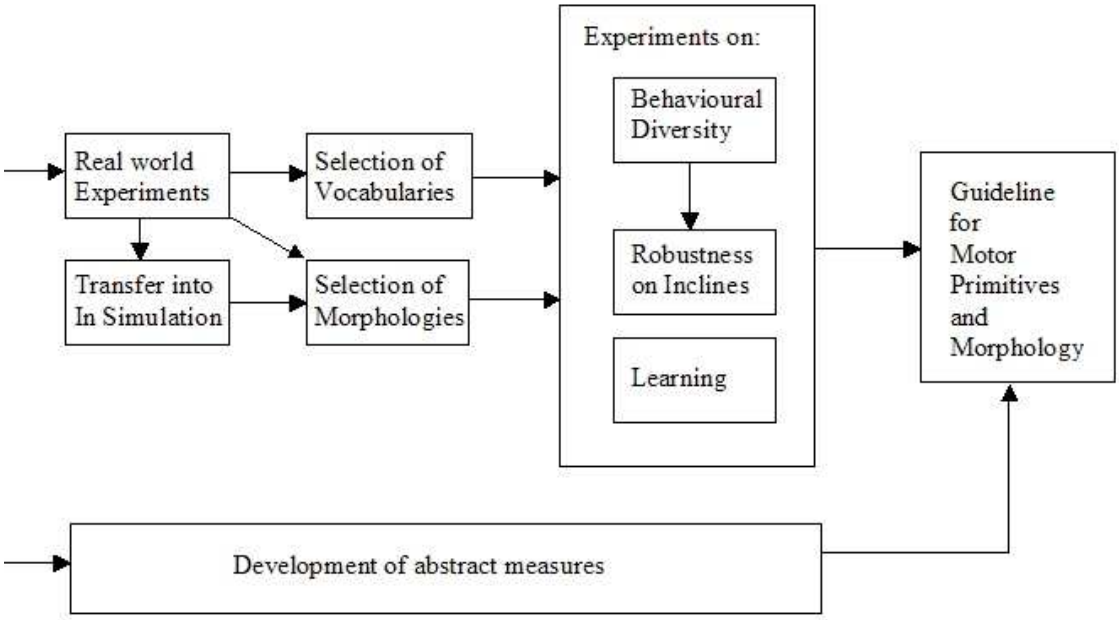
7. In the first row of experiments for the evaluation of our vocabularies, diverse standing up sequences will be generated as a trial and error combination of the underlying motor primitives.

8. The second set of experiments addresses the robustness of these sequences on inclines.

9. In the third batch of experiments, Reinforcement Learning is engaged.

10. At the end, the main results of our work will be summed up. Together with these concluding remarks, future assignments that directly hook up on the framework presented in this thesis will be elaborated.

The deduction of appropriate measures is independent of the experiments. Hence, their developement can be considered coexistent. The dependenies which affect the time plan of the project can be illustrated as follows:



The time flow increments from left to right hand side of the graphic. Tasks that are arranged one below the other can be executed in parallel. The arrows depict the dependency among them. If two task are dependent and despite of that aligned vertically, the dependency is weakened into influence.

In the next chapter, the first item on the afore mentioned list will be realised.

3. Real world experiments

In this chapter, we will give details on the physical configuration and locomotion repertoire of MiniDog6M. Then we will provide a description of our experiments in the real world which build the basis of our further studies.

3.1. Morphology of MiniDog6M

In this section, MiniDog6M's mechanical structure, as well as the choice and placement of sensors and actuators will be presented.

As seen in Figure 1.1 on page 4, the body is 125 mm long, 73 mm wide, 30 mm high. The complete dog weighs 194.9 g. The control area has a base of approximately 58·29 mm^2, is up to 50 mm high, consists of power switch, acceleration sensor as well as the interfaces for motors and PC (via RS232) and is applied to the head, which results in a significant frontal weight overhead of about 49.7 g.

MiniDog6M has six motors, one for each leg plus two forming the spine, which provide an approximately semicircular trajectory addressed by values between 0 and 255. The motors in hip and shoulders move the legs, the front spinal motor enables the dog to bend (bend motor) and the hind spinal motor causes a rotational movement of the hind respective to the front (twist motor). The motors chosen for the MiniDog6M project are S-811 MG servos offering 33 Ncm at a rate of 0.09 $\frac{s}{60^\circ}$ at an operating voltage of 6V. Each servo weighs 19g on 29.8·12·29.6 mm^3. Assembled with hot glue, they form the robot's main body. Sufficient power is supplied by an electric generator and stabilised by an accompanying battery.

Since the centre of mass is relatively high in the original model, it should be lowered so that MiniDog6M can cope with small disturbances without falling. With an additional weight of 5 g attached at each foot, this decisive point is shifted towards the lower head. The effects can be summarised in stabilisation of locomotion, ease of standing up and an

increased rotational momentum enabling rolling over on its back.

Being part of the “running dog project”, MiniDog6M is inspired by anatomical studies of a canine. In contrast to the four-legged archetypes, each leg is designed identically. Every leg, as shown in Figure 3.1, has one active degree of freedom controlled by an independent servomotor (cuboid in the schematic) and one passive degree of freedom controlled by a spring (lightning-shape in schematic) connecting upper and lower leg. The spring constant is 481. A spring-damper system is a common model approximating the visco-elastic properties of a muscle. The weight of the legs is low with 7.8 g, since the upper is made of a 5 cm long piece of plastic and the lower part consists of 5.5 cm of aluminium. The latter enables MiniDog6M to slip over the ground very easily thanks to the low friction sole of its feet.

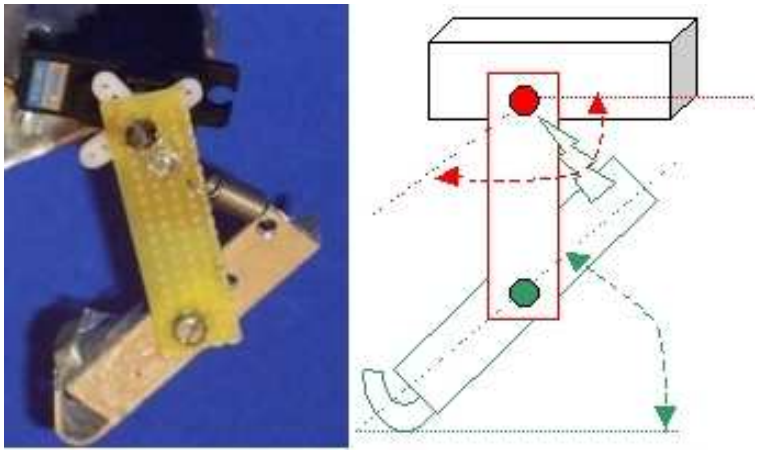


Figure 3.1.: MiniDog6M’s leg design in photo (left) and schematic (right)
Red: active degree of freedom controlled by a servo motor (cuboid)
Green: passive degree of freedom controlled by a spring (lightning shape)

The dashed arrows in the schematic indicate the qualitative movement of those parts of the leg that are highlighted with the same colour. Despite identical design and identical controller interval [0..255], the angular range of front and hind legs is uneven because of unlike attachment to the correspondig servo. Additionally, the scope of the left front leg is limited more than the other motors due to morphological restrictions. The angular range of each physical unit is listed in Table 3.1 on the next page.

So far the sensory system is limited to a tri-axial capacitive accelerometer located within the control area of MiniDog6M’s head. The sensor element consists of a fixed and a flexible electrode. Due to mass inertia, the distance between the two electrodes changes proportionally to the acceleration force acting on the flexible electrode plate. This change in the plates’ relative position results in the proportionate change of the static capacity between the electrodes. This deformation is detected by a capacitor on the opposing split electrode. Following this simple principle is possible to measure static and dynamic acceleration on all three axis within a single small, cost and energy efficient structure. We employ the Star Micronics ACA 302 sensor [Micronics 05], which is shown in Figure 3.2 on the facing page, with built-in amplifiers. The sensor is supplied with 2.7 to 5.5 volts DC. More sensors are planned.

Unit	Angular Range	
Head	bend down: 65°	bend up: 62°
	twist right: 72.5°	twist left: 73°
Front right upper leg	forward: 51°	backward: 108°
Hind right upper leg	forward: 51°	backward: 104.5°
Front left upper leg	forward: 34.5°	backward: 59°
Hind left upper leg	forward: 65°	backward: 92.5°
Lower legs	rest position: 60°	max deflection: 90°

Table 3.1.: Angular range physical units driven by servo or spring
Note that there is an imbalance between left and right hand legs.

Figure 3.2.: Acceleration sensor employed in MiniDog6
The sensor element consists of a fixed and a flexible electrode which makes it possible to measure static and dynamic acceleration on all three axes within a single small, cost and energy efficient structure.

This simple morphology with its minimal mechanical and electronic complexity increases the reliability and robustness by significantly reducing the major sources of failure while lowering the cost of the platform.

In the next section, MiniDog6M's locomotion repertoire, which arises from its simple sinusoidal control, will be presented.

3.2. Locomotion repertoire of MiniDog6M

Being part of the "running dog project", MiniDog6M is capable of more gaits than just running. They will be investigated in this section.

As MiniDog6M's locomotion is controlled by a simple sinusoidal controller equation as introduced above, everything we familiarised for the locomotion concept of the "running dog project" is also true for MiniDog6M. Further, we enabled not only running behaviour, but also trotting and backing up. During running, front and hind legs are moved in parallel, as are left and right hand legs during trotting. Both gaits can be used for forward and backward locomotion.

The hopping motion is provided by the servos in shoulder and hips which move the legs back and forth. Forward movement emerges in combination with low friction soles of feet and one spring per leg which, as passive joint, expands and contracts uncontrollably. For optimal support of MiniDog6M's front and hind legs, a phase delay of π is preferred. Thereby, the evasion of one variable, namely phase delay, occurs as pleasant side effect making the control even more parsimonious. The resulting controller equation for each servo is as follows:

$$\text{Front/right hand legs: } Motorvalue = A \cdot \sin(w \cdot f) + offsetFront \tag{3.1}$$

$$\text{Hind/left hand legs: } Motorvalue = A \cdot (-\sin(w \cdot f)) + offsetHind \tag{3.2}$$

Here w is an integer variable increased in each controller step and initialised with zero when the robot dog starts moving. Whether a gait is used for forward and backward locomotion purely depends on the offset. If the offset shifts the centre of the motion towards the front, the robot will be pushed in that direction. This effect also holds for backward motion. Regretably, running backwards is unstable, since, after just a few steps, the basic version of MiniDog6M cants over the small low friction feet. This misachievement is based on the disadvantageous location of the centre of mass. So, without additional weights attached to the hind of the robot, the way of fast backing up is only stable in simulation.

In order to represent the robot's locomotion repertoire, we defined the enumeration data type `Gaits`.

```
Gaits  {Still,  Trotting,  BackUp,  Running  FastBack};
Index     0         1         2        3         4
```

We collected data from the accelerometer to see if it is possible to discriminate different gaits and/or ground friction. In other words, if the acceleration values reflect the fact that trotting is slower than running, that backwards means negative and forward locomotion positive acceleration or that the same gait might be faster on ground with low friction than it is on ground with high friction. The gained data of the relevant gaits are captured over 200 means of 10 sensor values respectively. The results are listed in Table 3.2 on the next page for a rough surface with high friction and for slippery hard ground.

Regarding the expectation, we can see the tendency that for each axis backing up is less accelerated than trotting and that running has the highest acceleration. Further we find the tendency that all gaits are faster on low friction surface. Unfortunately these tendencies are insignificant regarding the extension of the whole interval.

One of the most surprising facts is that locomotion does not need constant servo control, but, thanks to the springs, manages to create a continuous movement out of discrete angles and control operating at coarse-grained time steps. Other locomotion approaches need a controller period significantly shorter than the period here (usually in the range of only few milliseconds instead of here 1s). The trick is that the intrinsic dynamics of our model

Gait	0	1	2	3
Axis 0:				
Minimum:	-2	-127	-124	-128
Expectation:	13	22	18	39
Maximum:	35	124	125	127
Axis 1:				
Minimum:	15	-120	-120	-128
Expectation:	30	36	26	44
Maximum:	49	126	118	126
Axis 2:				
Minimum:	14	-43	-67	-128
Expectation:	29	33	22	43
Maximum:	45	124	116	126

Gait	0	1	2	3
Axis 0:				
Minimum:	-2	-128	-128	-128
Expectation:	13	22	26	39
Maximum:	35	126	122	127
Axis 1:				
Minimum:	15	-123	-41	-128
Expectation:	30	38	36	53
Maximum:	49	123	127	127
Axis 2:				
Minimum:	14	-116	-38	-128
Expectation:	29	35	32	52
Maximum:	45	114	119	127

Table 3.2.: Sensor data of the accelerometer of different gaits and different surface condition
All gaits were tested on a rough surface with high friction (left) and slippery hard ground (right)
The excepted acceleration is lower for backing up than it is for trotting. Further the robot is less accelerated during trotting than it is during running. The excepted acceleration is higher on low friction than on high friction surface. Gaits and surface cannot be discriminated on the basis of acceleration since these tendencies are insignificant regarding the extension of the whole interval.

take care of the rest. This cohesion is qualitatively pictured in Figure 3.3 on the following page where the black line denotes the robot's displacement from its initial position over time. The red arrows record the sparse motor commands sent by the controller. The dashed line marks the intended leg trajectory.

We will not go into detail on the subject of gaits and speed variations, since the main focus of the case study lies on standing up. First experiments on this will be described in the next section.

3.3. Controller-based standing-up approach

To accomplish the task of standing up behaviour from arbitrary postures, we define two subtasks, namely state recognition and standing up. How a classification of the robot's position is conducted, is described in the first subsection. Then our experimental results derived from pre-programmed sequences are presented. Finally, the resulting controller is explained.

3.3.1. Classifying the robot's position

First, as a trigger for the standing up behaviour, the robot must recognise whether it is standing or lying down. Furthermore, since it is not possible to stand up using the same behaviour sequence from all positions, it is in our interest to find out on which side it fell.

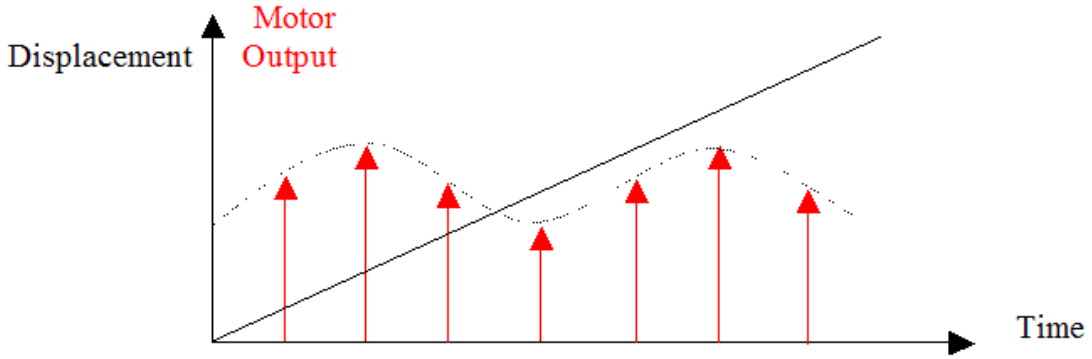


Figure 3.3.: Relationship between motor output (red arrows) and the robot dog's displacement from its initial position over time (black line). The trick is that the intrinsic dynamics of our model take over in the meantime between the motor outputs.

A classification of the basic positions can be achieved by using not more than the inexact values provided by the acceleration sensor. These main positions are standing (*Stand*), lying on the left side (*Left*), lying on the right side (*Right*), lying on the back (*Back*), standing on the head (*Head*) and sitting (*Sit*). As the sensor is located in the head, these positions only refer to the posture of the head.

We define an enumeration data type `Positions` to represent those basic orthogonal positions.

```
Positions {Stand, Left, Right, Back Head Sit};
  Index      0      1      2     3    4    5
```

In order to achieve a reliable position classification over 200 means of 10 sensor values respectively were recorded. The test data is shown in Table 3.3 on page 44 and illustrated in Figure 3.4 on the facing page. For each axis the lines represent the interval for one of the extreme positions. They are sorted top down as follows: *Stand* (black), *Left* (light blue), *Right* (red), *Back* (green), *Head* (dark blue), *Sit* (orange).

For all initial positions the expectation is (almost) in the middle of the interval. Looking at the extention of the intervals, it becomes obvious that we cannot discriminate any inclined position between the extremes or different ground angles, since the intervals of the orthogonal positions are too close to each other. So if a more precise resolution of the position is needed an extra gyroskop must be acquired.

The classification of these positions is performed by a fixed guessing algorithm based on expectations gained from the test data mapped in the six basic positions.

The estimation is based on the difference between the actual acceleration values and the expectation of every position. The least difference is taken as first guess and the next

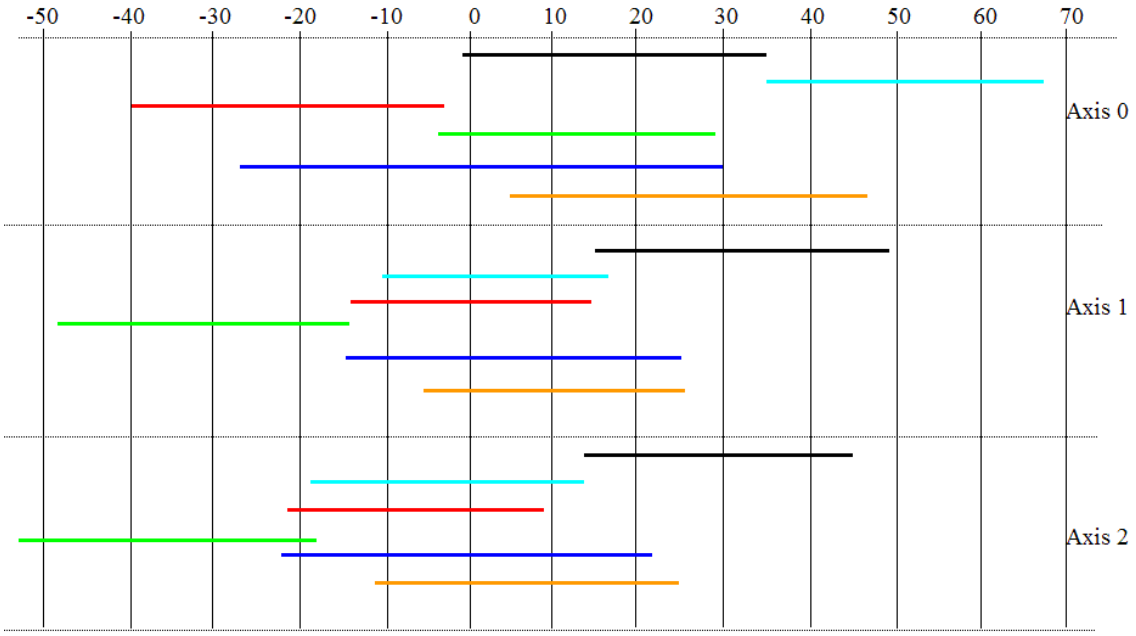


Figure 3.4.: Range of averages of ten accelerometer value (Respective limiting values are listed in Table 3.2).
Inclined position between the orthogonal positions cannot be discriminated, since the intervals of the orthogonal positions are too close to each other. For all initial positions the expectation is (almost) in the middle of the interval.

higher as second guess. If either estimates head position, the guessing process needs to be refined. In this second step, we compare the current sensor data with the expectation of the acceleration axis with the most unlike expectation for the positions being involved.

This classification of the head's position is carried out by the procedure `sensePosition`. With the parameter `stable`, a programmer can choose whether the classification is based upon the very last (`stable = false`) or the average of the last ten accelerator measurements (`stable = true`). The procedure returns the resulting position. The overall guessing algorithm is listed in Algorithm 1 on page 48 below.

As every robot that relies on sensor information, MiniDog6M must face the problem of category errors. In the case of standing up, this is not a hard problem, since the consequences are not too severe. If MiniDog6M is running and suddenly assumes a lying position, trying to stand up would cause the robot dog to fall down. If the robot is lying down, but estimates the wrong position, it either starts running (if *Stand* is assumed) or tries to stand up and will most probably fail. The result in either case is that the robot dog is still lying at the end. Consequently, the robot has to try once more. Moreover, there is no risk for the physical structure of the robot, since its primitives are designed to prevent harmful postures. The classification results based on the averages of the last ten accelerometer values is shown in Table 3.4 on the following page below. The rows are labeled with the actual position and the columns with the estimated position. Each cell contains the probability of the corresponding pairs. The main diagonal denotes the prob-

Positions	0	1	2	3	4	5
Axis 0:						
Minimum:	-2	35	-39	-4	26	5
Expectation:	13	51	-23	12	3	23
Maximum:	35	67	-3	29	30	46
Axis 1:						
Minimum:	15	-11	-14	-48	-13	-7
Expectation:	30	2	-3	-32	4	7
Maximum:	49	16	13	-14	25	26
Axis 2:						
Minimum:	14	-18	-21	-53	-22	-12
Expectation:	29	-2	-8	-38	0	3
Maximum:	45	14	9	-18	22	25

Table 3.3.: Sensor data of the accelerometer in basic positions (Interval is visualised in Figure 3.4).
Inclined position between the orthogonal positions cannot be discriminated, since the intervals of the orthogonal positions are too close to each other. For all initial positions the expectation is (almost) in the middle of the interval.

abilities for correct classification, the others signify the probabilities for the respective misclassifications.

P_{est} / P_{act}	0	1	2	3	4	5
0	0.980	0.000	0.000	0.000	0.015	0.005
1	0.000	1.000	0.000	0.000	0.000	0.000
2	0.000	0.000	0.980	0.000	0.020	0.000
3	0.000	0.000	0.000	1.000	0.000	0.000
4	0.029	0.201	0.000	0.000	0.601	0.169
5	0.023	0.013	0.000	0.000	0.134	0.830

Table 3.4.: Classification results on test data
P_{act}: actual position (rows)
P_{est}: estimated position (columns)
The main diagonal denotes the probabilities for correct classification, the others signify the probabilities for the respective misclassifications.
If head position is estimated, the guessing process needs to be refined.

It is understood that apart from the position of the head, the position of the motors is also of interest. Since they cannot be read directly from the servos, an efferent copy of the motor command is stored in a global variable (though it is currently not needed).

The next subsection describes the standing up trajectories found in first pre-programmed experiments.

3.3.2. Trajectories of standing up

During a variety of pre-programmed experiments, several standing up trajectories were found. A model of those trajectories is illustrated in Figure 3.6 on page 49. The dots marked with the basic positions represent the possible initial states.

The red trajectories indicate dynamic motions exploiting the body properties. In the upper case, inertia is used to swing from right to left by rolling over on the back. In the lower example, gravity pulls the dog down on its belly.

The dashed arrow marks gradual movement which means introducing four sub-goal positions between the current and mid position. The reason why this humble way to slow down the motion was considered, is that the robot dog cants over again if the legs return directly this fast. This reduction is necessary only in this special case because the uneven angular range of the legs (and thus unequal starting points) produce unequal momentums that do not outweigh each other. Reducing speed also means downsizing the momentums generated during the move so that they can be absorbed by the dynamics of the body.

The afore mentioned imbalance of motor range is the reason that standing up from mirrored positions does not result in mirrored movements. Thus, for instance, a successful sequence for standing up from the left cannot be transformed into an equally successful sequence for the right. The irregular diagram results from this morphological constraint.

In many cases, the directionality of the arrows can be revised, but, due to asymetrical dynamics resulting from imbalanced motor range, this does not hold for all of them. Thus rolling over on the back from left to right hand side is just as impossible as the reversion of the red arrows.

Anyhow, the selected set of trajectories, as shown in Figure 3.6 on page 49, is certainly incomplete, because neither static trajectories nor trajectories emerging from the dynamic interchange (red transitions) can be fully overviewed.

The controller that implements these results is described in the next subsection.

3.3.3. Controller algorithm

For our locomotion task we decided on a simple cyclic control strategy. The dog runs until it stumbles and falls. Then it stops the running motion, stands up and resumes its way.

At the beginning - after setting up the serial communication - the dog is initialised by setting all motors to mid position. After sensing the position, the respective action is carried out. At the moment, this action is “running” as long as the robot remains in an upright position or “standing up” otherwise.

A classification of the position on the raw accelerometer inputs takes place after each controller step. To take misclassification into account, the running motion is only stopped

after perceiving an assumed position different from *'Stand'* for more than `triggermax` times.

Since fixed behaviours such as the standing up trajectories in Figure 3.6 on page 49 require a predefined starting point, the robot must adopt a distinct posture before "standing up" is called. So first of all the robot must stop the hopping behaviour and bring the servos to the initial position, if MiniDog6M's sensors state that it is lying down. After stopping the motion, a more reliable classification on the average of ten accelerometer measurements is performed. The result triggers the appropriate routines for standing up from the respective posture or running if *'Stand'* is assumed (due to current or earlier misclassification).

To assure that each motor's goal position is reached, every controller signal (except during running and trotting) is repeated `NumTicks` times. A subsequent guess of the new position proves success respectively failure. Measuring *'Stand'* the robot will resume his way, else another try is initiated from the current position.

As already mentioned, there are different solutions for most of the initial positions. At present, the next step among several possibilities is chosen according to preset probabilities. Therefore the robot will eventually be able to stand up even if a distinct movement is blocked (for instance by an object), since after several attempts, it will end up in a behaviour that does not need to carry out the impracticable move.

The resulting algorithm is listed in Algorithm 2 on page 50 below.

The next section gives an insight into the robustness of our solutions in inclined environment.

3.4. Standing up in a sloped environment

In this section, the pre-programmed behaviours are tested on their capability to be tolerant against minor, medium and steep inclinations.

First vital experiments with different ground angles showed that the robot dog can still get up in territories with relatively steep incline, but only from particular postures. Anyhow, the selected set of trajectories, as shown in Figure 3.6 on page 49, is certainly incomplete, because neither static trajectories nor trajectories emerging from the dynamic interchange (red transitions) can be fully overviewed. Whether the quadruped can stand up or not depends purely on the position of the head. Due to its heavy head and the high centre of mass, the robot fails even in flattish terrain if being in a disadvantageous posture. We made efforts examining eight designated initial positions in combination with three different ground angles. The qualitative results of ten trials are shown in Figure 3.5 on the facing page. Here '+' means that each of our test succeeded, none in case of '-' and with alternating success if marked with '0'.Anyhow, the selected set of trajectories, as shown in Figure 3.6 on page 49, is certainly incomplete, because neither static trajectories nor trajectories emerging from the dynamic interchange (red transitions) can be fully overviewed.

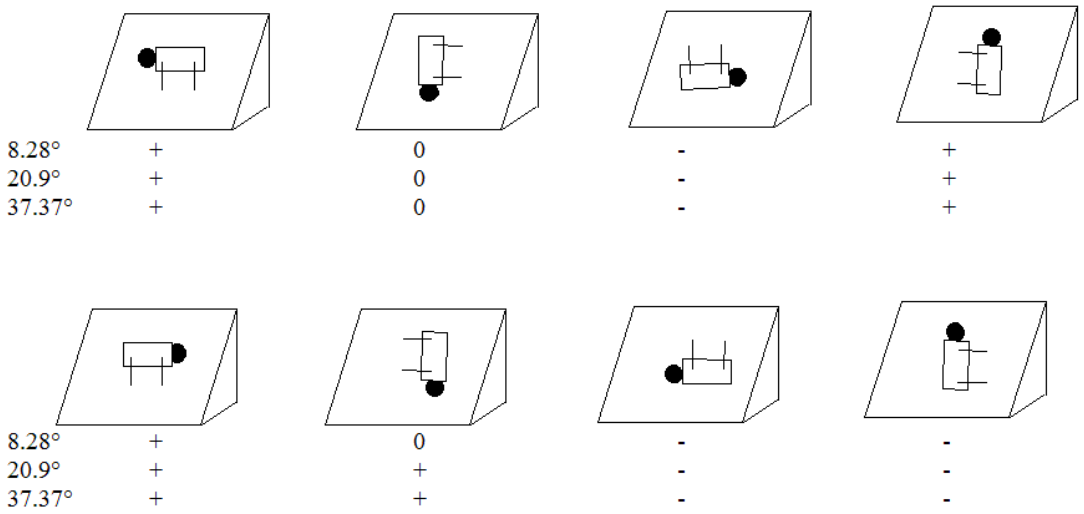


Figure 3.5.: Successful standing up with different ground angles in ten trials.
Success or not heavily depends on position of the head.
+: Each test succeeded
-: No test succeeded
0: Some tests succeeded

Unfortunately, not all positions and ground angles can be marked with '+'. Partly succeeding combinations always arose from too high momentums which in some cases overthrew the quadruped after standing up successfully. In these special cases, slower movements might help, but regarding combinations that always fail, a lower centre of mass or a higher behavioural diversity would be much more promising.

In the next chapter, we will work out an assortment of motor primitives and further establish a general means to evaluate them in regard of the frequencies they assign to the motors.

Algorithm 1 Position classification with `sensePosition`
`stable`: Parameter (`false`: classification is based upon the latest; `true`: classification is based upon the average of the last ten sensor values).
i∈{0, 1, 2}: Index of acceleration axis, j∈{0..5}: Position index
E(axis[i] for j): expectation of axis i for position j

```
1. if (stable)
     initialise average[0], average[1] and average[2]
       with average of next ten accelerometer values;
2. else
     initialise average[0], average[1] and average[2]
       with next accelerometer values;
3. for all j
       count[j] = ∑_{i=0...2} abs(average[i]- E(axis[i]
       for j))
4. First_guess = k with count[k] = min_{all j}count[j]
5. Second_guess = n with
       count[k] ≤ count[n] ≤ min_{all j\{n;k}}count[j]
6. if (First_guess == Head)
     if (Second_guess == Stand && count[Stand]==0)
       for i = 1..2
         temp = abs(E(axis[i] for Head) - average[i]);
         temp2 = abs(E(axis[i] for Stand)- average[i]);
         if (temp < temp2)
           indication for Stand;
         else
           indication for Head;
       if (more indications for Stand)
         First_guess = Stand
    if (Second_guess == Left && count[Left]==0
      && average[1] < critical value)
      First_guess = Left
    if (Second_guess == Left && count[Left]==0
      && for all i: average[i] < critical value)
      First_guess = Back

7.  if (First_guess == Head && Second_guess == Sit
      || vice versa)
      if  (abs(average[0]- E(axis[0] for Sit))
        < abs(average[0]- E(axis[0] for Head)))
        First_guess = Sit
8.  if (First_guess == Head && Second_guess == Right
      || vice versa)
      if  (abs(average[0]- E(axis[0] for Right))
        < abs(average[0]- E(axis[0] for Head))
        First_guess = Right
9. return First_guess;
```

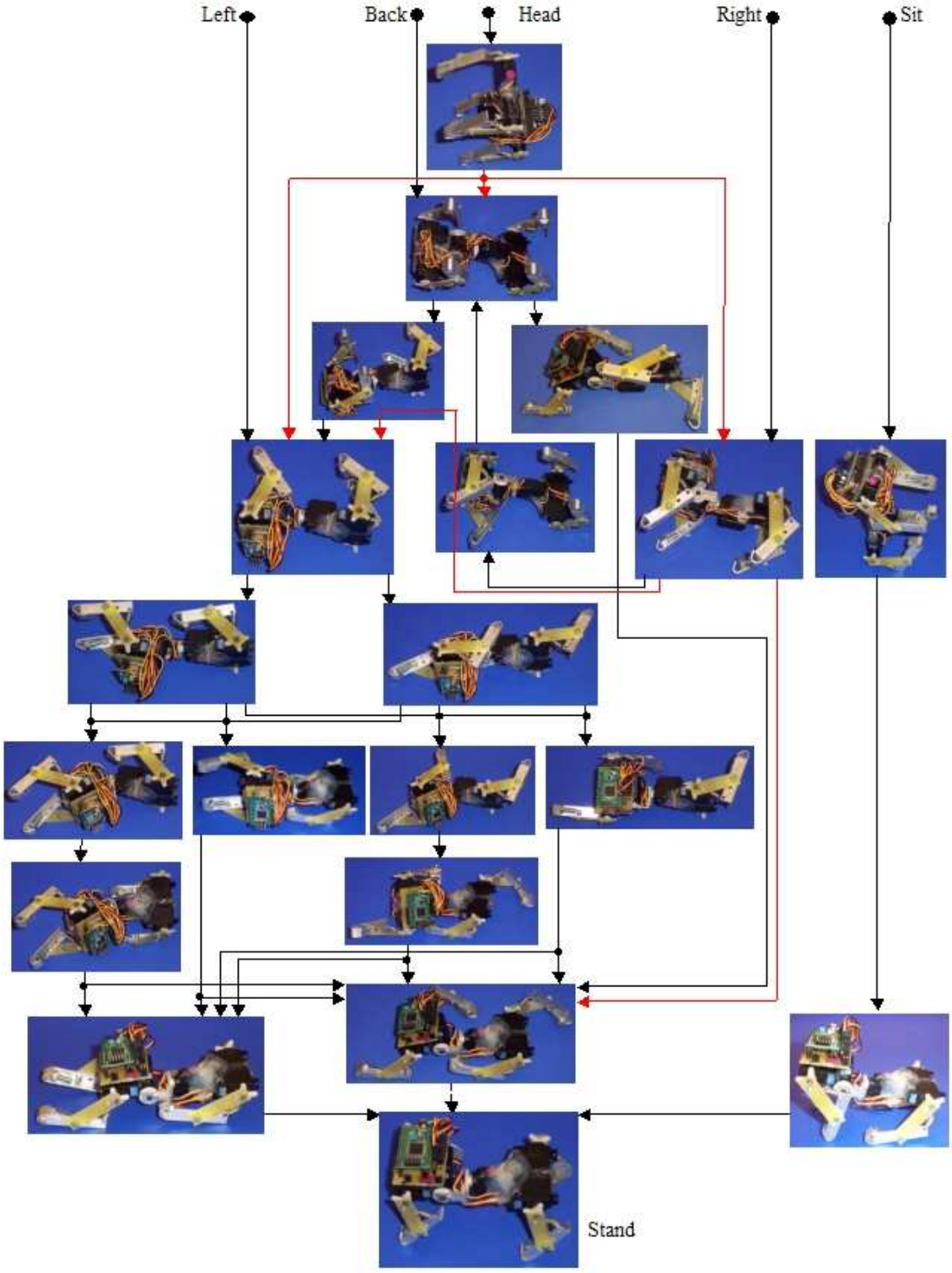


Figure 3.6.: Standing up trajectories found
This set was found during a variety of pre-programmed experiments, but is certainly far beyond completeness.
Dots: Possible initial states
Red trajectories: Dynamic motions exploiting the body properties
Dashed arrows: Gradual movement

Algorithm 2 Standing up controller

```
Initialise random number generator
Initialise probabilities to select one trajectory
    out of several possibilities
Initialise communication
Initialise MiniDog6M (set all motors to position 128)
While !(Program canceled) {
  Position = sensePosition(true);
  if (Position == Stand)
    do
      run();
    while ((stop < triggermax) && !(Program canceled));
    Gait = Still;
  else
    switch (Position) {
      case Right:
        if (probability < marginal value)
          trigger one trajectory;
        else
          trigger another;
        break;
      case Back:
        if (probability < marginal value)
          trigger one trajectory;
        else
          trigger another;
        break;
      case Left:
        if (probability < marginal value)
          trigger one trajectory;
        else
          trigger another;
        break;
      case Head:
        trigger standing up;
        break;
      case Sit:
        trigger standing up;
        break;
      default:
        run();
    }
}
Close communication;
```

4. Generation and evaluation of motor primitives

In this section, we will derive a first set of motor primitives and establish an abstract task and platform independent measure to categorise and evaluate them. In this context, six exemplary vocabularies will be nominated for further studies.

4.1. Deriving a first vocabulary

In this section, a first set of motor primitives which is derived fromth real world experiment will be worked out.

Analysing the pre-programmed standing up sequences, we find that all of them basically consist of nine distinct sets of motor positions as shown in Table 4.1 on the following page. Each line represents one motor primitive each of which can be defined by the goal positions of all motors. With goal position we mean the very position where a motor ends up after carrying out the current motor primitive. A value of zero denotes middle position. The 'max/min' notation refers to frontmost/backmost respectively rightmost/leftmost motor position. In ODE, these parameters are defined as `dParamHiStop`/`dParamLoStop` for each motor individually. 'x' designates "Don't care", which means that the respective motor is not considered in this motor primitive. "Don't care" can be thought of as a special stop symbol which causes a motor to remain in its current position. Therefore, the resulting posture is no longer determined only by the current motor primitive. The position of the unspecified motors depends on the direct predecessor(s) of the currently executed primitive. This means that their position depends on the last motor primitive which explicitly specified and set these motors in the past. Consequently, the current motor positions are the result of mixing up two or more motor primitives. Thus the recent history of selected components must be taken into account to get the current angles of all six motors.

First of all, we have to adapt the original motor primitives to the principle of sinusoidal control. Therefore, the parameter motorposition is substituted by frequency, amplitude

	FrontLeft	FrontRight	HindLeft	HindRight	Bend	Twist
Init	0	0	0	0	0	0
Bend	x	x	x	x	max/min	x
Twist	x	x	x	x	x	max/min
Legs Equal	max/min	max/min	max/min	max/min	0	0
Legs Oppos.	max/min	min/max	max/min	min/max	0	0

Table 4.1.: Set of motor positions defining each motor primitive.
+/- : frontmost / backmost respectively rightmost / leftmost motor position
x: Don't care

and offset. As all other components, except *Run*, define offset as zero and amplitude as maximum motor amplitude, only the frequency discriminates the respective component. Since the duration is also identical for all our primitives, the goal position purely depends on the frequency. Hence if different motors operate at the same frequency, their goal position is equal[1] and the corresponding legs move in parallel. Available frequencies are f_{Run}, 0 and $\pm f$ where

$$f = \frac{1}{NumTicks}, \tag{4.1}$$

so that the leg swings between maximum and zero position during one action period. Since, sinus is an odd function, it only depends on the sign of the corresponding frequency, whether a motor ends up in minimum or maximum position. This cohesion can be expressed as follows:

$$\sin(f \cdot t) = -\sin(-f \cdot t) \Leftrightarrow -\sin(f \cdot t) = \sin(-f \cdot t) \text{ where } -\frac{\pi}{2} \leq f \leq \frac{\pi}{2} \tag{4.2}$$

As a consequence, the legs swing forward if $f > 0$ and backward if $f < 0$. For $f = 0$ the leg return to mid position. The newly derived motor primitives are listed in Table 4.2 on the next page.

On that basis, MiniDog6M should be capable of finding proper sequences on its own. In the following, "frequency" and "goal position" are used synonymously.

The next section elaborates a criterion to rate motor primitives as well as entire vocabularies.

4.2. Evaluation of vocabularies

In this section, we establish a general means to assess individual motor primitives and sets of these by examining the symmetry and flexibility of each component. The meaning of symmetry and flexibility will be elaborated in the following.

[1] The term "equal" instead of "identical" is used on purpose, since maximum respectively minimum motor angle has the same quality for all motors, but does not refer to excately the same value.

	FrontLeft	FrontRight	HindLeft	HindRight	Bend	Twist
Run	f_{Run}	f_{Run}	f_{Run}	f_{Run}	0	0
Init	0	0	0	0	0	0
Bend0	x	x	x	x	x	max/min
Bend1	max/min	max/min	max/min	max/min	0	0
Twist0	x	x	x	x	x	$-f$
Twist1	x	x	x	x	x	f
LegsEqual0	$-f$	$-f$	$-f$	$-f$	0	0
LegsEqual1	f	$+f$	f	f	0	0
LegsOpposite0	f	$-f$	f	$-f$	0	0
LegsOpposite1	$-f$	$+f$	$-f$	f	0	0

Table 4.2.: Adapted motor primitives for sinusoidal control.

4.2.1. Flexibility-Index

Using the term flexibility as the antonym of stiffness implies that motor primitives bear potential for adaptation. In this sense, we adapt motor primitives by introducing the afore mentioned “Don’t care” symbol. In this connotation, the exemplary vocabularies can be divided into two groups, A and B. Motor primitives in group A always specify each motor position, whereas group B allows the just specified type of uncertainty. To make this unambiguous let us construct an example.

Let 1..5 be motor primitives of type A. Here +/- specifies the maximal/minimal motor angle.

1:	-	-	-	-	+	-
2:	+	+	+	+	-	+
3:	+	-	+	-	-	-
4:	-	-	+	+	-	-
5:	+	+	+	+	+	+

Then the goal posture[2] implied by the following action sequences is always the same.

1	2	3	4	5	$\rightarrow$	+	+	+	+	+	+	$(\hat{=}5)$
1	4	5	3	5	$\rightarrow$	+	+	+	+	+	+	$(\hat{=}5)$
1	5				$\rightarrow$	+	+	+	+	+	+	$(\hat{=}5)$
5					$\rightarrow$	+	+	+	+	+	+	$(\hat{=}5)$
3	3	3	3	5	$\rightarrow$	+	+	+	+	+	+	$(\hat{=}5)$

[2]To be precise that is the posture specified by motor primitive 5 as this is the common finish of all sequences.

Considering group B exemplary motor primitives might look as follows:

1:	x	x	-	-	+	x
2:	+	+	+	+	x	+
3:	+	x	+	-	x	x
4:	+	+	+	+	+	+
5:	-	-	x	x	-	-

where x designates stop/Don't-Care.

4	1	5	$\rightarrow$	-	-	-	-	-	-
4	5		$\rightarrow$	-	-	+	+	-	-
4	3	5	$\rightarrow$	-	-	+	-	-	-

Other than the example of type A, each sequence ends up in a different posture.

To honour this uncertainty, we introduce a special Flexibility-Index Flx that represents the ratio between the actual amount and the theoretically maximum of Don't-Care terms.

$$Flx = \frac{\text{actual amount of } x}{\text{max amount of } x} \tag{4.3}$$

Note, that the number of Don't-Care terms in single motor primitive lies between 0 and 6, but the mean of nine different primitives cannot exceed a maximum of 5.11.

In contrast to other architectures, that have to deal with concurrent behaviours nominated by different behaviours or control levels (as BISAM), MiniDog6M only enables the selection of one action sequence or primitive. The combination of motor primitives result as just explained through Don't-Care terms. Consequently, the currently selected primitive can only be mixed with primitives selected in the near past and not with primitives selected by concurrent behaviours. The reason for this lies within the lack of coexistence. In the current version of MiniDog6M, we consider only sequential and not parallel behaviours, namely running and standing up. For now, it is easier to stick to our simple assignment since we want to evaluate the usability of vocabularies for a given task.

How primitives that serve concurrent duties can be merged to achieve the best possible robot performance heavily depends on the semantics of the articulated behaviours and is therefore not considered here.

4.2.2. Coherence-Index

A second criterion for action primitives is its inner correlation respectively homogenity of a posture. Herefore we define the state of all motors in mid position as root posture (motor primitive: *init*). The similarity to this root posture and the symmetry of motor positions is

expressed by the Coherence-Index *Co* which is elaborated in the following[3].

First, we divide the motors in "clusters of interest" attributing the fact that certain motors serve different purposes regarding their position and effective direction within the robot. For MiniDog6M we identify three groups - namely leg motors, bend motor and twist motor.

As bend and twist motors build a class of their own, their goal position needs to be compared to the root posture only. Consequently, mid position is judged with correlation 1, whereas each motor finding itself in minimal and maximal motor angle has correlation 0. Don't-Care is estimated with 0.33, since the probability to reach 0 is one third. This leads us to the following spinal Coherence-Index values listed in Table 4.3. "+/-" denotes minimum or maximum angle, "0" stands for mid position and "x" signifies Don't-Care.

Bend/Twist	+/-	x	0.0						
Co(Bend/Twist)	0.0	0.33	1.0						
Spine Bend	+/-	x	+/-	x	+/-	0.0	x	0.0	0.0
Twist	+/-	+/-	x	x	0.0	+/-	0.0	x	0.0
Co(Spine)	0.0	0.165	0.165	0.33	0.5	0.5	0.665	0.665	1.0

Table 4.3.: Coherence-Index Co(..) for spinal motors
0: mid position
+/-: minimum or maximum angle
x: Don't-Care

Since the legs' group contains more than one motor, this assignment is not enough to criticise the combination of motor angles. Hence, we regard this case of one motor per group as special case and extend our measure for groups with more than one motor. In doing so, we regard symmetry the primary criterion and mid position as secondary one. Symmetry in this connotation denotes parallel movement of front, hind, left and/or right hand legs.

For our quadruped, we only distinguish four basic categories of symmetry. Either of these is of equal quality. These categories, a detailed coherence value and probability distribution for a single motor primitive can be found in Table 4.4 on the next page. Here '0' stands for mid position, 'x' allows each angle and '*' stands for minimum or maximum. The circle denotes parallel movement. Note that the orientation of the robot is not given, so the head may be set on either side. Hence an encircled pair of legs can be deemed front, hind, left or right hand, without loss of generality.

The main categories are subdivided with distance to mid position, which means that we additionally increase the Coherence-Index if the parallel legs are in mid position. Having to consider parallelism, the probability distribution for Don't-Care in the leg motors' group is much more complex than for the spine.

[3] In general, the root posture can also be starting posture, home posture or any other posture which is of particular interest.

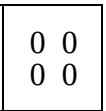
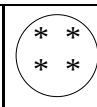
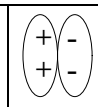
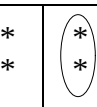
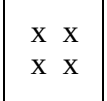
Main category							
Sub category							
Co(legs)	1.0	0.833	0.667	0.5	0.333	0.167	0
probability	0.01	0.03	0.1	0.05	0.22	0.38	0.21

Table 4.4.: Coherence index and probability distribution for each categories of symmetry of the legs
"0": mid position
"x": allows each angle
"*": minimum or maximum angle
The circle denotes parallel movement of encircled legs. Head may be attached on either side

It is understood that this general probability distribution must be concretised in reference to the actual vocabulary.

Merging the Coherence-Index of spine and legs, we weight the corresponding values in ratio of the number of motors. Just as we did for the spine.

$$Co(motor\,primitive) = \frac{1}{n_{total}} \cdot \sum_{all\,groups} Co(group) \cdot n_{grp} \tag{4.4}$$

with n_{total} being the number of motors and n_{grp} the number of motors in the group.

In our case study, this means

$$\begin{aligned} Co(motor\,primitive) &= \frac{Co(bend) + Co(twist) + 4 \cdot Co(legs)}{6} \\ &= \frac{2 \cdot Co(spine) + 4 \cdot Co(legs)}{6} \end{aligned} \tag{4.5}$$

The Coherence-Index of an entire vocabulary as set of motor primitives is defined as:

$$Co(vocabulary) = \frac{\sum_{all\,motor\,primitives} Co(motor\,primitive)}{max} \tag{4.6}$$

with max being the theoretical maximum of accumulated coherence of all primitives.

The different sets of motor primitives that will be discussed in our ongoing experiments are nominated and evaluated in the next section.

4.3. Choice of further vocabularies

As we cannot evaluate all possible vocabularies, six exemplary sets were chosen in this section to be investigated in this thesis.

We decided on three sets belonging to group A and three to group B. They are listed in Appendix B (group A in section B.1, group B in section B.2). Obviously, *Run* must be part of all of them. We stick to a total of ten actions, so that we can compare how the selected primitives influence behavioural diversity, robustness and learning progress. It is understood that using less actions will automatically accelerate the learning progress and most likely reduce the behavioural diversity. Thus the size of a distinct vocabulary is considered a trade-off between fast learning suitable for real world online learning and maximum behavioural diversity.

The vocabularies partly overlap in order to investigate how the significance of single components will change in dependency of the accompanying primitives. To overview the effect of similar vocabularies, one with and the other without Don't-Care, the sets can be paired as follows: 1 and 4, 2 and 5, 3 and 6. The original vocabulary that was worked out in section 4.1 is number 4.

The Flexibility-Index and the Coherence-Index should also be calculated for each task individually. As the ten primitives can be divided into two groups, namely locomotion (*Run)* and standing up (1-9), the subsequent tables only consider standing up. The analysis of the vocabularies is listed in table 4.5, and the individual rating of single primitives in table 4.6 on the next page.

	Vocabulary	Flx	Co
Group A	1	0.00	0.92
	2	0.00	0.86
	3	0.00	0.68
Group B	4	0.43	0.78
	5	0.24	0.74
	6	0.20	0.59

Table 4.5.: Coherence-Index and Flexibility-Index of vocabularies

In order to enable further experiments with different morphologies, we will proceed with a simulated version of MiniDog6M. Its implementation as well as the simulation platform is introduced in the subsequent chapter.

	1		2		3		4		5		6	
	Co	Flx	Co	Flx	Co	Flx	Co	Flx	Co	Flx	Co	Flx
1	1.00	0	0.79	0	0.71	0	1.00	0	0.79	0	0.71	0
2	0.93	0	0.64	0	0.71	0	0.28	83	0.64	0	0.71	0
3	0.93	0	0.71	0	0.29	0	0.28	83	0.42	0.67	0.29	0
4	0.93	0	0.93	0	0.21	0	0.28	83	0.55	0.83	0.21	0
5	0.93	0	0.71	0	0.93	0	0.28	83	0.71	0	0.25	0.83
6	0.86	0	0.86	0	0.50	0	0.86	0	0.86	0	0.50	0
7	0.86	0	1.00	0	0.86	0	0.86	0	1.00	0	0.22	0.67
8	0.57	0	0.57	0	0.86	0	0.57	0	0.57	0	0.57	0
9	0.57	0	0.93	0	0.57	0	0.57	0	0.41	0.33	0.86	0

Table 4.6.: Coherence-Index and Flexibility-Index of each motor primitive (first column) of all vocabularies (first line)
A clear distinction can be seen between group A with $Flx = 0$ and Group B with $Flx > 0$.

5. Virtual MiniDog6M for experiments in Simulation

Since we plan to investigate various vocabularies in combination with different morphologies which cannot be changed effortlessly in the real world, we transferred MiniDog6M into a simulated environment. The first section introduces a public library used for physically realistic simulation throughout our further studies. The model of MiniDog6M and the necessary changes of the controller are specified afterwards. Finally, the morphologies that will be investigated in the course of this thesis are selected.

5.1. Simulation with Open Dynamics Engine

In this section, the simulation platform for our experiments will be presented.

MiniDog6M was simulated in a physically realistic environment with the help of the Open Dynamics Engine (ODE), which is a free, industrial quality C++ library for simulating articulated rigid body dynamics [Smith 04]. Since it is fast, flexible and robust, with built-in collision detection, it is a comfortable way to simulate moving objects, such as legged creatures, in a virtual reality environment. In contrast to many other tools using only springy contacts of objects, ODE additionally supports hard contacts by declaring special non-penetration constraints for collision. Hereby speed and stability are emphasised over physical accuracy.

In ODE agents and objects must be defined as articulated structures which consist of rigid bodies of various shapes (box, sphere, ray, triangular mesh or capped cylinder) connected with joints of various types (ball-and-socket, hinge, slider, hinge-2, fixed, angular motor, universal).

Each body is determined by constant and dynamic properties. Constant properties are its point of reference which is settled on the position of the centre of mass, its total mass and its inertia matrix that describes how the body's mass is distributed around its centre. A body's motion is described as the dynamic properties of its point of reference, namely the

position vector, the orientation, the linear and angular velocity vector. The orientation is thereby represented either as quaternion or a rotation matrix.

Each joint is determined by the position of its anchor and a number of parameters controlling its geometry. Since all these properties are independent of the bodies they are attached to, it is possible for the bodies to be in positions where the joint constraints are not met.

Often the user is to blame for such a "joint error" by setting the position/orientation of one body without correctly setting the position/orientation of the other body, but this can also happen due to numerical errors. During the simulation, rounding errors can accumulate so that bodies drift apart. This drift can be reduced by setting the Error Reduction Parameter ERP, a value between 0 and 1. The author recommends to set $ERP = 0.1 \ldots 0.8$. The default is 0.2.

Another important parameter to achieve fault tolerance is Constraint Force Mixing CFM. A positive value of CFM takes the system away from any singularity and improves the factoriser accuracy by allowing the original constraint that produces the constraint force to be violated by an amount proportional to CFM times the restoring force that is needed to enforce the constraint. In other words the constraint will be softened as CFM increases. The author recommends to set $CFM = 10^{-9} \ldots 1.0$, while the default is 10^{-5} for single and 10^{-8} for double accuracy.

The equations to model a body's motion are derived from a Lagrange multiplier velocity based model according to Trinkle/Stewart and Anitescu/Potra. To simulate the creature(s) through time, a first order integrator is being used. It's fast, but not accurate enough for quantitative engineering. Each integration step advances the current time by a given step size, allows all joints to apply so called constraint forces to bring the bodies they affect back into correct alignment and finally adjusts the state of all rigid bodies for the new time value. Higher order integrators are planned.

Collisions between objects within the simulated environment are handled as follows: Before performing a world step, the collision detection functions must be called to determine the intersection areas. These functions return a list of contact points specified by its position in space, its surface normal vector and its penetration depth. A special contact joint is created for each contact point, supplied with extra knowledge about friction, springyness, spongyness etc and put into a joint group.

In fact, CFM and ERP can used to control its spongyness and springyness of a joint. In fact, the user can realise the same effect as any spring constant k_p and/or damping constant k_d by setting ERP and CFM in dependency of the step size h as follows:

$$ERP = \frac{h * k_p}{h * k_p + k_d} \tag{5.1}$$

$$CFM = \frac{1}{h * k_p + k_d} \tag{5.2}$$

Then a simulation step is taken. Afterwards, all contact joints are removed. Other collision detection libraries can be used instead of the built-in collision functions as long as

they provide the right sort of information in the contact point.

To speed up the simulation at the cost of accuracy, the special option “quickstep” can be chosen. Another possibility to reduce computational time is to disable all object that do not interfere with enabled bodies. As a consequence, they no longer will be updated.

The contact and friction model is a fast approximation to the Coloumb friction model, but originally based on the Dantzig LCP solver described by Baraff. The Coulomb friction model is a simple, but effective way to model friction at contact points by defining a friction cone to model the relationship between the normal and tangential forces at a contact point. If the total friction force vector is within the cone then the friction force is enough to prevent the contacting surfaces from sliding. If it is on the surface of the cone then that contacting surfaces move with respect to each other.

For graphical output, ODE includes the drawstuff library using OpenGL. To render 3D objects in a virtual environment, OpenGL only needs to know the objects plus additional information on camera position and light sources. These information are handled by drawstuff through the GLUT extension library of OpenGL.

The ODE library can be compiled as fast release or as slower debugging version. In the latter, function arguments are checked and many additional run-time tests are done. These tests ensure the internal consistency, but they are also the reasons for a big loss of performance. The current version is 0.5.

How ODE was used to create a realistic simulation of MiniDog6M, is explained in the subsequent section.

5.2. Model for MiniDog6M

In this section, the simulated version of MiniDog6M is described.

The simulated Minidog6M was modelled for ODE. A schematic and screenshot of the virtual robot are illustrated in Figure 5.1 on the following page.

The shape and weight distribution of the control area (yellow cylinder in schematic) is not specified in detail and is approximated with a centred capped cylinder. The legs (grey rectangles in schematic) are approximated with capped cylinder for the upper (light grey) and for the lower (dark grey) legs. The servos are modelled as boxes (white boxes in schematic). The spine links connecting the front and hind part of the robot are also represented as small boxes (light blue). The additional weights in the feet are modelled as small capped cylinders (green dots) within the lower legs. The robot dog in our simulation is not altogether identical with the physical model, but yields a good approximation.

The proportions of each component are listed in Table 5.1. Since the parameters in ODE should not be too small (approximately 1.0), we set $0.1 \mathrel{\hat{=}} 1$ cm for all linear and $0.1 \mathrel{\hat{=}}$ 10g for all mass dimensions.

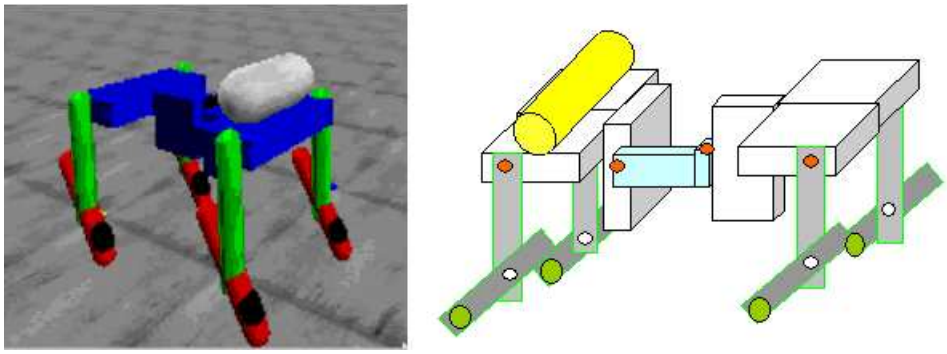


Figure 5.1.: Simulation screenshot (left) and schematic of MiniDog6M (right)
The control area, the legs and the feet are approximated with capped cylinders, spine links and servos with boxes. Actuation is implemented as flapping hinge joints (orange dots). The springs are implemented directly into the passive hinge joints (white dots).

The rigid parts of the body are connected with fixed joints, the upper legs as well as the spine are controlled and linked by hinge joints (orange dots in schematic). The springs are implemented directly in the lower hinge joints (white dots in schematic) by means of adjusting Constraint Force Mixing CFM and Error Reduction Parameter ERP as follows:

$$dParamStopERP = 1 + \frac{StepSize \cdot SprintConstant}{DampingConstant} \tag{5.3}$$

$$dParamStopCFM = \frac{1}{StepSize \cdot SpringConstant + DampingConstant} \tag{5.4}$$

The constants restricting the dynamics of the simulated dog robot are listed in Table 5.2 on page 64.

In ODE the ground is defined over its plane equation:

$$a \cdot x + b \cdot y + c \cdot z = d \tag{5.5}$$

The left hand parameters define the unit normal vector $(a\,b\,c)$ which stands orthogonal on the surface. For flat ground $(0\,0\,1)$ is fine, for incline α we use $\pm\cos\alpha$ for either a or b and $\sin\alpha$ for c. The remaining parameter d is zero throughout all experiment. The definition of the coordinate axes can be seen in Figure 5.2 on page 64

The CFM of the ODE world remains the default value, whereas ERP is raised to 0.8.

For reasons of reuse, encapsulation and information hiding, we decided to implement The simulated dog robot as C++ class `MiniDog6M`. To create an instance of `MiniDog6M`, a

Attribute	Value
Size of one servo	0.298·0.296·0.12
Mass of one servo	0.19
Radius of control area	0.12
Length of control area	0.296
Mass of the control area	0.55
Radius of the legs	0.05
Length of the upper legs	0.5
Mass of the upper legs	0.026
Length of the lower legs	0.6
Mass of the lower legs	0.102
Radius of one foot	0.05
Length of one foot	0.05
Mass of one foot	0.05
Height that the legs raise the chasis off the ground (in stance phase)	0.65
Size of spine link attached to servo bend	0.136·0.01·0.1
Size of spine link attached to servo bend	0.01·0.148·0.1
Mass of spine links	0.01

Table 5.1.: Linear and mass dimensions of elements building the simulated MiniDog6M $0.1 \mathrel{\hat{=}} 1$ cm for all linear and $0.1 \mathrel{\hat{=}} 10$g for all mass dimensions

user must not employ the standard constructor[1], but `MiniDog6M(dWorldID world, dSpaceID space, float StepSize)` instead. The parameters refer to the environment and the discretisation of time. This constructor creates a standing dog robot with the middle of the spinal link positioned over the margin of the x-y-plane.

The changes in control as well as the public procedures will be described in the next section.

5.3. Controller for simulated Minidog6M

In this section, the public methods of our base class will be defined. Further, MiniDog6M's controller will be adapted to fit the circumstances of simulation.

Since the virtual and the real MiniDog6M's controller, should be as similar as possible, we defined the same enumeration types for head position and gaits as introduced for the

[1] Thus the default constructor is declared `private`.

Attribute	Value
Springconstant	418
Dampingconstant	0.05
Angular range of spring driven joints	[30°; 60°]
Max force produced by one servo	5
Angular range of hind right servo	[-51°; 104.5°]
Angular range of hind left servo	[-55°; 92.5°]
Angular range of front right servo	[-51°; 108°]
Angular range of front left servo	[-50.5°; 90°]
Angular range of servo “bend”	[-65°; 62°]
Angular range of servo “twist”	[-72.5°; 73°]

Table 5.2.: Constants responsible for manoeuvrability

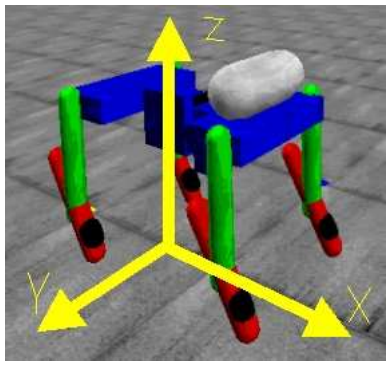


Figure 5.2.: Definition of coordinate axes in ODE

real world. The agent’s position and gait as well as the head posture, the current angle and frequency of each motor, can be read out with the respective get procedures.

The agent’s position is defined as the position of the head’s centre of mass and is delivered in form of a vector $(x\,y\,z)$ of real values by `GetDogPosition`.

The gait can be setwith `SetGait` respectively read with `GetGait`. To start running, trotting or backing up, the user has to call `Move`. These behaviour can be stopped if the gait is set to *Still* or if `Init` is called which stop the robot and brings all motors to mid position.

The acceleration sensor cannot be simulated physically realistically, but, at present, this is no problem since, even in the real world, only the pre-processed data in form of the agent’s position are presented to the controller. This effect can easily be reached in ODE since the absolute and relative position of the centre of mass of each rigid body and each joint anchor can be read out with `dGeomGetPosition` respectively `dJointGetHingeAnchor`. This is why we classify the position on the basis of height differences between

head, backside, control area, servo “bend”, joint anchors of upper front legs along z-axis. Basically, we also employ the algorithm 1 on page 48, but substitute the expectations of the acceleration values with the corresponding height differences in the orthogonal positions. Further, we leave out step 5 to 8, since there is no refinement necessary. The classification process can be kicked off with `SensePosition`. The resulting position can be read with `GetPosition`.

The angle of the hinge joints is indirectly controlled by its angular velocity `dParamVel`, instead of assigning values of 0..255 to the distinct positions (as it is in the servo motors). As it is in real motors, the target position is not always reached exactly in one step. Hence we implement a controller that constantly adjusts the joint error using the following equation:

$$dParamVel = TargetMotorAngle - CurrentMotorAngle \tag{5.6}$$

The resulting leg trajectory is qualitatively pictured in Figure 5.3.

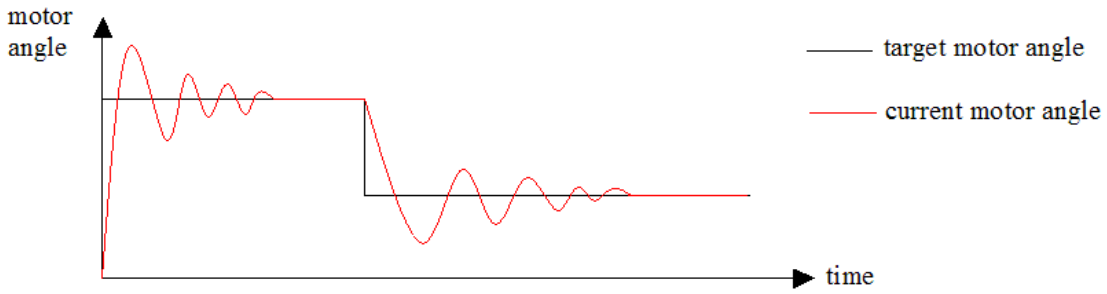


Figure 5.3.: Motor control according to equation 5.6
The controller constantly tries to nullify the joint error between current (red) and target (black) motor angle

The current motor angle as array of double values can be read with `GetCurrentMotorPos`. The controller frequency/amplitude of each motor can be read respectively set as array of double values with `GetCurrentMotorFreq / GetCurrentMotorAmpl` respectively `setControllerParameter / setMotorAmpl`.

The classical pre-programmed standing up trajectories are provided by `StandUp`.

For facility of inspection, the above mentioned public methods are summarised below:

`MiniDog6M()`	Standard constructor (do not use)
`MiniDog6M( dWorldID world, dSpaceID space, float StepSize)`	Constructor for `MiniDog6M`
`~MiniDog6M()`	Destructor for `MiniDog6M`
`void Init()`	Stops agent's motion and brings all motors into mid position
`void Move()`	Triggers locomotion with preset gait in `MiniDog6M::Gait`
`void SensePosition()`	Classifies position of the agent's head. Classification result is saved in `MiniDog6M::position`
`const dReal* GetDogPosition()`	Returns position of agent's head as vector of real values
`double* GetCurrentMotorPos()`	Returns current motor angle of all motors as vector of real values
`double* GetCurrentMotorAmpl()`	Returns current motor angle of all motors as vector of real values
`double* GetCurrentMotorFreq()`	Returns current motor angle of all motors as vector of real values
`Positions GetPosition()`	Returns position of dog robot's head
`Gaits GetGait()`	Returns current gait
`void SetGait(Gaits gait)`	Sets the agent's gait
`void setControllerParameter( double* freq)`	Sets motor frequencies as vector of real values
`void setMotorAmpl(double* Amp)`	Sets motor amplitudes as vector of real values
`void StandUp(bool &ready)`	Triggers pre-programmed standing up sequences

The motor primitives are also transferred into simulation. Like in the real dog each motor primitive is carried out `NumTicks`. The step size in all our experiments is set to 0.05.

In the next section we will select three different morphologies for our further studies.

5.4. Selection of different morphologies

In this section, we will point out further effects of morphology and on that basis select three different morphological variations for our ongoing studies.

Some central effects of morphology are already described in chapter 3, namely increased manoeuvrability through additional weights in the feet, the imbalance and directionality in Figure 3.6 on page 49 and the impact of inclined ground in section 3.4. Other plain

examples can easily be found in simulation. They which inspired us to investigate not only the consequences of different vocabularies, but also their dependency of different morphologies.

In one experiment, the shape of the head is changed into a cuboid. As soon as the robot dog lands or turns on its back, the weight of the head pulls it on the whole bearing area of the cuboid and the robot will not be able to free itself from that position. The agent gets stuck because its hind and legs cannot reach the ground anymore and because the pure rotational moment is not enough to turn over the edges of the head. This is a classical deadlock.

Moreover, the effect of rolling over on its back from one side to the other just by twisting the spine first in one and then in the opposite direction simply relies on the mismatch in the weight ratio of head and hind. If the weights were equal, the quadruped would only twist back and forth and end up in the same position it started from.

All these consequences do not result from changes in the controller strategy, but from more or less significant changes in the weight distribution, motor range and shape of the head. From now on we concentrate on the head because our present experiments imply that changing its shape has the largest impact on the standing up task. Thus we introduce three additional settings keeping up the original weight distribution, but with different forms of the head. The experimental morphologies with either control area in shape of a horizontal or vertical capped cylinder or a round head are pictured in Figure 5.4. We would like to survey more variations here, but further exploration goes beyond the scope of this thesis.

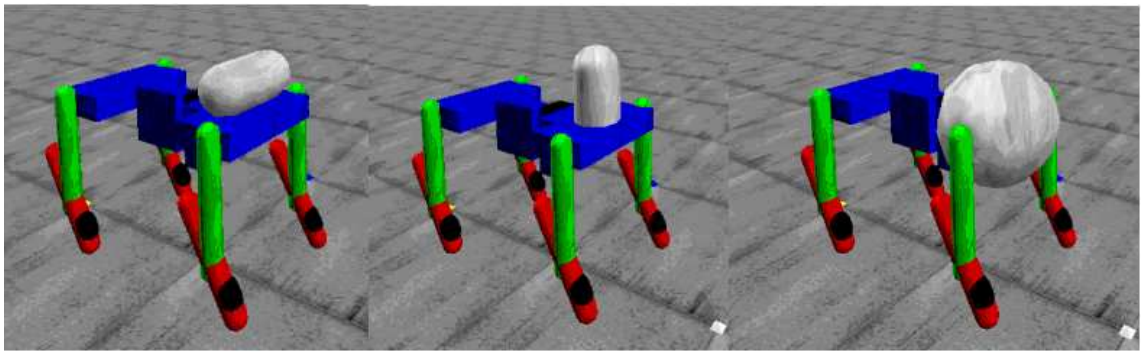


Figure 5.4.: Experimental morphologies (original, vertical, round)
Changes in the shape of the head have biggest impact on the agent's behaviour. Mass and centre of mass are equal in all versions.
Vertical: cannot roll over on the back
Round: cannot stand on the head, but rolls over on the back most easily

A plain consequence of this choice is the fact that the robot's head cannot roll over on the back with vertical head and that it cannot stand on the head with the round head.

Note that the original MiniDog6M can use each motor primitive in vocabulary 4 to stand up, whereas some of the actions in the other sets will be completely useless. Anyway, the eligibility of certain components may change dramatically as we alter the robot dog's morphology.

To investigate how the shape of the head affects the nature and amount of solutions, a measure for behavioural diversity and a full search algorithm to generate it are developed in the next chapter.

6. Generating and evaluation of behavioural diversity

This section describes the first row of experiments for the evaluation of the vocabularies selected above. First, we establish a general means to compare the behavioural diversity of different tasks or vocabularies. To overview the variety of legal standing up sequences on flat ground and inclines, we ran a full search simulation.

6.1. Measure for behavioural diversity

In this section, a measure which is independent of task, morphology and control concepts, is derived in order to evaluate and compare behavioural diversity.

6.1.1. An intuitive approach to behavioural diversity

Up to this day, researchers all over the world cannot agree on a single definition of the term intelligence. Despite the great variety of explanations, there is one central aspect in most of them and this is the generation of behavioural diversity. An organism that always exhibits the same behaviour is generally not considered an intelligent being. This concept holds for all levels of intelligence, for abstract thinking as well as for simple tasks, in humans and in all sorts of animals.

It is understood that behavioural diversity must always be seen in contemplation of the given task. Whether a person can open a bottle of beer in just one or 20 different ways is meaningless if he or she is sitting in front of a bottle of wine. In order to provide a base for comparing behavioural diversity related to the same task, we establish a measure that will be employed later on.

If we follow the different ways through Figure 3.6 on page 49, we can easily find multiple paths of equal or different length from each initial position to goal position. Regarding initial position *Left*, we find six sequences requiring five and also six sequences requiring

six time steps (disregarding loops). Providing the pictures with numbers top down and from left to right beginning with 0, these twelve sequences are:

4	→	8 or 9	→	10	→	14	→	16	→	19	(length: 6)
4	→	8 or 9	→	10	→	14	→	17	→	19	(length: 6)
4	→	8 or 9	→	11	→	16	→	19			(length: 5)
4	→	8 or 9	→	11	→	17	→	19			(length: 5)
4	→	8 or 9	→	12	→	15	→	17	→	19	(length: 6)
4	→	8 or 9	→	13	→	17	→	19	→		(length: 5)

A simple means to measure behavioural diversity could be the plain number of different sequences e.g. 12. It is obvious that 12 would be considered better than 3. Having a closer look we will find that it is not that easy.

Imagine two robots achieving the same goal. Robot A can choose between three different behaviours in order to succeed, whereas robot B manages five behaviours. Is robot B indeed preferable to A? Not always. To make a point let us consider the task of grasping an object on the basis of motor primitives such as grab with right hand, step towards the table, stretching arm out while standing on the tiptoes, bend wrist etc. which can be combined to the following sequences:

A:	1	→	3	→	5			
	4	→	6	→	5			
	7	→	2	→	3	→	1	
B:	1	→	4	→	5	→	2	
	1	→	4	→	5	→	3	
	1	→	4	→	5	→	7	
	1	→	4	→	6			
	2	→	4	→	6			

If, for any reason, it is impossible to carry out motor primitive 4, for instance because of a hindered joint, robot B would have to give up, whereas robot A could still get along and reach the goal position of its task.

Now picture an obstacle between the robot and the object. It is clear that robot A, though being defeated comparing the plain number of available solutions, will be more likely to succeed. Whereas robot B can only try to get around the obstacle with its final move (at least in four out of five possible choices), robot A can adapt much earlier e.g. step aside or take the other arm. Since adaptability is the most important reason for behavioural diversity, robot A would be better suited than B because of low correlation between the behaviours.

On the other hand, we can also think of an example where the advantage is on robot B, for instance, if one sequence is especially energy efficient or if special sub-goals can only be reached with particular parts of a sequence. Moreover, if the agent has to adapt in the

course of a particular (long) sequence, then robot A would have to revise everything and select a new sequence, whereas robot B can switch in the middle of sequence to another one which so far equals our first choice. Thus it would be desirable to maintain a balance between diversity and heterogeneity.

Moreover, a researcher should take into account the capability of a vocabulary to transfer a given problem or task into another, for instance standing up from the left to standing up from the right. So, if a robot is hindered by a wall or any other obstacle and thus cannot get up from its current position, it would be very useful to roll over to another position where it has enough legroom. Therefore, the more transfer capability is engaged, the better.

6.1.2. Behavioural-Diversity-Index BDI

Providing a quantitative scale unit to compare the behavioural diversity generated by different vocabularies, we derived a measure Behavioural-Diversity-Index BDI according to Algorithm 3 on the next page.

First, we will reduce the entirety of all behaviours to a base of significant behaviours. Thereby, all needless sequences, which consist of another (shorter) sequence plus a prefix of meaningless actions, will be deleted. We do this, since we can always lengthen a sequence by performing senseless actions beforehand. The remaining sequences are divided into groups in which all members either have the same beginning or the same ending. Further, these groups are divided in to subgroups of identical length. For each group a value called diversity factor D is calculated which expresses the heterogenity of the corresponding group. The BDI is the product of the total amount of legal sequences with the mean diversity factor.

Note that motor primitives involving Don't-Care must not be viewed in isolation but in context of their forerunners and hence must be substituted with its fully specified version. Each version counts as separate component.

One might argue that, under certain circumstances, equal weights of the non-member and the member contribution to equation 6.1 on the following page is inappropriate. For these cases we recommend to use equation 6.3 instead of equation 6.1. The additional weights W_{nonmbr} for the non-member part and W_{mbr} for the member part are calculated by equation 6.4 and equation 6.5 respectively.

$$\begin{aligned} D = {} & W_{nonmbr}(n) \cdot \left(1 - \frac{n_{grpmbr}}{n_{total}}\right) \\ & + W_{mbr}(n) \cdot \sum_{allgroups} \left(\frac{n_{submbr}}{n_{total}} \cdot \left(1 - \frac{n_{equal}}{l}\right)\right) \end{aligned} \tag{6.3}$$

$$W_{mbr} = \begin{cases} 1 & n < 1 \\ 1 + \sin\left(\frac{4n}{\pi}\right) & 1 \leq n < 2N \\ 2 & 2N \leq n \end{cases} \tag{6.4}$$

$$W_{nonmbr} = \begin{cases} 2 & n < 1 \\ 1 + \cos\left(\frac{4n}{\pi}\right) & 1 \leq n < 2N \\ 1 & 2N \leq n \end{cases} \tag{6.5}$$

Algorithm 3 Behavioural Diversity

1. Delete all sequences that consist of another (shorter) sequence plus some prefix.
2. Extract groups of sequences which either have the same beginning or the same ending. Each sequence may appear in more than one group.
3. Divide every group into subgroups of identical length.
4. Calculate the diversity factor D for every group as follows:

$$D = 1 - \frac{n_{grpmbr}}{n_{total}} + \sum_{all\ subgroups} \left(\frac{n_{submbr}}{n_{total}} \right) \cdot \left(1 - \frac{n_{equal}}{l} \right) \tag{6.1}$$

with

$n_{grpmbr} = \sum_{all\ subgroups} n_{submbr}$: number of group members

n_{submbr} : number subgroup members

n_{total} : total amount of legal sequences

n_{equal} : number of equal steps

l : length of sequence (=number of steps)

5. Calculate BDI as the product of the total amount of legal sequences with the mean diversity factor.

$$BDI = \frac{n_{total}}{n_{group}} \cdot \sum_{all\ groups} D \tag{6.2}$$

with

n_{group} :number of groups

Here n denotes the number of legal sequences produced by a distinct vocabulary. The resulting weight distribution is pictured in Figure 6.1. The threshold N can be chosen intuitively or as average amount of sequences over all vocabularies that we like to compare.

Another criterion can be the average length of the distinct sequences. Generally speaking: If you want a job done, the sooner the better. If a robot autonomously explores its environment, it is convenient to discover a maximum area before returning to the charging station. In our case, all motor primitives have the same duration respectively action period and therefore we measure the duration of the entire sequence by counting the steps. Nevertheless, we consider speed as non-functional design criterion and thus less important than diversity and total amount of sequences.

We calculate both, average length and behavioural diversity, separately for all initial position. Merging all information of one vocabulary, we weight the individual values with the probability of its initial position.

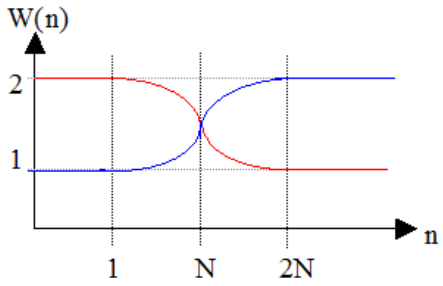


Figure 6.1.: Suggested weight distribution W_{mbr} (blue, eq. 6.4) and W_{nonmbr} (red, eq. 6.5 on page 71)
Recommended weight distribution if equal weights between the non-member and the member part of equation 6.1 on the facing page is inappropriate. The threshold N can be chosen intuitively or as average amount of sequences over all vocabularies.

6.1.3. Example

Let us look at the following example:

Vocabulary A

A_1: 1 → 3 → 8 → 5
A_2: 4 → 6 → 7 → 5
A_3: 7 → 2 → 3

Vocabulary B

B_1: 1 → 4 → 5 → 2
B_2: 1 → 4 → 5 → 3
B_3: 1 → 4 → 5 → 7
B_4: 1 → 4 → 6
B_5: 2 → 4 → 6
B_6: 5 → 7 → 2 → 4 → 6

Calculating BDI:

1. Delete sequences:
   Robot A: NoOp
   $\Rightarrow n_{total} = 3$
   Robot B: Delete sequence B_6
   $\Rightarrow n_{total} = 5$

2. Extracting groups:
   Robot A: $G_1 = \{A_1, A_2\} \in \{\ldots \rightarrow 5\}$
   $\Rightarrow n_{grpmbr} = 2,\ n_{equal} = 1$
   Robot B: $G_1 = \{B_1, B_2, B_3\} \in \{\rightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow \ldots\}$
   $\Rightarrow n_{grpmbr} = 3,\ n_{equal} = 3$
   $G_2 = \{B_1, B_2, B_3, B_4\} \in \{\rightarrow 1 \rightarrow 4 \rightarrow \ldots\}$
   $\Rightarrow n_{grpmbr} = 4,\ n_{equal} = 2$
   $G_3 = \{B_4, B_5\} \in \{\ldots \rightarrow 4 \rightarrow 6\}$
   $\Rightarrow n_{grpmbr} = 2,\ n_{equal} = 2$

3. Divide into subgroups:
   Robot A: G_1: no subgroups necessary
   $\Rightarrow n_{submbr} = 2,\ l = 3$
   Robot B: G_1: no subgroups necessary
   $\Rightarrow n_{submbr} = 3,\ l = 4$
   $G_{2.1} = \{B_1, B_2, B_3\}$
   $\Rightarrow n_{submbr} = 3,\ l = 4$
   $G_{2.2} = \{B_4\}$
   $\Rightarrow n_{submbr} = 1,\ l = 3$
   G_3: no subgroups necessary
   $\Rightarrow n_{submbr} = 2,\ l = 3$

4. Calculate D:
   Robot A: $D = 1 - \frac{2}{3} + \frac{2}{3} \cdot \left(1 - \frac{1}{4}\right) \approx 0.83$
   Robot B: $D_1 = 1 - \frac{3}{5} + \frac{3}{5} \cdot \left(1 - \frac{3}{4}\right) = 0.55$
   $D_2 = 1 - \frac{4}{5} + \frac{3}{5} \cdot \left(1 - \frac{2}{4}\right) + \frac{1}{5} \cdot \left(1 - \frac{2}{3}\right) \approx 0.57$
   $D_3 = 1 - \frac{2}{5} + \frac{2}{5} \cdot \left(1 - \frac{2}{3}\right) \approx 0.73$

5. Calculate BDI :
   Robot A: $BDI = n_{total} \cdot D = 3 \cdot 0.83 = 2.49$
   Robot B: $BDI = n_{total} \cdot \frac{D_1 + D_2 + D_3}{3} \approx 5 \cdot 0.62 \approx 3.08$

Calculating the average length:
Robot A:

$$\frac{\parallel A_1 \parallel + \parallel A_2 \parallel + \parallel A_3 \parallel}{n_{total}} = \frac{4 + 4 + 3}{3} \approx 3.67$$

Robot B:

$$\frac{\parallel B_1 \parallel + \parallel B_2 \parallel + \parallel B_3 \parallel + \parallel B_4 \parallel + \parallel B_5 \parallel}{n_{total}} = \frac{4 + 4 + 4 + 3 + 3}{5} = 4.2$$

In the next section, we create behavioural diversity with a full search trial and error algorithm.

6.2. Full search for behavioural diversity

As a tool for overlooking the behavioural diversity generated by each combination of head, vocabulary and ground angle, we set up an algorithm that performs a structured walk through the entire search space and which will be described in this section.

As the longest sequence from one of the basic positions to stand takes five steps in the real world, we decided to give the simulated robot six steps to try. After these six steps, the controller derives the next sequence, execute it and afterwards evaluates its result.

In order calculate the next sequence, we take the whole sequence as integer which is increased stepwise. Seeing that this results in six to the power of ten possibilities, we considered a reduction of the search space in five steps:

1. The last action must be Init(), since this is the goal motor position.
2. The first action must never be Init(), since this is also the starting motor position and hence would be a wasted step.
3. Never the same action twice in a row, as this would also be a waste of time.
4. When an action sequence is only differentiated from a previously successful sequence by the very last statements of that successful sequence, there is no further need to investigate the current sequence further. Executing them would bring no new insights.
5. A currently executed action sequence is skipped when its current step results in one of the basic position plus all motors in position zero.

If a sequence reaches position is *Stand* with all motors in mid position, success will be encountered and the sequence will be logged in the file `Ways2Stand.txt`. Leading to one of the other basic positions execution is also stopped, since following steps will be acquired by the learning process from that position. Action sequences that denote transitions between the basic positions are recorded in `BasicTransitions.txt`.

The resulting files for one initial position have the following structure:

```
8 0              1 0 --> basic position: 1
1 3 0            1 7 --> basic position: 2
1 4 3            1 2 0 --> basic position: 1
1 9 5            2 3 0 --> basic position: 1
1 5 7            4 5 0 --> basic position: 5
1 4 3 0          4 6 0 --> basic position: 1
1 5 3 1          4 8 0 --> basic position: 1
1 9 3 1          4 9 0 --> basic position: 4
1 3 4 7          4 2 7 --> basic position: 1
1 8 4 8          5 3 7 --> basic position: 5
2 9 4 0          7 5 7 --> basic position: 5
2 3 5 4          7 6 7 --> basic position: 1
2 4 5 0          7 8 7 --> basic position: 3
2 9 6 0          7 9 7 --> basic position: 3
```

The left column despicts an example for `Ways2Stand.txt`. The example for `BasicTransitions.txt` is shown on the right. Here, each line represents one solution sequence. The numbers refer to the index of the corresponding motor primitive. The execution of a sequence propagates from left to right. The arrows in can be translated with “bring the robot from the initial position to ”.

The overall full search algorithm is listed in Algorithm 4 on the next page below.

In order to find most advantageous members for a vocabulary, it would be reasonable to stepwise trade insignificant components for new ones and thus gradually improve the set.

Algorithm 4 Full search algorithm

```
    until (all sequences executed) do {
      dog->SensePosition();
      dog->GetCurrentMotorFreq();
      bool ready=((dog->GetPosition()==Stand)
          && (all motor freq==0));
      if ((all motor freq==0)&&(execution started)
          && (dog->GetPosition())!= Stand))
      {
        Log sequence in file BasicTransitions.txt;
        return -1;
      }
      if ((!ready)&&(execution not finished))
      {
        Execute next action;
        return 0;
      }
      else
        if (ready)       // StandingUp accomplished
        {
          Log sequence in file Ways2Stand.txt;
          return 1;
        }
        else            // ActionSequence failed
        {
          return -1;
        }
        if (return value != 0)
          calculate new sequence;
    }
```

The significance or importance of a primitive can be seen in terms of its contribution to the behavioural diversity of the whole vocabulary. If the behaviours are represented as sequence (for instance as the result of our full search algorithm), this can easily be measured by counting the frequency of occurrence. If the diverse behaviours are presented in a graph such as Figure 3.6 on page 49, one has to multiply the number of incoming and outgoing connections. The more sequences depend on a specific action, the more justified is its nomination for the final set.

We repeat this algorithm for all vocabularies and, inspired by the qualitative experiments in section 3.4, also for two alternative ground angles. As in the real world experiments, we concentrated on the four extreme head positions straight downhill, uphill, left or right with inclinations of 22.5° and 30°. The result are presented in the next section.

6.3. Results

In this section, we analyse sequences with two or three primitives derived from our full search algorrithm. We review the Behavioural-Diversity-Index, the average length, the transfer capabilities for all combinations of vocabularies and morphologies in flat terrain and the robustness on slopes with 22,5° and 30°. Moreover, we evaluate the significance of each primitive.

Though we tried to reproduce the physical robot dog as realistically as possible, there are several sequences that only work in simulation (the same holds for the other way around), but these minor mismatches are insignificant for our conceptual research. In the following, 'O' will be the shortcut for the original shape of the head, 'V' for the vertical cylinder and 'R' for the round head.

The evaluation of distinct vocabularies according to the measure for behavioural diversity postulated above is listed below. Table 6.1 schedules the average length and the Behavioural-Diversity-Index. Average length '∞' tributes to the fact that the quadruped cannot stand up from all initial positions (here: *Left*).

Vocabulary	Average Length			Behavioural-Diversity-Index		
	O	V	R	O	V	R
1	∞	2.73	∞	7.33	8.81	6.74
2	2.85	2.85	2.83	16.66	16.47	12.73
3	2.93	2.8	2.9	5.59	6.64	2.27
4	2.95	2.95	2.94	8.23	8.62	7.62
5	2.84	2.84	2.79	15.45	16.49	12.19
6	2.94	2.88	2.91	7.82	9.63	3.26

Table 6.1.: Behavioural-Diversity-Index of vocabularies evaluated for standing up within two or three steps
Those vocabularies are considered the better, the lower the average length of their solutions. '∞' means that the agent cannot stand up from at least one initial position (here: Left). Those vocabularies are considered the better, the higher the BDI of their solutions. Looking at the BDI, vocabulary 5 performs best and vocabulary 3 poorest, irrespective of the shape of the head.

Looking at the BDI, we find that vocabulary 5 performs best, vocabulary 2 second best and vocabulary 3 poorest, irrespective of the shape of the head. Rank three to five depend on the particular morphology. Generally, a Flexibility-Index greater than zero results in an higher BDI. This can be explained easily, if the actual meaning of Flx in the sense of intermixing of motor primitives, is considered. By that means, a single primitive can, depending on its predecessors, occur in many different variants. In fact, the set of actually ten primitives is extended with a number of "mutants". A thus increased vocabulary naturally results in more diverse behaviours.

The much more interesting fact is that the BDI of original and vertical head is always very similar. This is especially fascinating, since very often the original and the round

	1			2			3			4			5			6		
	O	V	R	O	V	R	O	V	R	O	V	R	O	V	R	O	V	R
1	2	3	2	1	1	1	1	1	1	2	2	2	1	1	2	1	1	2
2	5	4	3	8	9	9	5	4	2	7	7	6	9	8	9	7	8	3
3	6	4	3	7	4	6	6	6	9	8	9	9	7	6	8	9	7	9
4	6	4	3	7	4	5	4	5	3	5	4	2	6	3	6	5	6	3
5	3	2	3	4	3	4	3	3	3	3	3	4	4	5	5	3	4	1
6	6	4	4	6	4	7	8	7	9	4	1	1	5	4	7	8	6	9
7	4	4	3	2	2	2	7	3	9	6	6	5	2	2	1	6	5	3
8	1	1	1	3	3	3	6	6	9	1	5	3	3	5	3	4	3	4
9	5	4	3	5	4	6	2	2	3	7	7	6	8	7	4	2	2	2

Table 6.2.: Significance of each motor primitive (first colum) within its vocabulary (first row) in dependency of the shape of the head (second row)
1 signifies the most and 9 the least important component for the behavioural diversity. For a vocabulary with maximum behavioural diversity, a researcher should select the most significant primitives from each set.

head have at the most part identical solution, whereas the vertical head has not. This instance automatically proves that our measure is indeed independent of the underlying behaviours.

Table 6.2 shows the significance of each primitive in dependency of the shape of the head. The numbers from 1 to 9 signify the order of priority where 1 signifies the most and 9 the least important component within each set. In most cases, a significance value of 9 means the respective component is not used at all. For a vocabulary with maximum behavioural diversity, a researcher should select the most significant primitives from each set.

Regarding the above mentioned transfer capabilities, we find that irrespective of the shape of the head, those vocabularies with Flexibility-Index greater than zero (vocabulary 3 to 6) have equal or more possibilities to traverse between the basic positions than the according vocabulary in group A. There are only two exceptions: vertical head, vocabulary 4, initial position back and original head, vocabulary 5, initial position head. This result can also be explained refering to the additional variants. Actually, it is not at all surprising that the degree of behavioural diversity complies with the transfer capability, since, in a way, the transfer capabilities will extend to solutions if they were given more time. In doing so, they also contribute to the behavioural diversity. The greatest difference hereby is between vocabulary 2 and 5, the least between 3 and 6. The detailed results can be looked up in Table 6.3 on page 81. Here, the numbers represent the amount of valid sequences of length two or three that lead from the initial position on the left to the initial position in the column. It is clear that we left out transitions where the initial and the goal posture are identical because we can just do nothing instead.

Comparing the different morphologies, we find many relations between their solutions that not at all arbitrary, but that are grounded in the choice of motor primitives as well as

in the shape. Here the stress lies on shape, since the weight and the centre of mass of each part of the body is identical in all versions.

Though only few strict rules can be identified, there are many examples where the entirety of solutions for one initial position are equal or a subgroup of the solutions of another initial position within the same or different head forms. The solutions for the round head for instance intersect to a large extent with the solutions of the original head. The vertical head's solutions for *Right* often hold for *Back*, too. Further, we find that the set of solutions for each initial positions of the round head intersects with the respective set for the original head. Regarding the original head, the solutions for the right hand side are strongly related to left hand side solutions of the vertical head.

Moreover, we find that the solutions for all initial positions and heads are most dissimilar for vocabulary 3 and 6. The highest similarity can be detected for vocabulary 2 and 5. Herein we even observe symmetrical tendencies. This means for the original head that all solutions for the *Left* are also valid for *Back*. The same holds for about half of the solutions for *Right* and *Head*. However, this relation is inverted for the vertical head which means that the solutions for *Back* are identical to those for *Right* instead of *Left*.

We find that single solutions are valid for all initial positions of one head ('position stable'), some for the same initial position, but of different heads ('form stable'). Most surprisingly, some solutions are even valid for all forms and all initial positions (form and position stable). These especially robust sequences can be divided into two groups. In the first group, MiniDog6M needs to carry out the whole sequence for all basic positions, whereas in the second group the robot can still get up only with the last part of the sequence.

Actually, all vocabularies provide at least one type stability: though all of them produce form stable sequences for the right hand side, only vocabulary 2, 5 and 6 produce form stability for the left hand side. This imbalance is grounded in the asymmetry of motor range. Depending on vocabulary and initial position, the amount of stable sequences varies considerably. For vocabulary 5, the amount is nearly the same for *Left* and *Right*. Though vocabulary 5 is most stable for *Left*, is only second rank for *Right*. Still it the best choice regarding morphological changes. The biggest difference is, as usual, produced by vocabulary 1, which is the most form stable vocabulary for *Right*, but unfortunately cannot provide any solution for *Left*.

Further, we find that it seems to be much harder to create position stable solutions than to create form stable ones. So only the vertical head supports position stability for all vocabularies. In contrast to that, only vocabulary 2 and 5 achieve position stable solutions for the original head. Once again, vocabulary 5 is our first and vocabulary 2 our second choice. The last rank is preserved for vocabulary 4. Regarding those results, it is not astonishing that only vocabulary 2 and 5 achieve both, form and position stable solutions. Again 5 is better than 2.

An interesting fact is that sequences, that are stable in only one way, involve motor primitives with Flexibility-Index greater or equal zero, but that primitives belonging to form and position stable sequences all have Flexibility-Index zero.

Considering the above mentioned compliance between the solutions of different heads, one would assume that the respective agents behave similarly in changing environment, for instance on inclines. To be concrete, we expect that form stable sequences on even ground, perform for all heads identically on inclines. This should above all be true, if, as it is in our case, the weight distribution and the centre of mass are identical.

Surprisingly, this assumption is not true. Tables 6.4 on page 82, 6.5 on page 82 and 6.6 on page 83 show the validity for learned behaviours on inclines of 22.5° and 30° ($\sin$ and $\cos$ are an abbreviations for $\sin\alpha$ respectively $\cos\alpha$). The columns are labelled with the number of the vocabulary, the ground angle and the normal vector $(x\,y\,z)$ on the ground. 'R' stands for *Right* and 'L' for *Left*. '=' means that all behaviours for flat terrain are also successful on the respective inclination. '+' means that new behaviours emerged. The opposite is the case for 'X', where there is no solution at all. The other shortcuts are composed of the symbol for the shape of the head and the symbol for the initial position whose behaviour is assumed.

As we can see, the original and the round head perform equal or better in most cases. The only exception is vocabulary 1, which accomplished less behavioural diversity on steep inclines (but equal to vertical head, *Right*). Hence we conclude that the round head performs better than the original and that all vocabularies are of equal quality except of vocabulary 1. Nevertheless it is not that easy for the vertical head. We can divide its robustness into three different categories. Vocabulary 2 to 5 switches the solutions for *Left* and *Right*. The same holds for the right hand side solutions of vocabulary 1 as well as for the left hand side solutions of vocabulary 6. The left hand side solutions of vocabulary 1 behave as if with the original head (two exceptions on inclines of 22.5°). For the right hand side, vocabulary 6 performs equal or better than the original head. Note that VL for vocabulary 1 to 5 is for the most part the same as OR.

As a conclusion, we can say that the original and the round solutions are really robust for applications in environment with slopes of various degrees and directions. Those for the vertical head underlie a strange morphological effect that causes MiniDog6M to switch left and right hand solutions. So far, we cannot explain this strange morphological effect except that it is somehow grounded in the shape.

Generally trial and error at random is not a good strategy for an agent behaving in natural environment. To get along in new situation structured search and learning is much more suitable. Therefore, the support of the learning progress is an important feature of a good basic vocabulary.

O		1	2	3	4	5
1	1	-	0	0	0	0
	2	0	-	0	0	0
	3	14	12	-	0	0
	4	18	1	0	-	0
2	1	1	-	0	0	0
	2	0	-	0	0	0
	3	14	0	-	0	0
	4	2	10	12	-	0
3	1	-	2	6	0	0
	2	1	-	7	0	0
	3	1	12	-	0	0
	4	0	13	2	-	0
4	1	-	0	4	1	0
	2	0	-	0	2	0
	3	14	12	-	0	0
	4	18	7	0	-	0
5	1	-	1	4	0	0
	2	0	-	0	2	0
	3	12	1	-	2	0
	4	0	17	10	-	0
6	1	-	3	9	0	0
	2	1	-	8	0	0
	3	1	12	-	0	0
	4	0	13	3	-	0

V		1	2	3	4	5
1	1	-	0	0	0	0
	2	0	-	0	0	0
	3	14	12	-	0	0
2	1	-	0	0	0	4
	2	0	-	0	0	0
	3	28	0	-	0	0
3	1	-	0	4	0	0
	2	0	-	2	0	0
	3	2	1	-	0	0
4	1	-	0	0	2	0
	2	0	-	3	1	0
	3	2	2	-	0	0
5	1	-	0	0	4	4
	2	0	-	3	0	0
	3	20	2	-	1	0
6	1	-	0	0	0	0
	2	0	-	6	0	0
	3	12	5	-	0	0

R		1	2	3	4	5
1	1	-	0	0	0	0
	2	0	-	0	0	0
2	1	-	1	0	0	0
	2	2	-	0	0	0
3	1	-	1	7	0	0
	2	0	-	7	0	0
4	1	-	0	6	0	0
	2	0	-	0	0	0
5	1	-	3	4	0	0
	2	0	-	0	1	0
6	1	-	1	10	0	0
	2	0	-	9	0	0

Table 6.3.: Transfer capabilities of underlying vocabularies.

The first column labels the subsequent section with the number of the vocabulary. Each section is to be read as follows: standing up from [row] can be transferred into standing up from [column] by [item] sequences. We left out transitions where the initial and the goal posture are identical, because we can just do nothing instead.

We find that irrespective of the shape of the head, those vocabularies with flexibility index greater than zero have equal or more possibilities to traverse between the basic positions than the according vocabulary without.

Vocabulary	α	$(0, \cos, \sin)$		$(0, -\cos, \sin)$		$(\cos, 0, \sin)$		$(0, -\cos, \sin)$	
		R	L	R	L	R	L	R	L
1	22.5°	=	=	VR	=	VR	=	VR	=
	30.0°	=	=	=	=	=	=	=	=
2	22.5°	=	=	=	=	=	=	=	=
	30.0°	=	=	=	=	=	=	=	=
3	22.5°	=	=	=	=	=	=	=	=
	30.0°	=	=	=	=	=	=	=	=
4	22.5°	=	=	=	=	=	=	=	=
	30.0°	=	=	=	=	=	=	=	=
5	22.5°	=	=	=	=	=	=	=	=
	30.0°	=	=	=	=	=	=	=	=
6	22.5°	=	=	=	=	=	=	=	=
	30.0°	=	=	=	=	=	=	=	=

Table 6.4.: Validity for learned behaviours on inclines: Original head
=: All behaviours for flat terrain are also successful on the respective inclination
+: New behaviours emerged
X: There is no solution at all.
R: *Right*
L: *Left*

Vocabulary	α	$(0, \cos, \sin)$		$(0, -\cos, \sin)$		$(\cos, 0, \sin)$		$(0, -\cos, \sin)$	
		R	L	R	L	R	L	R	L
1	22.5°	=	=	VR	=	VR	=	VR	=
	30.0°	+	=	=	=	=	=	=	=
2	22.5°	=	=	=	=	=	=	=	=
	30.0°	+	=	=	=	=	=	=	=
3	22.5°	=	=	=	=	=	=	=	=
	30.0°	+	=	=	=	=	=	=	=
4	22.5°	=	=	=	=	=	=	=	=
	30.0°	+	=	=	=	=	=	=	=
5	22.5°	=	=	=	=	=	=	=	=
	30.0°	+	=	=	=	=	=	=	=
6	22.5°	=	=	=	=	=	=	=	=
	30.0°	+	=	=	=	=	=	=	=

Table 6.5.: Validity for learned behaviours on inclines: Round head
=: All behaviours for flat terrain are also successful on the respective inclination
+: New behaviours emerged
X: There is no solution at all.
R: *Right*
L: *Left*

Vocabulary	α	$(0, \cos, \sin)$		$(0, -\cos, \sin)$		$(\cos, 0, \sin)$		$(0, -\cos, \sin)$	
		R	L	R	L	R	L	R	L
1	22.5°	VL	OL	=	OL	X	OL	VL	OL
	30.0°	VL	OL	VL	OL	VL	OL	VL	OL
2	22.5°	VL	VR	VL	VR	VL	VR	VL	VR
	30.0°	VL	VR	VL	VR	VL	VR	VL	VR
3	22.5°	VL	VR	VL	VR	VL	VR	VL	VR
	30.0°	VL	VR	VL	VR	VL	VR	VL	VR
4	22.5°	VL	VR	VL	VR	VL	VR	VL	VR
	30.0°	VL	VR	VL	VR	VL	VR	VL	VR
5	22.5°	VL	VR	VL	VR	VL	VR	VL	VR
	30.0°	VL	VR	VL	VR	VL	VR	VL	VR
6	22.5°	OR	VR	OR	VR	OR	VR	OR	VR
	30.0°	OR+	VR	OR+	VR	OR+	VR	OR+	VR

Table 6.6.: Validity for learned behaviours on inclines: Vertical head
=: All behaviours for flat terrain are also successful on the respective inclination
+: New behaviours emerged
X: There is no solution at all.
R: *Right*
L: *Left*

7. Effects on learning progress

In this chapter the design of the learning environment and the experimental setup is described. Finally the learning progress is evaluated.

7.1. The learning method

In this section, the concepts that build the basis of our learning framework are described.

7.1.1. Q-Learning

Learning is particularly difficult in robotics because sensing and acting in the physical world involves uncertainty due to incomplete and noisy sensory information and a dynamically changing environment. Here learning means acquisition of a task fulfilling sensory-motor control strategy through trial and error. In doing so, learning strategies disagrees with adaptive control by allowing failure during the process of learning. This behaviour resembles the way that humans and animals acquire new strategies in thinking and movement. Reinforcement Learning (RL) is a wide spread approach to solve a great variety of learning problems without relying on a teacher or supervisor. Based on early conditioning work in psychology, learning is engaged by interaction with the environment [Sutton 98, Neumann 05].

During the process of learning, the adaptive system undertakes some actions which affect its environment. Hereupon, it is reinforced by receiving a scalar evaluation of its actions. This reinforcement signal is generally known as 'reward'. The reinforcement learning strategy stresses outputs that maximise the received reward over time. To maximise the gained reward, those actions must be preferred that in the past led to the highest reward in the given situation. This act of taking advantage of gathered knowledge is called exploitation. Yet, there might be new actions that are unexplored, but lead to even higher reward. Therefore the trade-off between exploitation and exploration is one of the key aspects in RL.

At each time step t, the learning system receives the state s of the environment. Depending on that, an action a is performed, which transfers the system into the new state s'. This transition is reinforced by the reward R. Time is generally preceived as discrete.

A central idea of reinforcement learning, together with trial and error search and delayed reward, is the estimation of how good it is for the agent to be in a given state or to take a certain action in a given state. This estimation is based either on a value functions V or a Q-Function Q which belong to the class of temporal difference learning (TD). To make long-term predictions about the dynamical system, $V(s)$ depends on the current state and $Q(s, a)$ on the current action-state pair. An action selection mechanism called 'policy' chooses the highest rated action for each state.

In this context, Q-Learning is one of the most popular type of Reinforcement Learning. Its aim is to find a satisfactory Action-Value-Function which maximises the future discounted reward if the agent chooses the action a in state s and then resumes policy π. This can be expressed in equation 7.1.

$$Q(s, a) = E[R(s, a, s') + \gamma * Q(s', a')] \tag{7.1}$$

where action a' was chosen according to the policy π and $R(s, a, s')$ is the reward gathered during the last step. $E(x)$ designates the expectation of x. Needless to say, that an action which was selected in a distinct state in the past and led to maximum reward are preferred whenever this state reoccurs.

Therefore, the Q-value $Q(s_{t,}a_t)$ at time index t is a matrix with one value for each distinguishable state and each action initialised with zero and updated during the learning process by

$$\begin{aligned} Q(s_{t-1}, a_{t-1}) &= \alpha \cdot err(t) \\ err(t) &= R(t) + \gamma \cdot \max\{Q(s_t, a)|a \in A\} - Q(s_{t-1}, a_{t-1}) \end{aligned} \tag{7.2}$$

with learning rate α and discount factor γ. A designates the batch of actions that are available in the current state.

Often the Q-Function is represented as a table, but since the motor angles in our model are implemented as real values, a table is not feasible.

7.1.2. Linear function approximation

To solve the problem of continuous state space mathematically, a common tool to estimate functions of all kinds is the so-called linear approximation. A linear model for a function $f(x)$ consists of a set of basis functions which are combined in a weighted sum as follows:

$$f(X) = w_i \cdot h_i(X) \;\; i = 0 \ldots m, \; X = (x_1, \ldots, x_m) \tag{7.3}$$

Whereas all parameters of the basis functions h_i are fixed (otherwise it would be nonlinear), the weights w_i are adapted so that the resulting sum resembles the original function $f(X)$ as much as possible. For theoretical concerns, homogenous function sets are of special interest. In this context, homogenous means of the same type, e.g. polynomial, exponential or sinusoidal. The latter are of particular interest for Fourier analysis.

7.1.3. Radial Basis Function network

In order to realise the linear approximation of the Q-function and because of its excellent generalisation ability, we decided on a Radial Basis Function network (RBF-NN) [Orr 96], which is described in the following.

In general, an artificial neuron (see Figure 7.1) consists of three parts, namely input, computation and output. The computational part can be decomposed into combination function which combines the inputs, the activation function which calculates the resulting activity, and the output function which delivers the corresponding output. All these elements must be defined for each neuron in the network.

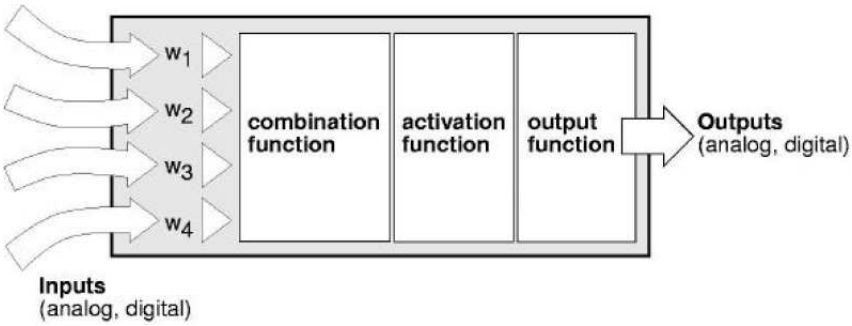


Figure 7.1.: Artificial neuron
Each artificial neuron has three parts: input, computation and output. The computational part can be decomposed into combination function, activation function and output function.

RBF-NNs are three layer networks with a radial activation functions in the hidden layer which can either be self-organised or fixed. Herein the most popular set consists of Gaussian functions with mean c and standard deviation r which in case of scalar input can be computed as follows

$$h_i(x) = e^{\frac{-(x-c)^2}{r^2}} \tag{7.4}$$

This approach follows the idea of bounded input pursued by bounded output. The resulting activation for a one respectively two dimensional input can be visualised as shown in Figure 7.2 on the next page. Each neuron has its own activation function which is only unequal zero for a small part of the input space. This area is limited by means of standard deviation. The resultant regions of limited size are called local receptive fields and

allow localised learning within the boundaries of such a region. This dynamic allocation of resources significantly reduces the computational effort for RBFs and, for that reason, makes them suitable for online function approximation. This is also the reason for a much better performance than in ordinary feed forward NN. Moreover the same property supports the switching between behaviours (as mentioned in section 6.1) by adding an offset to all input values and thus shifting the activation to another part of the network, where a diverse behaviour might be realised.

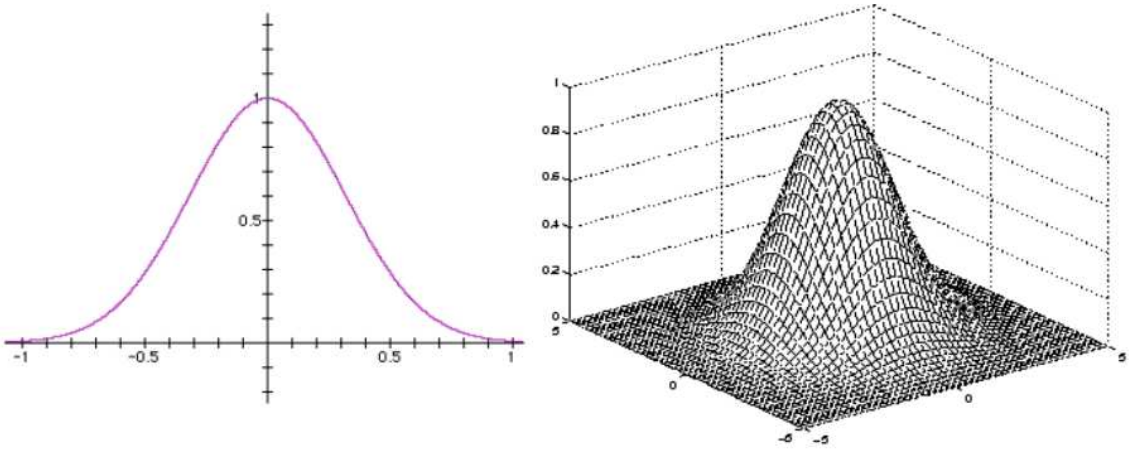


Figure 7.2.: Two (left) respectively three (right) dimensional Gauss function with mean 0.0 and standard deviation 1.0

7.1.4. A fertile interplay of all three concepts

Combining the issues mentioned above, we linearly approximate the Q-Function by means of a RBF-NN. In this context, the NN can be implemented as shown in Figure 7.3 on the facing page.

The layers are fully connected and the number of neurons in the hidden layer is fixed. The input vector, which is almost directly fed into the hidden layer neurons and the weights between the hidden and the output layer are represented by w_i from equation 7.3 on page 86. The base functions $h_i(X)$ stand for the activation of the hidden knot, the output is the estimated Q-Value and is made up of the weighted sum of all activations.

Actually not all inner neurons are active, since only those neurons having an activation unequal zero whose RBF-centers lie within the range of two sigma around the current input vector. In the special case of one dimensional input, the activation of each neuron can be directly read off from the Gaussian distribution and can be pictured in Figure 7.4 on page 90.

Another reason for choosing RBF-NN, is the fact that, since their activation is scaled with distance from the centre, linear function approximation using localised receptive fields with an activation factor between [0.0; 1.0] generalise better than discrete states. Note that the discrete state representation can be considered a special case with only one active

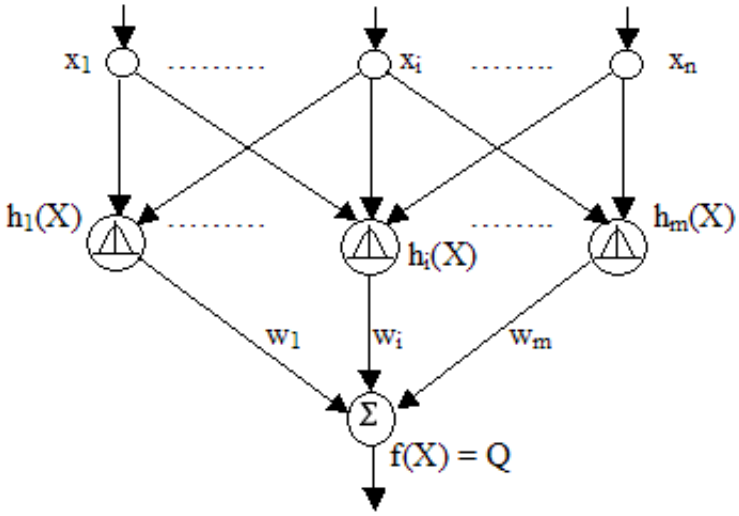


Figure 7.3.: RBF-NN as linear approximator
The layers are fully connected. X is the input vector. The weights between the hidden and the output layer are represented by w_i from equation 7.3. The basis functions $h_i(X)$ stand for the activation of the hidden neuron. The output being the weighted sum of all activations is the estimated Q-Value.

feature (thus the activation is either 1 or 0).

Since the current input has only local influence in RBF-networks, only the weights of neighbouring features must be adapted in each learning step. Using fixed centres and sigmas, we just have to learn the linear scale factors of the RBF-Functions. We chose those centres according to the minimal, maximal and mid position of each motor, since they are the only relevant goal positions for standing up.

We decided to use this approach, since it merges the advantages of both, RBF-NN and Reinforcement Learning. Solutions that purely rely on the neural network part, have on one hand the great generalisation ability combined with optimal computational efficiency, but are highly non-linear and very difficult to analyse. RL, on the other hand, is well formulated in mathematical terms and, for the external observer, it is quite easy to understand what is going on in the internal process. Further, the idea of representing complex function as a linear combination of much simpler functions is a well established theory in maths and physics.

In order to get quick results, we have the Reinforcement Learingn Toolbox, which will be described in the next section, take care of the learning process.

7.2. Reinforcement Learning Toolbox

In this section, a framework for RL, namely the Reinforcement Learning Toolbox (RLT) [Neumann 05] will be described. We decided to use the RLT, since, on one hand, we can easily extend our system for hierarchy and other future assignments without having

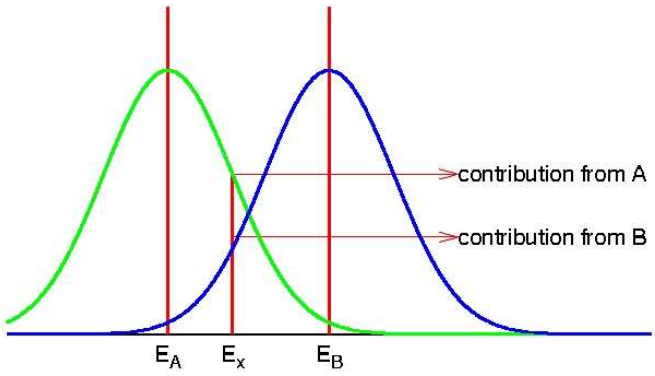


Figure 7.4.: Activation of two neighbouring RBF-neurons A and B in two dimensional case
E_A, E_B: mean of A respectively B
E_X: current input
Activation of neurons A and B, which are next to the current input, can be directly read off from the respective Gaussian distribution.

to redesign everything and since, on the other hand, it is available to anyone who plans similar studies. So some decisions were met to fit the concept of the RLT and not because we think it was the best possible choice.

The RLT is a C++ framework for a variety of reinforcement learning algorithms. Being developed by Gerhard and Stephan Neumann from TU Graz, this library is for the most part designed for researchers with the intent of letting the user concentrate on the learning problem itself and not on the implementation of the learning functions, policies etc.

At present the following learning functions are covered: TD-lambda Q-Learning learning, TD-Lambda V-Learning (TD learning also with continuous Time Residuals), Actor critic learning, Advantage Learning, model based reinforcement learning (prioritised sweeping, value iteration), policy search algorithm (PEGASUS and CONJPOMDP) and VAPS.

Additional basic features are tools that support error recognition, hierarchical reinforcement learning and logging of Q-function, policy or whole episodes. A semi Marcov Decision Process (MDP) learning environment is alleged.

For reasons of reuse and complexity, the RLT Q-Learning treats each line of the action-value matrix individually which means learning a separate value function for each action. The value function V(s) of state *s* is defined as future discounted reward if the agent starts in state s and follows the policy π. In regard of the prefix “future”, the reward can only be predicted and thus the expectation $E[R]$ is used. Herein, the discount factor γ is used to stress reward that are expected in the near future.

$$V(s) = E[\sum_{k=0}^{m} \gamma^k R(t+k)] \tag{7.5}$$

This kind of value estimation, where the entire trajectory is considered beginning in state s, is often referred to as Monte-Carlo-Method.

Using the RLT, it is possible to create one or more learning objects with diverse learning algorithms, reward functions or even different state discretisations, each of which can learn simultaneously from the same training trial. The user has to provide the RLT with an environment model, a set of actions and a reward function. Alternatively, the user can provide his own pre-programmed controller and then try to improve it with reinforcement learning. One can choose to learn from single steps or whole episodes. The current version is 2.0.

Furthermore, the toolbox offers the possibility to use a linearly approximating RBF-NN or to introduce external neural networks (NN) from the Torch library. Torch is a BSD licensed machine learning library containing all sorts of artificial neural networks (including convolution network and time-delay neural networks), supporting vector machines for regression and classification, Gaussian mixture models, hidden Markov models, K means, K nearest neighbours, Parzen windows, bagging and adaboost. For convenience of the user Torch is included in the reinforcement learning toolbox, but can also be downloaded separately, the current version being 3.1.

In the next section, the experimental setup will be described.

7.3. Experimental setup

As already described, we utilise a linear Q-approximating RBF-NN. This section presents the parameter setting, the model and the reward function of our learning environments as well as the interplay with ODE.

The class `DogModel` encapsulates the model of MiniDog6M and provides the interface to the RLT. Motor angles are continuous variables and the position of the head is a discrete variable. Further, it provides the reward function as well as functions to reset and update the environment model. Thus it must be inherited from `CEnvironmentModel` and `CRewardFunction`.

Each of the previously nominated motor primitives supplies one action of the type `CPrimitiveAction` to the learning envirenment. Every action is available in each step of our learning environment.

As already mentioned, the learning progress itself is invisible to the user. In our RBF-NN, the neurons of the hidden layer are called feature. Their centres, which are the

means of the Gassian activation function, are set on minimum, zero and maximum of each motor's angular range. The standard deviation is calculated automatically so that 2 centres have a distance of twice the standard deviation. To calculate the activation of each feature, we use the RLT's `CSingleStateRBFFeatureCalculator`, which means that each feature has its own set of parameters. Using linear interpolation, only those two features are active, which are the nearest to the current input vector. Thus we get $2N$ active features, where N is the number of input dimensions. Their activation Ψ is calculated as follows:

$$\Psi_i(s) = \frac{a_i(s)}{\sum_j a_j(s)} \tag{7.6}$$

where i and j are indices of a feature of continuous state variables and a_i denotes the Gaussian activation function with mean μ and standard deviation σ:

$$a_i(s) = e^{-\frac{1}{2}\sum_j \left(\frac{s_{i,j}-\mu_{i,j}}{\sigma_{i,j}}\right)^2} \tag{7.7}$$

As can easily be seen the activation is normalised with the sum taken over the remaining features and thus lies between 0.0 and 1.0.

For discrete state variables, the activation is very simple since only one feature is active. The remaining features bear activation 0.0. As a result, the number of features for continuous state variables multiply with the number of discrete features. We imagine this as having one set of continuous features per discrete value. So the activation must only calculated for those continuous state variables that belong to the currently active discrete feature.

To finally select an action on the basis of our Q-Function, an Epsilon-Greedy-Policy with $\epsilon = 0.3$ is pursued. This strategy selects the Greedy-Action which means the action with the highest Q-Value, with the probability of 0.7 and chooses a random action with a probability of 0.3. The learning rate $\alpha$is set to 0.4, discount factor γ has a default value of 0.95. Replacing eligibility traces with $\lambda = 0.9$ are used. λ is used to diminish the responsability of past actions for the currently given reward respectively the current TD-error. This problem is often refered to as credit assignment problem. A value of 0.9 means that past action have a strong influence on the current state. The attribute "replacing" means that the trace will be reseted as soon as a nongreedy action is selected. On one hand this might slow down the the learning progres, since a learner has apotentially short sighted knowledge base. On the other hand, it makes the learning progress make stable, since a learner, who gernerally assumes that the greedy action was taken, might be confused be a resulting high TD-error.

The Q-function that arises from this combination of the Q-learning base class with replacing eligibility traces and direct gradient calculation can be summarised as follows:

$$Q(s,a) = \sum_i \Phi_i(s) \cdot w_{i,a} \tag{7.8}$$

where

$$\Phi_i(s) = \frac{Q(s,a)}{dw_{i,a}} \tag{7.9}$$

represents the gradient of the linear approximator. Note that for all other actions

$$\frac{Q(s,a_1)}{dw_{i,a_2}} = 0, \text{ with } a_1,\ a_2 \in A\backslash\{a\} \tag{7.10}$$

Using a linear approximator, the Q-Function is not updated directly. Instead, we have to update the weight of each feature separately. Regarding the concept of receptive fields in RBF networks, computational effort is reduced and the update is accelerated, since we only have to update the active features in each step. In our case these are only two. The resulting update function is:

$$\Delta w_{i,a} = \alpha \cdot (r_t + \gamma \cdot \max_{a'} Q(s_t+1, a') - Q(s_t, a_t)) \cdot e_t(w) \tag{7.11}$$

with

$$e_t(w) = \lambda \cdot \gamma \cdot e_{t-1}(w) + \frac{dQ(s_{t-1}, a_{t-1})}{dw_{i,a}} \tag{7.12}$$

Here i refers to a continuous state variable and $w_{i,a}$ refers to the functions approximator's weight vector assigning a scaling factor to each feature i for action a. Δw is the weight update. e denotes the eligibility trace which, in case of function approximation, is not used to trace the recent state history, but to directly trace the approximator's weights instead. Thus it is designated as $e(s)$ instead of $e(w)$.

If a robot autonomously explores its environment, it is convenient to accomplish a maximum of tasks before returning to the charging station. Therefore and for reasons of efficiency, we decided on a time limit. The learning process is segmented into episodes of six steps. As mentioned above, each steps consists of `NumTicks` ODE world steps. An episode is claimed to have failed, when the simulated dog robot neither stood nor ran within the given six steps. Here "stand" and "run" are defined as upright posture, gait *Still* with legs in mid position or gait *Running*.

For running (as just defined) a reward of 150 is assigned, for standing the reward is 100. To speed up learning by encouraging exploration, we also dispense negative reward, giving a penalty of -1, if the agent did not succeed within six steps. In all other steps the reinforcement signal is zero. The resultant reward function is

$$R(n) = \begin{cases} 150.0 & \text{running in } n\text{th step} \\ 100.0 & \text{standing in } n\text{th step} \\ -1.0 & \text{lying down and } n = 5 \\ 0.0 & \text{otherwise} \end{cases} \tag{7.13}$$

with $n = 0 \ldots 5$ being the step counter in the current episode.

The initial position for each learning episode is determined by applying a random force to knock over the running dog. The probability distribution can be seen in Table 7.1 below.

	1	2	3	4	5
Original	0.39	0.39	0.15	0.07	<0.01
Vertical	0.44	0.44	0.12	<0.01	<0.01
Round	0.48	0.52	<0.01	0.0	<0.01

Table 7.1.: Probability distribution for initial position (first row) in dependency of the shape of the head (first column)

The overall learning algorithm is listed in Algorithm 5 below.

Algorithm 5 Learning algorithm

```
if(steps==0)
  model->Reset the environment model;
model->refresh inputs for RBF-NN
if(steps==0)
  agent->start new learning episode;
agent->perform step through NN;
model->Read the network outputs and set them for ODE
model->Get reward
update weights in NN;
if(steps < 5)
  steps++;
else
{
  model->new MiniDog6M;
  step = 0 ;
}
Throw over dog
```

Using the RLT, the only thing the user has to do is to provide an environment model, a set of actions and a reward function. Moreover, the user has to do everything that is external to the toolbox which means everything that is not directly concerned with Q-function, RBF-NN and action selection. Access and control of ODE, for instance throwing the dog over, carrying out the selected action and collision detection, fall in this category. The other steps are carried out automatically by the RLT through calling the respective functions. Most of these methods are internal which means the user can either access them indirectly by setting some decisive parameters, such as learning rate, discount factor etc or not at all. Some methods need to be overwritten by the user and made available in the model class. These methods are:

- `virtual double getReward(CStateCollection *oldState, CAction *action, CStateCollection *newState)`

- `virtual void getState(CState *state)`
- `virtual void doNextState(CPrimitiveAction *action)`
- `virtual void doResetModel()`

First, `getState` updates the model state by collecting the respective information from the ODE environment. On that basis, `getReward` returns the reward gathered throughout the last step. `doNextState` passes the selected action on to ODE. Further, it sets the reset as well as the failed flag to determine whether an episode has successfully ended, has failed or has not ended at all. `DoResetModel`, which prepares the learning model for a new episode, is triggered by the reset flag.

7.4. Results

In this section, we present the learning progress of the first 200 episodes.

The success rate denotes the probability that the dog is able to resume its way within six steps. So failure here does not mean that the dog was unable to stand up at all, but only that it would need more than the given six steps. In the following, we disregard initial positions that occur with a probability less than 0.01.

For our environment model, we tried several configuration of state variables. First, we took all motors as continuous and the position of the head as discrete variable. Seeing, that this results in an enormous state space, we substituted the front left and hind right motor with a single discrete variable that indicates whether front and hind or left and right hand legs move in parallel. Aiming at a minimalistic design, we soon found out that the agent even succeeded if it only knows the angle of the spinal motors and the position of the head. We stick to the latter configuration, since it leads to the fastest success.

The evaluation of learning progress of one shape of the head in dependency of the respective vocabulary is shown in Figure 7.5 on the next page to on page 98 below. The evaluation of learning progress of the one vocabulary in dependency of the shape of the head can be viewed in Figure 7.8 on page 99.

Comparing the results gained from learning progress and behavioural diversity, we find that a Behavioural-Diversity-Index greater than 10 guarantees success in not more than 100 episodes, whereas in many cases vocabularies with a lower BDI do not even reach 100% success at all.

Having a closer look at the final Q-Function, we still find a lot of values equal to 0. The reason for this is obviously that the morphology prevents certain postures or that possible postures are not reachable with the underlying vocabulary. The latter is also the reason why MiniDog6M with the original head cannot stand up from the left with vocabulary 1 and consequently cannot reach a success rate higher than 0.61 as seen in Figure 7.5 on the following page.

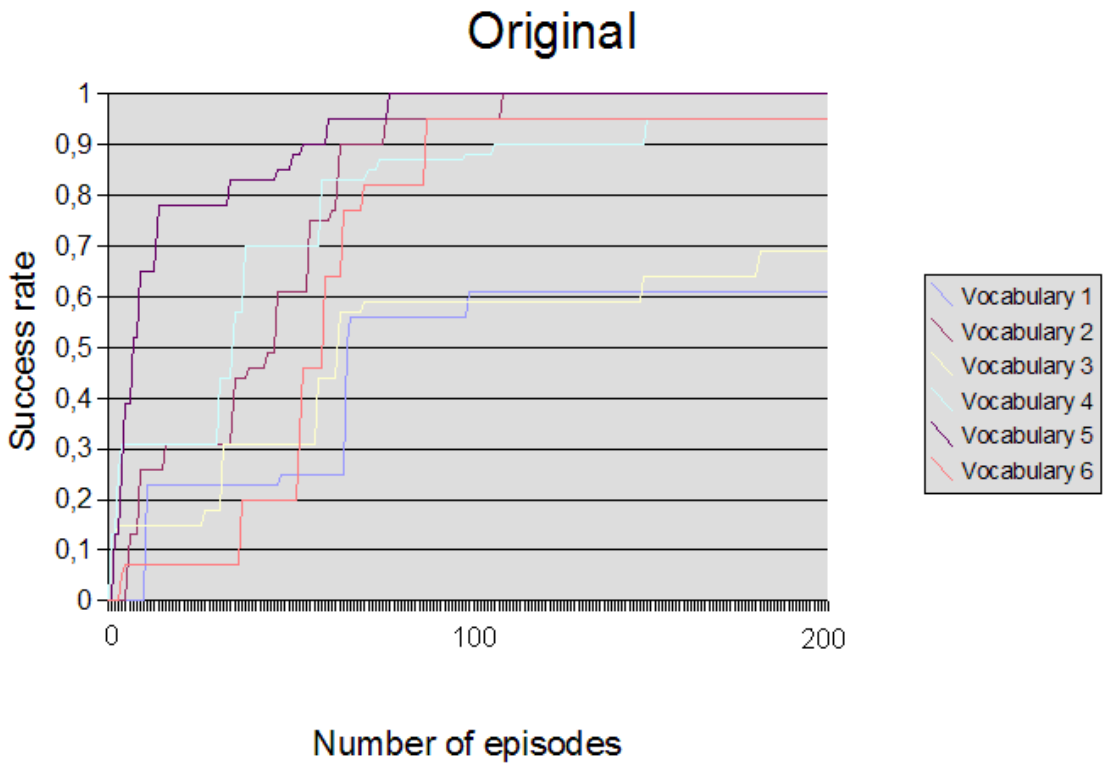


Figure 7.5.: Success rate of original head over 200 episodes BDI greater than 10 guarantees success rate = 1.0 in not more than 100 episodes otherwise 1 may not be reached at all. MiniDog6M cannot stand up from the left with vocabulary 1 and thus cannot reach a success rate higher than 0.61. Flexibility-Index greater than zero accelerates the learning process.

However, this is not the reason why the original head equipped with vocabulary 3 respectively the round head equipped with vocabulary 4 do not exceed 0.7. Moreover it does not explain why the round as well as the vertical head do not learn at all with vocabulary 6. We already suggested that their low BDI is a good indication. For most cases this is not only an indication, but even the very reason, since finding a proper sequence for these configurations equals the notorious search for the proverbial needle in a haystack. In cases of moderate BDI, e.g. 8 or 9, the explanation should be found in an disadvantages search strategy through the state space. If the success rate lies between 0.9 and 1.0, the reason is most of our runs already reached 1.0, but not all of them. Consequently the mean success rate is still less than 1.0. So far, it seems that that a high BDI is necessary, but that it is not a sufficient morphological explanation for the observed performance differences.

Interestingly, a Flexibility-Index greater than zero has different consequences for different morphologies. Actually, it was expected that this type of flexibilty hinders the learning process for all heads. This is because the goal posture is no longer unambigous and thus the learner might be confused as the transitions between current state and new state are no longer injective.

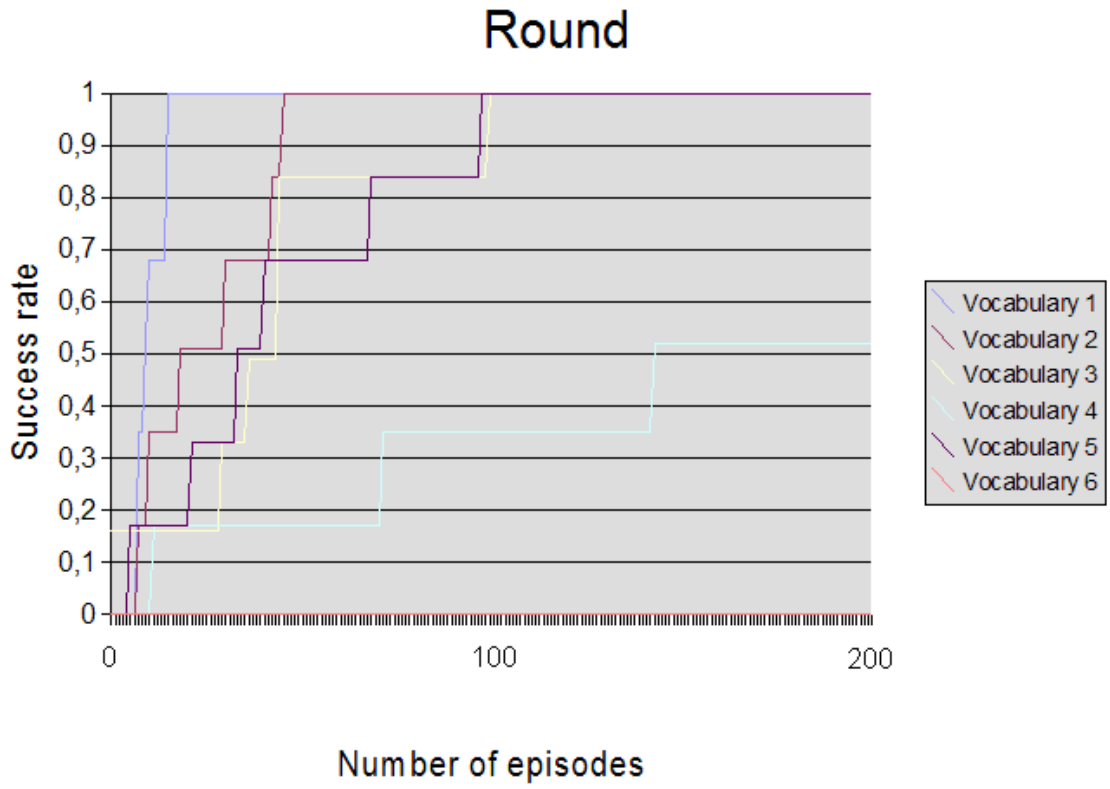


Figure 7.6.: Success rate of round head over 200 episodes BDI greater than 10 guarantees success rate = 1.0 in not more than 100 episodes otherwise 1 may not be reached at all. MiniDog6M cannot stand up from the left with vocabulary 1 and thus cannot reach a success rate higher than 0.61. Flexibility-Index greater than zero accelerates the learning process.

Considering the probability distribution of the initial states, it becomes obvious that the learning progress is the faster the less initial positions are supported, but only for vocabularies of type A. It can easily be seen that in case of Flexibility-Index of zero the round head learns the fastest and the original head the slowest. For Flexibility-Index significantely greater than zero, it is just the other way around. The fastest progress for vertical and original head can be observed with vocabulary 5, while for the round head MiniDog6M performs best with vocabulary 1. To put it simple, $Flx > 0$ accelerates the learning process for the original as well as for the vertical head, but in contrast to that decelerates it for the round head.

This insight is especially astonishing, since the solutions for round are for the most part subsets of the solutions for the original head. Thus, it should be granted that the round head learn slower, simply as its solutions are harder to find, and that further, their learning progress should be quite similar, irrespective of the vocabulary. Nevertheless, it seems that this special type of flexibility outweighs the additional effort resulting from more initial positions. Anyway, this effect comes effortlessly if it is considered in the design phase already.

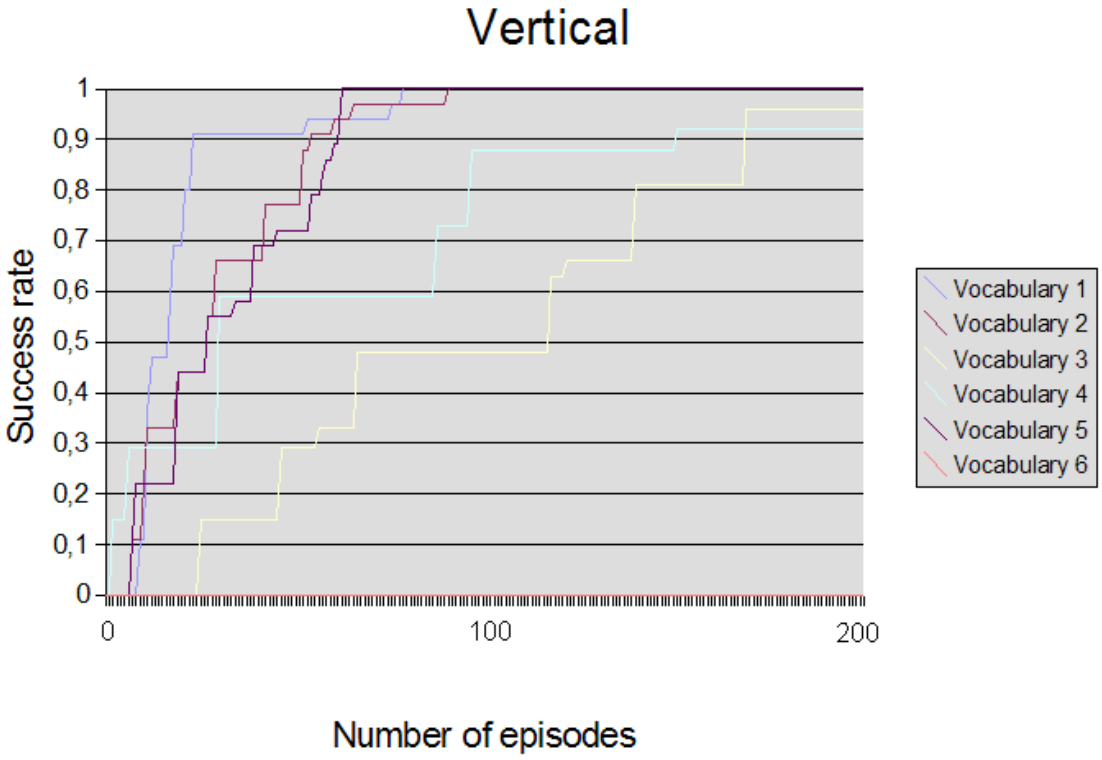


Figure 7.7.: Success rate of vertical head over 200 episodes BDI greater than 10 guarantees success rate = 1.0 in not more than 100 episodes otherwise 1 may not be reached at all. Flexibility-Index greater than zero accelerates the learning process.

As the search strategy for nongreedy actions is unfortunately not completely randomised, the learning progress is additionally influenced by the location of the single solutions within the search space. This problem is one reason why we cannot derive quantitative results from these experiments.

In the next chapter, the gathered results are presented and an outlook on future assignments will be given.

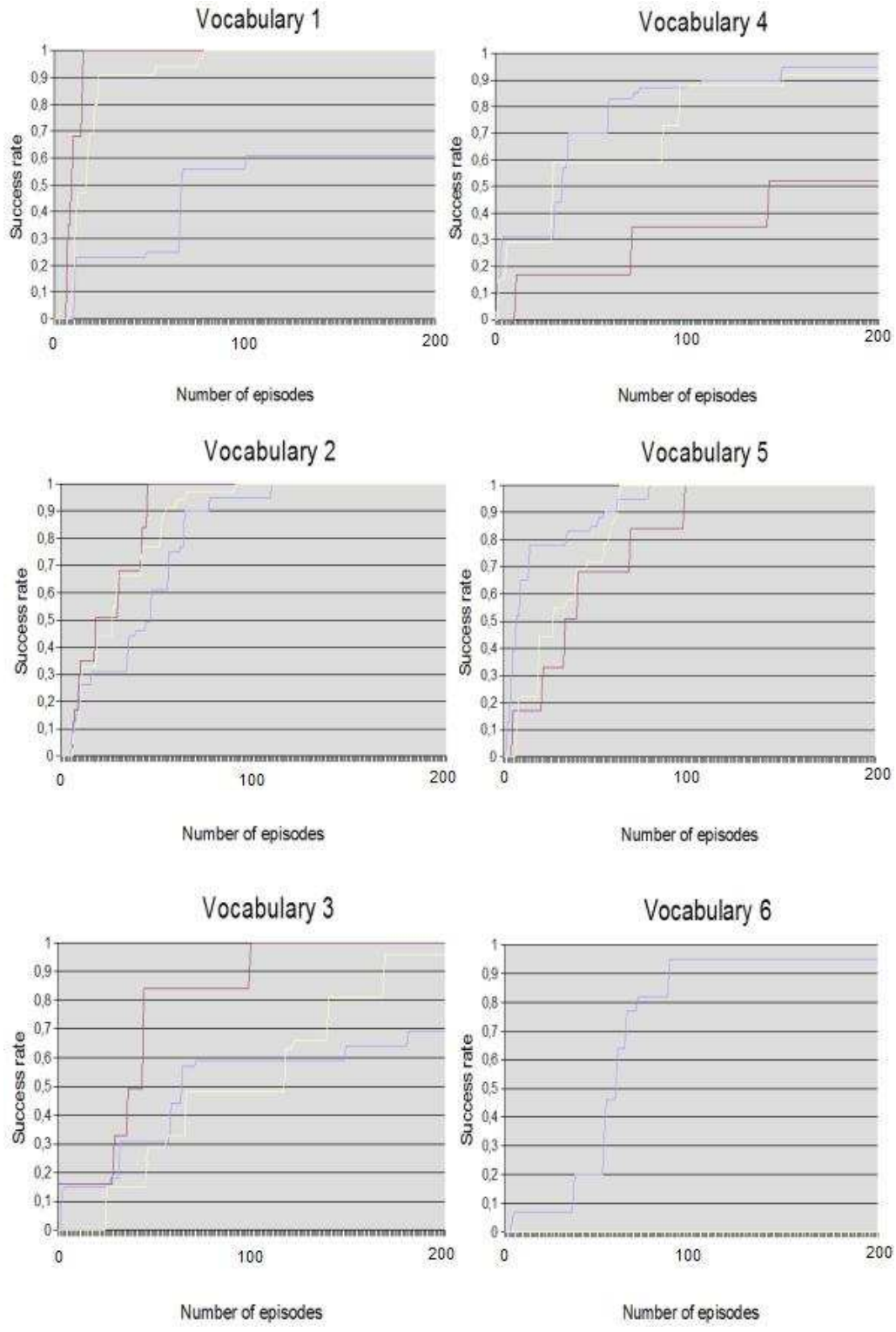


Figure 7.8.: Success rate of all vocabularies over 200 episodes
Red line: Round head
Yellow line: Vertical head
Blue line: Original head
In case of Flexibility-Index = 0 the round head learns the fastest and the original head the slowest. For Flexibility-Index > 0 it is just the other way around.

8. Conclusion and future work

In this chapter, the results gained from this thesis will be described. Furthermore, future assignments that directly hook up on our framework as well as entitlements to the MiniDog6M project are presented.

8.1. Results and implications

The goal of this thesis was to explore how the morphological properties contribute to generating these discrete entities in the continuous sensory space. In doing so, we investigated how different vocabularies affect behavioural diversity, robustness and learning process. Robustness in this context means that the behaviours are tolerant against changes in morphology, environment and posture. For that purpose, we examined six different vocabularies, three different forms of the head and nine different ground configurations with slopes of $0°$, $22.5°$ and $30°$. Further, we provided abstract, task and platform independent measures to categorise and evaluate single motor primitives, entire vocabularies and behavioural diversity.

8.1.1. Cheap design

We introduced the project of passive quadruped running to show that our design is in deed "cheap" and that symmetrical gaits are to be favoured for energy efficient locomotion. As a result, all of MiniDog6M's gaits are symmetrical along one axis or the other. MiniDog6M realises the principle of cheap design by means of using low friction feet and springs for locomotion. With its simple sinusoidal controller and the lack of sensory feedback, MiniDog6M definitely undercuts the minimum requirement of Scout 2.

Comparing videos of Kenken and MiniDog6M, one can see that Kenken hops much higher. This advantage of height simplifies moving on uneven ground, but comes at cost of more complex control. Talking about cheap design, MiniDog6M's assembly and controller concept are more fitting, if the roughness of the terrain is reasonable. Though not MiniDog6M, Puppy will eventually match equal jumping behaviour if its additional knee actuator will be introduced.

After all, we stabilised locomotion and increased manoeuvrability by attaching additional weights to the feet instead of improving control. This leads to an ecologically well balanced design. This kind of unburdening the controller by bringing forward a proper design, was introduced as morphological computation. The MiniDog6M project realises this concept by means of its springy locomotion that manages to create a continuous movement out of coarse-grained control.

8.1.2. Behavioural diversity

Despite the restriction to a distinct vocabulary, diverse activities can come up by means of sequential combination of single primitives. The red trajectories in Figure 3.6 on page 49 only emerge because of dynamical interactions. Furthermore, the representation in Figure 3.6 on page 49 can be compared with Figure 2.14 on page 24 in the sense that only the critical points are decisive for success. The precise postures in between, which are substituted with arrows, may vary a little.

Moreover, it was found that more behaviours come forward, if a special kind of flexibility is introduced, which arises from the introduction of special "don't care" symbols. Discovering the suitability of a multitude of complex sequences for different morphologies, tasks and environments, we suggest to introduce a higher level of hierarchy in order that task-oriented implementations can make use of all those different solutions. It would then be the charge of such a hierarchical higher level to choose the best fitting variant (according to one or more nonfunctional criteria) and to switch to an alternative behaviour if the best choice does not work out [1].

Regarding the transfer capabilities, we find that irrespective of the shape of the head, those vocabularies with Flexibility-Index greater than zero (vocabulary 3 to 6) have equal or more possibilities to traverse between the basic positions than the corresponding vocabulary with $Flx = 0$.

8.1.3. Impact on learning

Applying learning, we find that a Behavioural-Diversity-Index greater than 10 guarantees a success rate of 1.0 in not more than 100 episodes, whereas vocabularies with a BDI less than in many cases do not even reach 100% success at all. Finding a proper sequences for configurations with a low BDI equals the notorious search for the proverbial needle in a haystack.

Among other things, it was interesting to see that a Flexibility-Index greater than zero has different consequences for different morphologies. It accelerates the learning process for the original as well as for the vertical head, but in contrast to that decelerates it for the round head. In doing so, it somehow outweighs the additional effort resulting from more initial positions.

[1] This can easily be achieved in RBF networks by adding a parametric bias, which means an offset to all inputs, and thus shift the computation from one receptive field to another. Other models of hierarchical recurrent neural network can be found in publications of Jun Tani and Rainer W. Paine [Tani 02, Paine 04].

8.1.4. Robustness of behaviours

Introducing inclines, we showed the feasibility of the gained knowledge in changing environment without adaptation. This means no longer learning and still being able to succeed in new situations. The only thing a robot designer has to do, is to care for enough behavioural diversity. For this innovative robustness the transfer capabilities which we mentioned above are of particular importance. It was also shown that particular morphologies are to be preferred for operating in unpredictable environment, since the resulting behaviours are more robust. This ability is practically independent of the underlying vocabulary.

Talking about robustness we can say that the solutions for the original and the round head are also robust for applications in environment with slopes of various degrees and directions. Those for the vertical head underlie a strange morphological effect that causes MiniDog6M to switch left and right hand solutions. This is especially astonishing, since the weight as well as the location of the centre of mass or identical in all versions. Considering further the compliance between the solutions of different heads, one would assume that the respective agents behave similarly in changing environment for instance on inclines. Surprisingly, this assumption is not true. So far, we cannot explain this strange morphological effect, except that it is somehow grounded in the shape.

Moreover, it was demonstrated that single solutions are valid for all initial positions of one head ("position stable"), some for the same initial position, but of different heads ("form stable"). Most surprisingly, some solutions are even valid for all forms and all initial positions (form- and position stable). Further we find that it seems to be much harder to create position stable solutions than to create form stable ones. So only the vertical head supports position stability for all vocabularies. An interesting fact is that sequences that are stable in only one way involve motor primitives with Flexibility-Index greater or equal zero, but that primitives belonging to form- and position stable sequences all have Flexibility-Index zero.

It was further shown, that the behaviours that arise from our vocabularies, are also robist against perturbations of posture. Hence, it was enough to provide our learning model with not more than the agent's basic position and the angles of the two spinal motors.

8.1.5. Overall results

While it is obvious that the introduction of discrete actions alone reduces the complexity of a learning task by avoiding online trajectory planning, it was also testified that learning and control processes are closely related to the morphological properties of the executing agent. As an effect of proper or improper shape and weight distribution, the given task can be simplified or in contrast to that be complicated or even be ruled out by creating situations from which it is impossible to solve the task at all. This effect can easily exploited if considered early in the design phase. Looking at toys a robot's infrastructure can be hidden under bizarre shaped plastic covers. If designed properly the decorative shell can serve as effortless enhancement of performance. Further a designer must also consider the used material very carefully.

The overall ranking elaborated throughout the thesis is listed in Table 8.1 on the following page. The numbers determine the rank of the respective vocabulary depending on the

shape of the head and the property. Co and Flx are ordered numerically. The higher the value, the higher the rank. Note that for those measure a higher value is not a sign of superior quality, since they only serve to categorise the distinct vocabularies with respect to their inner correlation and flexibility. Unfortunately, since ODE lies its main attention on speed and not on accuracy, it was impossible to derive quantitative results.

Vocabulary	Co	Flx	Learning Progress			BDI			Robust-ness		
			O	V	R	O	V	R	O	V	R
1	1	4	6	3	1	5	4	4	4	5	5
2	2	4	2	2	2	2	2	2	2	2	2
3	5	4	5	4	4	6	6	6	6	4	4
4	3	1	4	5	5	3	5	3	6	6	6
5	4	2	1	1	3	1	1	1	1	1	1
6	6	3	3	6	6	4	3	5	3	3	3

Table 8.1.: Final ranking of vocabularies in respect of morphology and property
A value of 1 signifies the best, 6 the worst result in the respective category (exceptions are Co and Flx which are ordered numerically without valuation of quality). The overall ranking depends on how important a researcher considers the respective property.

The ranking of the vocabularies 2 and 5 is almost constant, irrespective of property and morphology. The overall ranking depends on how important a researcher considers the respective property.

Since those vocabularies that sick out, whether in good or not, are mainly the same, there seems to lie a hidden system behind our results. This qualitative conclusion encourages us to follow up our matter.

Our framework is just a small step on our way to understand why nature favours certain principles, how most advantageous motor primitives can be derived and to what extent they depend on the particular morphology. Such insights would put researchers in the position to design best possible robots with a maximum of usability.

Yet, being unable to derive clear quantitative rules at this point in time, a fundament for systematic investigations of many different morphologies was established in order to build a framework of benchmarks. In doing so, those landmarks of exemplary morphologies serve to predict the performance of unexplored robots, of new robots, if they share one or more of the morphological characteristics with one of the basis points which means an agent that was examined earlier. This kind of parameter directory follows the example of Bongard and Pfeifer's framework for evolved behaviours in section 2.3. Merging both methodologies would definitely enrich the basis for potential design principles.

In the last section, we will elaborate several assignments for our ongoing studies.

8.2. Outlook

This project bears much more potential than can be investigated in the timeframe of this thesis. Some of the ideas that directly hook up on the current framework and/or an MiniDog6M will be presented.

8.2.1. Future trends for MiniDog6M

Having a closer at the irregular motion patterns in Figure 2.11 on page 20, the robot seems to tumble, but instead of falling down, it recovers and carries on it way. Similar patterns can also be found other walking machines with cyclic movement. Whereas fully controlled motions only reveal this type of pattern when for instance the robot snags its foot on a little stone or steps into a hole, we find that this irregularity is symptomatic for the springy locomotion of the "running dog project". Since the just mentioned recovering qualities from minor disturbances are an inherent property of oscillation, similar effects of self-stabilisation come forward in MiniDog6M's simple sinusoidal control. However, this theory has to be proven in a subsequent project.

Another objective for MiniDog6M is to advance gait control and hopping height respectively width by adequate support of the spine. In addition, learning of speed variation control on different surfaces by means of amplitude, frequency, phase difference and maybe offset variation should be put forward.

Moreover, further sensors such as pressure sensors, whiskers [Fend 04b, Fend 04a] etc could be integrated in a subsequent project. Extending MiniDog6M's sensory system will help to better attitude recognition. Talking about the emergence of behaviours, one framework supporting this very principle is Distributive Adaptive Control (DAC). DAC is also called embedded artificial neural network, since it is not trained in isolation, but learns through physical interaction with the environment. Hence behaviours can emerge in the course of proper physical design. This is possible only because of correlation between various sensory channels. Being know as the so called "redundancy principle", this overlap in the information channels of the agent must be provided by different sensor modalities. Thinking of behavioural diversity, additional sensor data could be used as a basis to choose between the different behaviours. Another option here is that the quadruped learns how to use the twist motor and/or a catlike tail to counteract falling.

In section 2.4.3, a mainly reactive, behaviour based architecture with few planning components for BISAM was recapitulated. This model consists of a network of different competences, each of which generates motor output as soon as its specific goal is not met. These concurrent behaviours are merged in either by superposition or by special knots. This architecture is most compatible with our idea of intermixing motor primitives. Moreover, it also complies with the principle of parallel, loosely coupled processes [Pfeifer 03b, Pfeifer 99]. "Loosely coupled refer to coupling through the interaction with the environment and in that sense is used as opposite of the strong coupling in a hierarchical architecture. Such an architecture also encourages the emergence of new behaviours. We keep this architecture in mind for a later stage of our project, when MiniDog6M has to cope a larger variety of tasks.

8.2.2. Conceptual research extending our framework

In order to derive more general design principles for meaningful motor primitives, it is necessary to examine more variations. Therefore, we will investigate not only more vocabularies, but also further morphological changes such as length or rest position of the legs, changing shape and weight of head and rear etc. It would be nice to investigate the feasibility of the gathered behaviours on uneven or elastic surface. Moreover, the experiments must be extended to many different platforms, e.g. biped and hexapod.

As a reaction to significant environmental changes and new challeges, the robot, even if its behaviours are extremely robust, should be able to slightly change the frequencies in order to adapt to the new situation. Needless to say that, for efficient adaptation, robust motor primitives with least possible requirement of adaptation should be preferred. In order to enable adaptation, we can either allow the controller to slightly change the respective frequencies or we can provide our robot with a set of different frequencies which can be exchanged with the original frequency is necessary.

Anyway, it is still important for an autonomous agent to be equipped with online learning. Be it for administrative levels in a hierarchical architecture, that learns how to chose from a batch of available solutions or to acquire solutions for new tasks. Unfortunately a constant learning rate hits one of the weak points of Q-Learning. Due to a constant learning rate unequal zero there will always be sub-optimal decisions. Since the agent does not always act according to the learned policy, it is possible that the policy will be unlearned. So, for future projects, instead of our ordinary Q-learning, we suggest to improve the algorithm according to “risk-free reinforcement learning” proposed by Matthias Heger and Karsten Berns [Heger 92]. The enhancements range from elimination of parameters, over insensibility towards quantification errors, provision for state loops and constancy problems in state recognition to improvements concerning stochastic behaviour.

However, the evasion of the symbol-grounding problem still remains an open duty in our project. An important assignment herein is to examine the competence introduced by sensory-motor-coordination through analysis of sensor patterns resulting from the different vocabularies. For this purpose, a platform with many different sensor modalities is needed. Furthermore, we would like to see if supplementary sensors influence the performance of our learning progress.

Following potential guidlines for motor primitive, a good vocabulary could be self-acquired by a robot with a morphology that was optimised beforehand. The designer would just have to provide the agent with the desired weights for the different qualities e.g. high BDI required, slow learning can be tolerated. One approach could be the stepwise refinement of motor primitives. Bernstein [Rosenbaum 96] found that in order to solve the degree-of-freedom-problem, humans often freeze some of the joints and thereby reduce their active degrees of freedom. Those joints are freed up with practice and thus serve to optimise the learn behaviour. This idea can easily be transferred to our concepts by systematically restricting joints to mid position, for instance running, as it is here, can later be improved by support from the spinal motors.

After all, it would be very interesting to compare the learning progress of our approach with that of other learning methods, such as actor critic, Hebb learning etc and thereby introduce an additional dimension into our framework. A researcher would then be in the position to choose the most suitable learning algorithm and setting for his or her particular robot.

The ultimate goal of this project would be the self-acquisition of the (here predefined) motor primitives, while building up a body image of its own. One way of acquiring a body image is by learning a model for the state transitions in dependency of the executed action. This involves not only the standing up task, but also the control of running behaviour. The robot is then in the position, that it can substitute trial and error with planning in advance. Otherwise, it would be much more interesting, if a body image could (partly) emerge and herein acquire the best fitting vocabulary.

Thinking it out, it would be possible to create a tool that, if provided with some parameters, automatically computes a (near) optimal morphology and controller for given and emerging tasks.

A. Glossary

Action Period

Duration respectively execution time of the given action (here: *NumTicks* · ODE step size)

Behavioural diversity

If a task can be solved in more than one way.

Behavioural-Diversity-Index BDI

Product of diversity factor and average number of solutions for a given task

Cheap design

Parsimonious approach to designing robotic systems

Clusters of interest

Group of motors that serve the same purposes regarding their position and effective direction within the robot e.g. shoulder and hip motors for locomotion

Coherence-Index Co

Task an platform independent measure for the inner correlation and similarity to a special root posture. Yields a numerical value between 0 and 1. (0 denotes a maximum unlike posture. 1 means that all motors are controlled with the same frequency and that the agent ends up in root posture.)

Diversity factor D

Task and platform independent measure for the dissimilarity within a set of solution for a given task. Yields numerical value between 0 and 1. (The more the solutions have in common, the higher the value.)

Don't-Care

Special stop symbol that causes the motor to remain in its current postion. Don't-Care is used if a motor primitive does not consider one or more motors. These motors can consequently have an arbitrary position since their position is determined by the last motor primitive that directly set it.

Ecological balance

Equal complexity of a robot's task, morphology and controller

Feature

In the RLT, the neurons of the hidden layer are called feature.

Form stable

Behaviours are form stable if they are tolerant against morphological changes.

Flexibility-Index Flx

Ratio of Don't-Care terms within a motor primitive and total amount of motors

Goal position

Motor angle after executing a motor primitve

Linear approximator

Mathematical method to approximate an unknown function by means of linear combination of a set of basis functions

Local receptive field

Speciality in a RBF-NN, the activation function (here: Gauss) is only unequal zero for a small part of the input space. This region of limited size is called local receptive field and is bounded by means of standard deviation. The mean lies in the centre of such a region.

Motor primitive

Low level motor program assigning a controller frequency for each motor

Morphology

Shape, sensor placement, actuators and materials of a robot

Morphological computation

Reduction of controller complexity by exploiting morphological properties such as shape, material, sensor placement etc.

Open Dynamics Engine

Free, industrial strength C++ library for simulating articulated rigid body dynamics in a physically realistic environment

Position stable

Behaviour are position stable if they are valid for all initial positions

Q-Learning

Popular approach of reinforcement learning that tries to estimate the future discounted reward if an agent chooses the unexplored or suboptimal action a in state s and then resumes the policy π. The name is derived from this estimation function called Q-Function Q(s, a).

Reinforment learning

Acquisition of a task fulfilling sensory-motor control strategy through trial and error. During the process of learning, the adaptive system undertakes some actions that affect its environment. Hereupon it is reinforced by receiving a scalar evaluation of its actions. This reinforcement signal is generally known as “reward”. The reinforcement learning strategy stresses outputs that maximise the received reward over time.

Reinforcement Learning Toolbox (RLT)

C++ based framework that provides a variety of reinforcement learning algorithms.

Radial Basis Function network

Fully connected three layer neural network with radial activation function (mostly the Gauss function) in the inner layer

Robustness

Property of behaviours that are tolerant against changes in morphology (form stable), environment and posture (position stable)

Root posture

Basic posture of a robot. This can be natural rest position or a special designated posture, e.g. starting posture. (Here: all motors in position 0 respectively frequency 0)

Step size

In ODE, each integration step advances the current time by a given step size.

Transfer capability

Ability to transform a given problem into another (basic) problem e.g standing up from the left into standing up from the right hand side.

Vocabulary

Set of motor primitives

B. Vocabularies in this thesis

B.1. Group A of vocabularies

Set1	FrontRight	Bend	HindRight	Twist	HindLeft	FrontLeft
Run:	f_R	0	f_R	0	f_R	f_R
AC1:	0	0	0	0	0	0
AC2:	0	$-f$	0	0	0	0
AC3:	0	f	0	0	0	0
AC4:	0	0	0	f	0	0
AC5:	0	0	0	$-f$	0	0
AC6:	$-f$	0	$-f$	0	$-f$	$-f$
AC7:	f	0	f	0	f	f
AC8:	f	0	f	0	$-f$	$-f$
AC9:	$-f$	0	$-f$	0	f	f

Set2	FrontRight	Bend	HindRight	Twist	HindLeft	FrontLeft
Run:	f_R	0	f_R	0	f_R	f_R
AC1:	$-f$	0	$-f$	f	$-f$	$-f$
AC2:	$-f$	$-f$	0	0	0	$-f$
AC3:	0	0	$-f$	0	$-f$	0
AC4:	0	0	0	f	0	0
AC5:	f	0	f	0	0	0
AC6:	$-f$	0	$-f$	0	$-f$	$-f$
AC7:	0	0	0	0	0	0
AC8:	f	0	f	0	$-f$	$-f$
AC9:	0	f	0	0	0	0

Set3	FrontRight	Bend	HindRight	Twist	HindLeft	FrontLeft
Run:	f_R	0	f_R	0	f_R	f_R
AC1:	$-f$	f	$-f$	$-f$	$-f$	$-f$
AC2:	f	f	f	f	f	f
AC3:	f	$-f$	0	f	0	0
AC4:	f	$-f$	f	0	f	$-f$
AC5:	0	0	0	$-f$	0	0
AC6:	f	0	f	f	$-f$	$-f$
AC7:	$-f$	0	$-f$	0	$-f$	$-f$
AC8:	0	f	0	f	0	0
AC9:	f	0	f	0	$-f$	$-f$

B.2. Group B of vocabularies

Set4	FrontRight	Bend	HindRight	Twist	HindLeft	FrontLeft
Run:	f_R	0	f_R	0	f_R	f_R
AC1:	0	0	0	0	0	0
AC2:	x	$-f$	x	x	x	x
AC3:	x	f	x	x	x	x
AC4:	x	x	x	f	x	x
AC5:	x	x	x	$-f$	x	x
AC6:	$-f$	0	$-f$	0	$-f$	$-f$
AC7:	f	0	f	0	f	f
AC8:	f	0	f	0	$-f$	$-f$
AC9:	$-f$	0	$-f$	0	f	f

Set5	FrontRight	Bend	HindRight	Twist	HindLeft	FrontLeft
Run:	f_R	0	f_R	0	f_R	f_R
AC1:	$-f$	0	$-f$	f	$-f$	$-f$
AC2:	$-f$	$-f$	0	0	0	$-f$
AC3:	x	x	$-f$	x	$-f$	x
AC4:	x	x	x	f	x	x
AC5:	f	0	f	0	0	0
AC6:	$-f$	0	$-f$	0	$-f$	$-f$
AC7:	0	0	0	0	0	0
AC8:	f	0	f	0	$-f$	$-f$
AC9:	0	f	x	x	0	0

Set6	FrontRight	Bend	HindRight	Twist	HindLeft	FrontLeft
Run:	f_R	0	f_R	0	f_R	f_R
AC1:	$-f$	f	$-f$	$-f$	$-f$	$-f$
AC2:	f	f	f	f	f	f
AC3:	f	$-f$	0	f	0	0
AC4:	f	$-f$	f	0	f	$-f$
AC5:	x	x	x	$-f$	x	x
AC6:	f	0	f	f	$-f$	$-f$
AC7:	$-f$	0	$-f$	0	$-f$	$-f$
AC8:	x	f	x	f	x	x
AC9:	f	0	f	0	$-f$	$-f$

C. Assortment of transfer capabilities and fastest sequences to stand up

The data below represents the behavioural diversity and transfer capabilities of the simulated MiniDog6M accommodated with vocabulary 4. Yet, the goal positions are assessed directly which means without sinusoidal control function.

Back

```
1 8 4 0 --> basic position: 0
2 8 4 0 --> basic position: 0
3 2 5 0 --> basic position: 0
3 2 8 0 --> basic position: 0
3 5 3 0 --> basic position: 0
3 5 4 0 --> basic position: 0
3 8 3 0 --> basic position: 0
4 1 7 0 --> basic position: 0
4 2 5 0 --> basic position: 0
4 2 6 0 --> basic position: 0
4 2 7 0 --> basic position: 0
4 3 7 0 --> basic position: 0
4 5 4 0 --> basic position: 0
4 6 4 0 --> basic position: 0
4 6 3 0 --> basic position: 0
4 7 1 0 --> basic position: 0
4 7 4 0 --> basic position: 0
5 1 6 0 --> basic position: 0
7 3 1 0 --> basic position: 0
7 3 5 0 --> basic position: 0
7 4 2 0 --> basic position: 0
8 1 4 0 --> basic position: 0
8 3 4 0 --> basic position: 0
8 4 5 0 --> basic position: 0
```

```
3 0 --> basic position: 1
3 1 0 --> basic position: 1
3 2 0 --> basic position: 1
3 4 0 --> basic position: 1
3 5 0 --> basic position: 1
3 7 0 --> basic position: 1
3 8 0 --> basic position: 1
7 3 0 --> basic position: 1

4 0 --> basic position: 2
4 1 0 --> basic position: 2
4 2 0 --> basic position: 2
4 3 0 --> basic position: 2
4 5 0 --> basic position: 2
4 6 0 --> basic position: 2
4 7 0 --> basic position: 2
4 8 0 --> basic position: 2
7 4 0 --> basic position: 2
8 4 0 --> basic position: 2

5 1 0 --> basic position: 4
1 5 1 0 --> basic position: 4
2 5 1 0 --> basic position: 4
4 7 2 0 --> basic position: 4
5 1 2 0 --> basic position: 4
```

```
5 1 3 0 --> basic position: 4
5 1 4 0 --> basic position: 4
5 1 5 0 --> basic position: 4
5 3 1 0 --> basic position: 4
5 4 1 0 --> basic position: 4

3 2 4 0 --> basic position: 5
3 7 4 0 --> basic position: 5
4 2 3 0 --> basic position: 5


Head

6 0 --> basic position: 0
7 0 --> basic position: 0
8 0 --> basic position: 0
2 6 0 --> basic position: 0
2 7 0 --> basic position: 0
2 8 0 --> basic position: 0
3 7 0 --> basic position: 0
3 8 0 --> basic position: 0
4 6 0 --> basic position: 0
6 1 0 --> basic position: 0
6 3 0 --> basic position: 0
6 4 0 --> basic position: 0
7 1 0 --> basic position: 0
7 4 0 --> basic position: 0
8 1 0 --> basic position: 0
8 3 0 --> basic position: 0

4 8 0 --> basic position: 1
8 2 0 --> basic position: 1
8 4 0 --> basic position: 1
1 7 3 0 --> basic position: 1
2 4 7 0 --> basic position: 1
2 8 4 0 --> basic position: 1
3 4 8 0 --> basic position: 1
4 1 8 0 --> basic position: 1
4 3 8 0 --> basic position: 1
4 8 1 0 --> basic position: 1
4 8 4 0 --> basic position: 1
8 1 2 0 --> basic position: 1
8 2 4 0 --> basic position: 1
8 2 5 0 --> basic position: 1
8 3 4 0 --> basic position: 1
8 4 1 0 --> basic position: 1
8 4 2 0 --> basic position: 1
8 4 5 0 --> basic position: 1
8 4 7 0 --> basic position: 1
```

```
7 3 0 --> basic position: 2
2 7 3 0 --> basic position: 2
3 7 3 0 --> basic position: 2
3 8 1 0 --> basic position: 2
3 8 2 0 --> basic position: 2
3 8 4 0 --> basic position: 2
6 2 3 0 --> basic position: 2
6 3 2 0 --> basic position: 2
7 1 3 0 --> basic position: 2
7 2 3 0 --> basic position: 2
7 2 5 0 --> basic position: 2
7 2 7 0 --> basic position: 2
7 3 1 0 --> basic position: 2
7 3 2 0 --> basic position: 2
7 3 5 0 --> basic position: 2
7 3 8 0 --> basic position: 2
7 4 3 0 --> basic position: 2
7 4 7 0 --> basic position: 2
8 1 4 0 --> basic position: 2

1 6 0 --> basic position: 3
1 7 0 --> basic position: 3
1 8 0 --> basic position: 3
4 7 0 --> basic position: 3

2 4 1 0 --> basic position: 5
7 4 8 0 --> basic position: 5


Right

1 7 0 --> basic position: 0
2 7 0 --> basic position: 0
3 7 0 --> basic position: 0
4 7 0 --> basic position: 0
5 4 0 --> basic position: 0
6 3 0 --> basic position: 0
6 4 0 --> basic position: 0
7 1 0 --> basic position: 0
7 4 0 --> basic position: 0

8 3 2 0 --> basic position: 1
8 2 3 0 --> basic position: 1
7 4 8 0 --> basic position: 1

2 4 0 --> basic position: 3
1 2 4 0 --> basic position: 3
2 4 2 0 --> basic position: 3
2 4 3 0 --> basic position: 3
2 4 6 0 --> basic position: 3
```

```
2 4 7 0 --> basic position: 3
2 8 3 0 --> basic position: 3
3 8 3 0 --> basic position: 3
6 1 4 0 --> basic position: 3
8 2 4 0 --> basic position: 3
8 3 5 0 --> basic position: 3
8 3 6 0 --> basic position: 3
8 3 1 0 --> basic position: 3
8 3 7 0 --> basic position: 3
8 4 6 0 --> basic position: 3

7 2 0 --> basic position: 4
1 7 2 0 --> basic position: 4
2 7 2 0 --> basic position: 4
3 2 4 0 --> basic position: 4
3 6 2 0 --> basic position: 4
3 7 2 0 --> basic position: 4
4 2 6 0 --> basic position: 4
4 7 2 0 --> basic position: 4
5 3 1 0 --> basic position: 4
6 4 2 0 --> basic position: 4
7 1 2 0 --> basic position: 4

4 2 3 0 --> basic position: 5
5 2 3 0 --> basic position: 5
5 4 2 0 --> basic position: 5


Left
```

```

5 3 0 --> basic position: 0
8 3 0 --> basic position: 0
5 4 0 --> basic position: 0
2 8 0 --> basic position: 0

7 4 0 --> basic position: 2
1 7 4 0 --> basic position: 2
2 7 4 0 --> basic position: 2
4 7 4 0 --> basic position: 2
6 4 2 0 --> basic position: 2
7 2 4 0 --> basic position: 2
7 3 4 0 --> basic position: 2
7 2 8 0 --> basic position: 2
7 4 5 0 --> basic position: 2
7 4 8 0 --> basic position: 2

2 3 0 --> basic position: 3
6 3 0 --> basic position: 3
6 4 0 --> basic position: 3

2 8 2 0 --> basic position: 4
4 2 3 0 --> basic position: 4
4 8 2 0 --> basic position: 4

3 2 4 0 --> basic position: 5
3 7 4 0 --> basic position: 5
5 2 4 0 --> basic position: 5
5 4 2 0 --> basic position: 5
```

D. Training patterns for supervised learning

The data below represents the learning base for supervised learning which is derived from the soultions for vocabulary 4 which are presented in Appendix C. Yet, the goal positions are assessed directly which means without sinusoidal control function.

```
0 o o o o o o -RUN
0 o o o L o o -7
0 H o H o L L -0
0 H o H L H H -0
0 H o H L L L -0
0 H L H o L L -0
0 L o L H H H -0
0 L o L o L L -0
0 H o H H L L -0
0 L L L o H H -0
0 o o o o H H -0
0 o o o H o o -8
0 H o H o H H -1
0 H L H o H H -0

1 o o o o o o -8
1 o L o o o o -5
1 L o L o L L -3
1 L o L o H H -2
1 o H o o o o -8
1 o H o H o o -5
1 o o o H o o -5
1 o o o L o o -5
1 H o H o L L -3
1 H H H o L L -5
```

```
1 H H H H L L -0
1 L H L o H H -1
1 L o L L H H -3
1 L o L L L L -0


2 o o o o o o -7
2 o L o o o o -7
2 o H o o o o -7
2 H o H o L L -0
2 L o L o L L -4
2 H o H o H H -3
2 H o H H H H -0
2 L o L H L L -0

3 x o x o x x -4

4 o o o o o o -6
4 o H o o o o -8

5 x o o o o x -5
5 x x L x L x -0
```

Bibliography

[Arbib 81] M. A. Arbib. Perceptual structures and distrubed motor control. 2:1449–1480, 1981.

[Bizzi 84] E. Bizzi, N. Acornero, W. Chapple, N. Hogan. Posture control and trajectory formation during arm movement. 4:2738–2744, 1984.

[Bongard 03] Josh C. Bongard, Rolf Pfeifer. Evolving Complete Agents using Artificial Ontogenyg. 2003.

[Büschges 05] Ansgar Büschges. Sensory Control and Organization of Neural Networks Mediating Coordination of Multisegmental Organs Organs for Locomotion. *Journal of Neurophysiology*, Februar 2005.

[Buehler 00] M. Buehler, U. Saranli, D. Papadopoulos, D. Koditschek. Dynamic locomotion with four and six-legged robots. In *International Symposium on Adaptive Motion of Animals and Machines*, 2000.

[FastRunning 05] Biologically-Inspired Fast Running Robots, Prof. Emura's Mechatronics Design Laboratory, Tohoku University, Japan. `http://www.mechatronics.mech.tohoku.ac.jp/research/Kenken/kenken_en.htm`, 2005.

[Feldman 80] A. Feldman. Superposition of motor programs, I. Rhythmic forearm movement in man. 5:81–90, 1980.

[Fend 04a] M. Fend, R. Abt, M. Diefenbacher, S. Bovet, M. Krafft. Morphology and Learning - A Case Study on Whiskers. In S. Schaal, A. Ijspeert, A. Billard, S. Vijayakumar, J. Hallam, J. Meyer (Hrsg.), *From Animals to Animats*, S. 114–121. MIT Press, 2004.

[Fend 04b] M. Fend, S. Bovet, , V Hafner. The Artificial Mouse - A Robot with Whiskers and Vision. 2004. on CD.

[Ferrell 95] Cynthia Ferrell. A Comparison of Three Insect Inspired Locomotion Controllers. 1995.

[Ferris 98] Daniel P. Ferris, Micky Louie, Claire T. Farley. Running in the real world: adjusting leg stiffness for different surfaces. *UCB Locomotion Laboratory, Berkeley University of California, USA, The Royal Society*, 1998.

[Giszter 93] Simon F. Giszter, Fernando A. Muss-Ivaldi, Emilio Bizzi. Convergent Force Fields Organized in the Frog's Spinal Cord. *Journal of Neuroscience*, Februar 1993.

[Heger 92] Matthias Heger, Karsten Berns. Risikoloses Reinforcement-Lernen. 1992.

[Iida 03] Fumiya Iida, Rolf Pfeifer. "Cheap" Rapid Locomotion of a Quadruped Robot: Self-Stabilization of Bounding Gait. 2003.

[Iida 04a] Fumiya Iida, Rolf Pfeifer. "Cheap" Rapid Locomotion of a Quadruped Robot: Self-Stabilization of Bounding gait. In *Intelligent Autonomous Systems 8*, 2004.

[Iida 04b] Fumiya Iida, Rolf Pfeifer. Self-Stabilization and Behavioural Diversity of Embodied Adaptive Locomotion. S. 119–129, 2004.

[Ilg 00] W. Ilg, J. Albiez, H. Witte, R. Dillmann. Adaptive posture control of a four-legged walking machine using some principles of mammalian locomotion. In *Proceedings of the International Symposium on Adaptive Motion of Animals and Machines*, 2000.

[Ilg 98] W. Ilg, K. Berns, H. Jedele, J. Albiez, R. Dillmann, M. Fischer, H. Witte, J. Biltzinger, R. Lehmann, N. Schilling. BISAM: From Small Mammals to a Four Legged Walking Machine. In *Proceedings of the Fifth International Conference on Simulation of Adaptive Behaviour*, 1998.

[Ilg 99] W. Ilg, J. Albiez, H. Jedele, K. Berns, R. Dillmann. Adaptive periodic movement control for the four legged walking machine BISAM. In *IEEE-Conference Robotics and Automation*, 1999.

[Kimura 00] H. Kimura, Y Fukuoka, Y. Hada, K. Takase. Three-dimensional adaptive dynamic walking of a quadruped robot by using neural system model. *CLAWAR 2000*, 2000.

[Kimura 01] Hiroshi Kimura, Yasuhiro Fukuoka. Dynamics Based Motion Adaptation for a Quadruped Robot. 2001.

[Kimura 03] Hiroshi Kimura, Yasuhiro Fukuoka, Ken Konaga. Adaptive dynamic walking of a quadruped robot by using neural system model. *The International Journal of Robotics Research*, 22:187–202, 2003.

[Kimura 04] Hiroshi Kimura, Yasuhiroo Fukuoka. Biologically Inspired Adaptive Dynamic Walking in Outdoor Environment Using a Self-contained Quadruped Robot: Tekken2. 2004.

[Kimura 99] Hiroshi Kimura, Seiichi Akiyama, Kazuaki Sakurama. Realization of Dynamic Walking and Running of the Quadruped Using Neural Oscillator. *Autonomous Robots*, 7(3), November 1999. Kluwer Academic Publishers, Boston.

[Kuniyoshi 04] Yasuo Kuniyoshi, Yasuaki Yorozu, Yoshiyuki Ohmura, Koji Terada, Akihiko Nagakubo, Tomoyuki Yamamoto. From Humanoid Embodiment to Theory of Mind. In Fumiya Iida, Rolf Pfeifer, Luc Steels, Yasuo Kuniyoshi (Hrsg.), *Embodied Artificial Intelligence (LNAI3139)*, S. 202–218. Springer, Berlin, 2004.

[Kuniyoshi 94] Yasuo Kuniyoshi, Masayuki Inaba, Hirochika Inoue. Learning by watching: Extracting Reusable Task Knowledge from Visual Observation of Human Performance. In *IEEE Transactions on Robotics and Automation*, 1994.

[Lichtensteiger 00] Lukas Lichtensteiger. Evolving Sensor Morphologies using Adaptive Hardware. `http://www.ifi.unizh.ch/ailab/projects/eyebot/`, 2000.

[Luksch 02] Tobias Luksch. Verhaltensbasierte freie Gangart für eine vierbeinige Laufmaschine. Diplomarbeit, 2002.

[Lund 97] H. Lund, J. Hallam, W. Lee. Evolving Robot Morphology, 1997.

[Mataric 02] Maja J. Mataric. Sensory-motor primitives as a basis for imitation: linking perception to action and biology to robotics. S. 391–422, 2002.

[Mataric 98] Maja J. Mataric. Behaviour-Based Robotics as a Tool for Synthesis of Artificial Behavior and Analysis of Natural Behavior. *Trends in Cognitive Science*, 2(3), 1998.

[McGeer 90a] T. McGeer. Passive dynamic walking. 9:62–82, 1990.

[McGeer 90b] T. McGeer. Passive walking with knees. In *IEEE COnference on Robotics and Automation*, S. 1640–1645, 1990.

[Micronics 05] Star Micronics. Tri-Axial Acceleration Sensor on a Single element. `http://www.designinfo.com/FeaturedProducts/Detail?ExhibitID=19346&fromSupplier=1`, 2005.

[Mussa-Ivaldi 94] F.A. Mussa-Ivaldi, S.F. Giszter, E. Bizzi. Linear combinations of primitives in vertebrate motor control. 1994.

[Neumann 05] Gerhard Neumann. The Reinforcement Learning Toolbox, Reinforcement Learning for Optimal Control Tasks. Diplomarbeit, Institute of Computer Science, TU-Graz, May 2005.

[Orr 96] Mark J.L. Orr. Introduction to Radial Basis Function Networks. 1996.

[Paine 04] Rainer W. Paine, Jun Tani. Adaptive Motor Primitive and Sequence Formation in Hierarchical Recurrent Neural Network. *SAB*, 2004.

[PassiveRunning 05] Analysis and Control of Passive Running Robots homepage, Prof. Emura's Mechatronics Design Laboratory, Tohoku University, Japan. `http://www.mechatronics.mech.tohoku.ac.jp/index.html.en`, 2005.

[Paul 01] Chandana Paul, Josh C. Bongard. The Road Less Travelled: Morphology in the Optimization of Biped Robot Locomotion. In *The IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

[Pfeifer 03a] Rolf Pfeifer. AI lectures of Tokyo. `http://tokyolectures.org`, 2003/4.

[Pfeifer 03b] Rolf Pfeifer, Fumiya Iida, Josh Bongard. *New Robotics: Design Principles for Intelligent Systems*. 2003.

[Pfeifer 05] Rolf Pfeifer, Fumiya Iida. Morphological computation: connecting body, brain and environment. 2005.

[Pfeifer 99] Rolf Pfeifer, Christian Scheier. *Understanding Intelligence*. MIT Press, 1999.

[Poulskakis 05] Ioannis Poulskakis, James A. Smith, Martin Buehler. Modeling and Experiments of Untethered Quadrupedal Running with a Bounding Gait: The Scout II Robot. *The International Journal of Robotics Research*, 24:239–256, 2005.

[Raibert 85] M. H. Raibert, Jr. H. B. Brown, M. Chepponis, J. Hodgins, J. Kroechling, Miller J., K. N. Murphy, S. S. Murthy, A. Stentz. Dynamically Stable Legged Locomotion. 1985.

[Rosenbaum 96] David A. Rosenbaum, Ruud G. J. Meulenbroek, Jonathan Vaughan. Three Approaches to the Degrees of Freedom Problem in Reaching. S. 169–185, 1996.

[RunningDog 05] Running Dog Robot Project Homepage. `http://www.ifi.unizh.ch/ailab/people/iida/index.html`, 2005.

[Schaal 00] Stefan Schaal, Shinya Kotosaka, Dagmar Sternad. Nonlinear Dynimcal Systems as Movement Primitives. In *IEEE International Conference on Humanoid Robotics*, 2000.

[Schaal 02] Stefan Schaal. *Learning Robot Control, The handbook of brain theory and neural networks*. MIT Press, Ausgabe 2, 2002.

[Schaal 04] S Schaal, J. Peters, J. Nakanishi, A. Ijspeert. Learning Movement Primitives. In *International Symposium on Robotics Research (ISRR2003)*. Springer, 2004.

[Schaal 99] Stefan Schaal. Is Imitation Learning the Route to humanoid Robots. *Trends in Cognitive Science*, 1999.

[Seyfarth 02] Andre Seyfarth, Hartmut Geyer, Michael Günther, Reinhard Blickhan. A movement criterion for running. 35:649–655, 2002.

[Smith 04] Russel Smith. *Open Dynamics Engine v0.5 User Guide*, 2004.

[Sutton 98] Richard S. Sutton, Andrew G. Barto. *Reinforcement Learning - An Introduction*. MIT Press, Boston, 1998.

[Tani 02] Jun Tani. Articulation of Sensory-Motor Experiences by "Forwarding Forward Model": From Robot Experiments to Phenomenology. *SAB*, 2002.

[Ziegler 05] M. Ziegler, F. Iida. Cheap underwater locomotion: Influence of material properties and behavioural diversity. 2005.

List of Algorithms

List of Figures

List of Tables