Development of a Planning System for the WalkNet

Dissertation
zur Erlangung des Akademischen Grades
des Doktors der Naturwissenschaften
an der Fakultät für Biologie,
Universität Bielefeld

vorgelegt von
Thorsten Roggendorf

Januar 2006

Charly

Contents

Versicherung der Autorenschaft

Ich versichere, dass ich die vorliegende Dissertation selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Stellen, die dem Sinn nach anderen Werken entnommen sind, habe ich ich in jedem einzelnen Fall unter genauer Angabe der Quelle deutlich als Zitat kenntlich gemacht.

Bielefeld, den 04.01.2006

(Thorsten Roggendorf)

Erklärung der Beteiligten der Gruppenarbeit

Dipl. Ing. Dipl. Inform. Axel Schneider (Tel. 106-5519) ist Autor der letzten beiden Absätze des Unterabschnitts "Problems with positive feedback as implemented in the Walknet" im Abschnitt 2.5.1.
Thorsten Roggendorf (Tel. 106-5528) ist Autor aller anderen Teile dieser Arbeit. Beide Autoren sind Mitglieder der Abteilung 4 "Biologische Kybernetik und Theoretische Biologie", Fakultät Biologie, Universität Bielefeld, Postfach 10 01 31.

Bielefeld, den 04.01.2006

(Thorsten Roggendorf) (Axel Schneider)

Dank ...

... gebührt: Charly, die etliche Deadlines verstreichen sah und ihren eigenen Beruf für diese Arbeit zurückgestellt hat; Holk, der das Thema vorschlug, Ideen gab, konstruktiv kritisierte und sich vorbehaltlos mit Kritik an seinen Vorstellungen auseinandersetze, den vorliegenden Text korrigiert und editiert hat und ein hervorragender Chef ist; Simone, die angenehmste Kollegin, die ich mir vorstellen kann; Abt4 für Kritik, Kaffee, Kekse und klug eingesetztes Halbwissen – Eure sinnfreie intelektuelle Unternehmungslust wird mir am meisten fehlen.

List of Publications

Roggendorf (2005) Comparing different controllers for the coordination of a six legged walker. *Biological Cybernetics,* 92:261–274, 2005
See chapter 4

Roggendorf (submitted) Angular-, Momentum-, Non-relaxing-MMC; Dynamical Weighting. Submitted to *Computers & Mathematics with Applications*
See chapter 3

Roggendorf and Schneider (in prep.) Stance Trajectory Controllers for Hexapod Walking and Climbing. In preparation
See chapter 2.

Zusammenfassung

Menschen sind offenbar in der Lage ihre Bewegungen mental durch zu spielen, ohne sie auszuführen. Dieses Phänomen wird allgemein als "Probehandeln" bezeichnet und als kognitive Fähigkeit aufgefasst [84]. Probehandeln kann bei der Bewältigung schwieriger motorischer Aufgaben helfen.
Man stelle sich z.B. einen Menschen vor, der einen Baum erklettert. Der Kletterer wird immer wieder in Situationen kommen, in denen er aus seiner momentanen Haltung den nächsten Ast nicht erreichen kann. Oft wird es aber möglich sein, den Ast zu erreichen, wenn der Kletterer seinen Körper in die Entsprechende Richtung verlagert. Ob eine solche Verlagerung ausreicht ist jedoch nicht immer offensichtlich. Hinzu kommt die Frage, in welcher Reihenfolge die Hände und Füße am besten auf neue Äste gesetzt werden, ohne dabei das Gleichgewicht zu verlieren.
Um solche Probleme mental durch zu spielen, benötigt der Kletterer eine mentale Repräsentation seines Körpers und der fraglichen Äste. Jedoch steht eine solches Repräsentation vermutlich nicht allein, sondern ist vielmehr mit zahlreichen reaktiven Systemen zur Bewegungssteuerung verknüpft [68, 53, 71]. Möglicher Weise ist das Planungs-System selbst ein Teil der motorischen Steuerung. Das hieße "Denken" ist neuronal identisch mit Handeln – lediglich die Muskeln sind während des Denkens "abgeschaltet" [106, 71].
Die vorliegende Simulations-Studie stellt die Entwicklung eines derartigen Planungssystems dar. Da für die Simulation reaktiver Steuerungs-Systeme für Stabheuschrecken deutlich mehr Vorarbeiten existieren als für Menschen, werden Bewegungen dieser Insekten simuliert und kontrolliert. Zur Entwicklung des Planungssystems war es zunächst notwendig, eine einfache und zuverlässige reaktive Steuerung zu entwickeln. Als Basis dafür diente das WalkNet [27] (siehe Kapitel 2 und teilweise Kapitel 4). Kapitel 3 stellt ein System zur Planung und Evaluation von Bewegungen vor. Das oben angesprochene Problem, zu bestimmen in welcher Reihenfolge die Gliedmaßen am besten repositioniert werden, wird in Kapitel 4 behandelt. Hier wird ein Planungs-System vorgestellt, das implizit vorausplant und mit reaktiven Systemen zur Bewältigung anderer Aufgaben verknüpft ist.

1 Introduction

Cognitive science can be said to have begun with Descartes's assumption that thinking is a faculty of the soul that is independent from the workings of our bodies [42]. The "Homunculus argument" has since become a classical argument to refute Descartes's and similar hypothesis [57, 58]. This argument asserts that in postulating a Homunculus one is simply shifting the problem of perception – and thinking – to the Homunculus. If someone's perception is actually the perception of his or her Homunculus, then how does the Homunculus perceive? A more general form of this argument is also known as infinite regress.

However, Descartes also claimed in the same work [42] that "what I thought I had seen with my eyes, I actually grasped solely with the faculty of judgment, which is in my mind". This statement is surprisingly close to the hypothesis of the contemporary philosopher Metzinger who proposes that subjective experience reflects the properties of a mental construct rather than sensory inputs [88, 87].

This proposition can be illustrated with the example of the Necker cube (Fig. 1.1). The Necker cube consists of 12 lines, each being parallel to three others. Yet we cannot help but see a wire box depicted in one of two isometric perspectives. The perceived information about the perspective cannot be extracted from the simple diagram, it is a property of the mental construct that the mind uses to represent the diagram. This is further illustrated by the following simple observation: If one looks at the figure long enough, the perspective will usually switch to the other possible perspective.

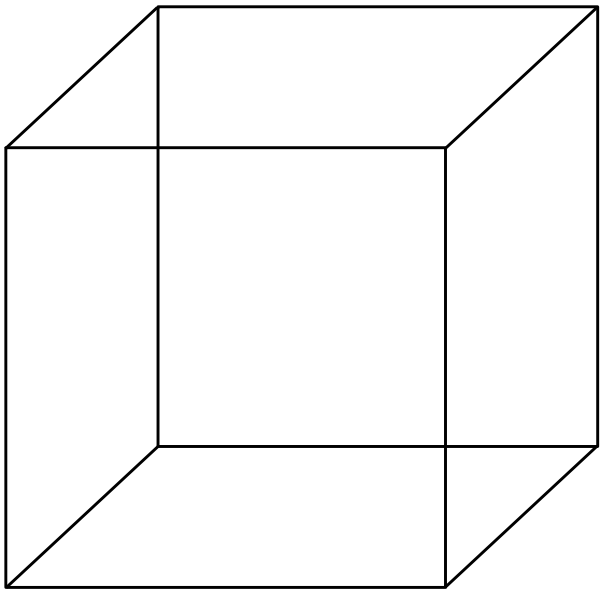

Figure 1.1: The Necker cube

Further evidence for the validity of the hypothesis of constructed reality is for example provided by the existence of phantom limbs as documented by Ramachandran et al. [105]. Many patients with amputated limbs still perceive the void that once was filled with part of their body as a living part of themselves. One of Ramachandran's patients had a particular problem with his amputated arm which he perceived as sticking out at right angles to his body. He could not control the position of the phantom arm and the awkward posture was obstructive in tasks as for example passing through doors. Ramachandran could correct the posture of

the phantom arm with repeated sessions in a setup where – using mirrors – the patient saw his still existing arm in the position of the amputated limb.

Phantom limbs indicate the existence of a very special kind of mental construct: a model of the physical body of the model's owner. Metzinger would indeed argue that there is no owner of the model. According to his view the body model is part of a more comprehensive self model. The entity that experiences subjectivity, i.e. the purported owner of the model is the self model itself. That self model continuously mistakes its construct of reality for reality. This implies that "We are systems that were configured by evolution in such a way that they constantly confuse themselves with the content of their phenomenal self model" [88].

This raises the question why evolution has equipped humans with such elaborate representations and – presumably – not cockroaches. Cruse [30] and Damasio [36] independently argued that the body is predestined as the subject for the – evolutionary – first mental models, because it is always part of the reality as perceived by the modeling mind. That means the perceptions concerning the body are much more constant than perceptions concerning the environment. Cruse further proposed that the complex kinematics of human arms and hands, elephant trunks and octopus arms might have necessitated mental models to exploit the mechanical potential of such actuators [31]. He contrasts such serial complexity with the parallel complexity of e.g. millipedes. In the case of parallel complexity the problem can be solved by parallel feed forward controllers like the Walknet [27] and does not require models.

The biological basis for self models may – at least partly – be found in the sensory motor Homunculus [97, 40, 43]. Several other areas of the brain have also been proposed to play a role in such modelling, for example the cerebellum and the posterior parietal cortex [40, 43], the basal ganglia [117] and more (for an excellent overview see [36]).

Cruse has, together with Steinkühler, proposed a numerical approach that could serve as a body model [122]. The approach is based on redundant calculations of a system of vectors that represent a multi-segment limb and has been expanded to complex kinematic trees by Kindermann [76]. The system is called Mean of Multiple Computation (MMC) network after its main characteristic, the redundant calculations (see sec. 1.2).

Besides serving as a manipulable body, model MMC networks can also account for another observation. Recent research indicates that the faculties of perception and action might be more closely linked than was commonly supposed to be the case (see e.g. [69, 70, 103, 40]). MMC networks are recurrent networks where inputs and outputs are not clearly separated. One can for example provide Cartesian target coordinates which might be derived from visual information and have the MMC network compute target angles for the actuator. Conversely, when thinking – i.e. manipulating the model without using it to control the actual limb angles – it could be used to determine where given limb angles would put the end effector in Cartesian space.

Recent research has indeed provided evidence that thinking might be something similar to the process mentioned in the last use case for MMC networks, i.e. that thinking is neurally similar to acting (e.g. [68, 53, 71]). In order for this to be a plausible hypothesis one would expect to find neural structures that inhibit the execution of actions while

thinking. Support for the existence of such inhibitory structures has been provided by Rizzolatti & Arbib [106] and by Jeannerod [71].

Manipulations of internal body models while the control of actuators is inhibited might be used in complex motoric tasks. Imagine for example a human climbing up a tree. When the tree is large and the branches sparse, the climber will be continually looking for reliable branches to grip or step on. In many cases simply stretching out the according arm or leg will not suffice to reach the branch but shifting the whole body might. The human, being equipped with an elaborate self model, can perform the required body shift mentally and conserve energy in cases where such shifts will not suffice. According to Lanz and McFarland [83] systems endowed with such planning capabilities are called cognitive.

However, the climbing human, indeed any organism performing motoric tasks, will not rely solely on such internal models. Many details of the motions will be guided by tonic muscle reflexes like the patellar reflex [63] and presumably by other more complex reflexes.

Complex reflex systems have been studied comprehensively at the stick insect *Carausius morosus*. One result of this research is the Walknet [27], a hexapod walking controller that contains many concepts derived from observations that were made at stick insects.

The work presented in this thesis attempts the integration of advanced MMC models with reactive systems that were derived from the Walknet. Both the Walknet and the MMC principle had to be enhanced considerably to approach this goal. To lend more credibility to the simulation experiments, a dynamical simulation of the Walker body and its environment was developed.

1.1 Reactive Subsystems

The first attempts to implement control structures for robots were based on a modelling rather than a reactive approach. This strategy was derived from classical AI systems which tried to integrate as much knowledge – in this case about the environment – as possible into elaborate maps. Algorithms were then developed to find optimal paths through the mapped environment (see e.g. [90]). In addition fixed patterns were used to generate rhythmic walking behaviour.

While this approach provided promising results in laboratory settings, it turned out to be too inflexible in more realistic environments – especially in dynamical environments where the elaborate planning algorithms are frequently frustrated by sudden changes of their premises.

Perceiving this approach as a dead end development Brooks proposed the bottom up approach [15, 13]. Instead of starting with elaborate symbolic representations and deriving behaviour from these, the bottom up approach starts with relatively simple reactive systems. With his "Vehicles" [11], Braitenberg can be considered a harbinger of the "new AI" as proposed by Brooks.

In his seminal work, Braitenberg proposed very simple, wheeled robots that nevertheless exhibit interesting behaviour like orientation towards or away from given stimuli (see Fig. 1.2). A distinguishing quality of the Braitenberg vehicles is the direction of information flow inside of the systems: information always flows "feed forward" from the sensors to the motors. The vehicles are thus "reactive", i.e. the system state and motor output is exclusively determined by sensory

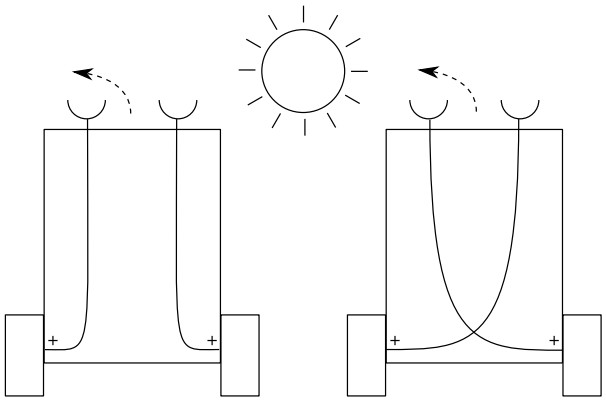

Figure 1.2: Two Braitenberg vehicles with two photo sensors and two motors each. The left vehicle will exhibit negative photo taxis, the right vehicle positive photo taxis.

input. Internal states have no influence on the behaviour of the system.

Braitenberg's vehicles also fulfill another requirement later put forward by Brooks. In contrast to earlier "Cartesian" researchers like Minsky, Brooks does not apprehend intelligence as a separable property that looms beyond the body. Instead he postulated that true intelligence has to be embodied [12, 14, 13].

Obviously though, embodiment is a quantitative rather than a qualitative property. A common desktop computer has sensors (mouse/keyboard ...) and actuators (monitor/speaker ...). Due to its immobile nature it might however only be used to implement some kind of plant intelligence. Yet even Braitenberg's vehicles would be considered to represent a form of "weak embodiment" by some contemporary researchers (see e.g. Sharkey and Ziemke [116]). With two wheels (i.e. two degrees of freedom) and two sensors a Braitenberg vehicle has very limited contingency for complex interaction with the environment – regardless of the amount of "intelligence" linking sensors to motors.

It is interesting to note that Braitenberg, who appears to have been well ahead of his time, studied medicine rather than computer science. Cruse, as a biologist another researcher who had not been exposed excessively to computer science, proposed his coordination rules [21] even before Braitenberg published "Vehicles". Observing stick insects, Cruse derived a few simple, local rules that are assumed to govern the coordination of the six legs of the walking stick. He demonstrated the feasibility of these rules in simulation studies. Compared to Braitenberg's vehicles, stick insects have ample sensory inputs and 18 degrees of freedom in their legs alone (three per leg). The body of the stick insect is also articulated as are the antennae with two degrees of freedom each. This presumably leaves ample room for strong embodiment.

Two decades of research, programming paradigm revolutions, and many simulations after their first implementation, the coordination rules had grown into the Walknet [27] (for a comprehensive study of the Walknet see [73]). The Walknet is a complex system of local rules that control the gait of a six legged walker. It splits the problem of gait generation into three sub problems for each leg:

1. Generating the swing trajectory to move the leg through the air from the posterior extreme position (PEP) to the anterior extreme position (AEP)

2. Generating the stance trajectory to move the leg on the ground from the AEP to the PEP and carry the body of the walker

3. Switching between swing and stance mode, i.e. determining the AEP and PEP

The last problem – determining AEP and PEP – can also be called the coordination problem. It is the only part of the Walknet where information is exchanged between legs. Roggendorf showed that velocity control should also be regarded as a part of the coordination problem (see section 4.5.1). However, the Walknet merely tries to keep the velocity constant and thus does not use velocity control as an aspect of coordination.

The decentralization of the Walknet is manifest in many aspects of its architecture, which is extremely modular. Simple cybernetic elements like neuroids and dynamical temporal filters form the basic elements of the Walknet. Those are combined in small functional groups that solve specific problems (aiming at anterior legs, generating the swing trajectory, various aspects of stance trajectory generation, coordination). The respective groups are combined into the controller for one leg. The leg controllers are finally combined into the Walknet that controls the gait of the walker. On each of these levels of abstraction the exchange of information between the constituting submodules is sparse.

The development goal for the Walknet was not to find a hexapod walking controller that exhibits optimal performance in all possible situations. Rather, the Walknet was implemented to test models that were derived from observations of stick insects. As shown in chapters 4 and 2, the Walknet's performance lags behind alternative solutions, in particular when considering it as a basis for an extension with higher level planning modules:

1. Many subsystems are implemented as artificial neural networks. Consequentially, it is very hard to change certain parameters of the subsystems (e.g. shifting the target position for swinging legs, i.e. the AEP). In most cases a parameter change implies retraining the whole weight matrix of the network.

2. The swing trajectory generation is not completely predictable.

3. The avoidance reflex that should move legs around obstacles, when they are hit during the swing trajectory, is so stereotypical that it frequently leads to lock up situations. In such situations evasion motions lead to hitting the obstacle again in the same position and repeating the same evasion motion continuously.

4. Search mechanisms for finding footholds when no ground is found at the target position after swinging are not implemented in the original version of the Walknet (see [27, 73]) – the swing motion will simply stop shortly after passing the target position. The search mechanisms introduced by Bläsing [6] are specialized for gap crossing behaviour and are tightly integrated with the otherwise insufficient swing trajectory generator (see above).

5. Subsystems of the stance trajectory generation system can counteract each other in certain situations, leading to lock ups. This can for example happen when climbing up an obstacle. The decentralized height control has the effect that front legs that are "trying" to raise the height and thus push against the ground, while middle legs "try" to lower the height and pull at the ground. In adverse configurations as depicted in Fig. 1.3 this can counteract the retraction, i.e. the pushing forward of the

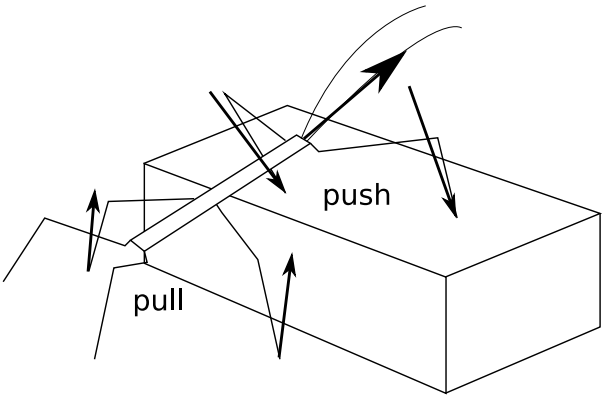

Figure 1.3: Hexapod walker climbing up an obstacle. The Walknet would in such a situation have front legs push against the ground and the middle legs pull. This can counteract the retraction motion and result in a lock up.

body.

6. The stance trajectory modules use positive feedback. The implementation of the positive feedback is technically not mature and can lead to unstable behaviour like building up of oscillations in the controller. As shown in chapter 2, the positive feedback as implemented in the Walknet does not work as intended in dynamics simulations or on real robots (the mechanism was developed and tested with a kinematics simulation).

7. The coordination rules disregard retraction velocity. This is a major reason for their failure to cope with difficult situations (see section 4.5.1).

The unpredictable nature of the Walknet and its incapability to cope with difficult situations render the Walknet unsuitable for extension with higher level planning modules. On the one hand it is essential that planned motions are executed as planned. This however can not be guaranteed when relying on the unpredictability of the Walknet. On the other hand it is desirable to have the lower level reactive controller solve as many problems as possible. Planning becomes important in kinematically complex situations, but the Walknet usually fails before the situation becomes interesting for planning.

Therefore a replacement of the Walknet had to be developed before extensions for planning could be built on top of the reactive controller. The design goal of the replacement controller was a simple, reliable and predictable controller. The architecture of the controller should be as close to the Walknet's architecture as possible, so that parts of the replacement controller could be exchanged for the analogous parts of the Walknet. Thus the replacement controller could also be used for further development of other reactive Walknet components.

The coordination problems of the Walknet are addressed in chapter 4. For this work coordination rules proposed by Porta and Celaya [102, 100] were adapted for operation in the Walknet architecture and extended to meet the design goals of the replacement controller.

The Walknet's problems with stance and swing trajectory generation are addressed in chapter 2. Two antithetic approaches were successfully followed for resolving the leg trajectory generation problems of the Walknet: One approach took an engineers perspective on the Walknet, reduced modularity in favor of some global controllers and used well established algorithms for solving the apparent problems. The other approach increased modularity by abandoning specialized controllers in favor of completely local joint controllers. The local approach also introduced a further variant of the coordination rules

proposed by Porta and Celaya[102, 100].

All of the proposed controllers (see chapters 4 and 2) fulfill the design criteria discussed above and thus constitute a sensible basis for further extension with higher level planning modules.

1.2 Planning Subsystems

As demonstrated by bacteria or stick insects – both presumably not endowed with cognition in the sense mentioned above – reactive controllers can be quite sufficient for many problems. When a reactive system has to make a decision between two or more paths of action, though, it cannot evaluate the consequences in advance. In finding a solution to complex problems feed forward systems are therefore limited to trial and error procedures.

This is fine as long as the cost of performing actions is low. However, when finding a solution to a problem would take too long, or when the mechanical system is in danger of being damaged during the trial and error process, the application of internal models of the manipulator and its environment constitute a significant advantage [23, 31]. The evaluation of different paths of actions using an internal model rather than the actual body might be much faster, is probably energetically cheaper and certainly less dangerous.

In order to be able to plan behaviour on the basis of an internal model, it is essential that the model is manipulable. Most systems classically used for representing information fall in one of two groups.

One group merely stores information for later retrieval. In classical AI systems this was done with hash tables, i.e. (usually long) tables where information could only be retrieved by issuing the correct keyword. More advanced forms of information storage were developed by the "connectionists" (see Rumelhart and McClelland [60, 61]). Connectionists design networks of artificial neurons. In these networks information is stored in a distributed fashion like the visual information in a hologram. Examples are the Perceptrons proposed by Rosenblatt [107] (see also Minsky [89]). These classical heteroassociators can be used to represent information in such a way that it can be recalled even if the input vector differs from the information originally stored in the network, i.e. the network can generalize. Yet more advanced forms of this approach are for example the Hopfield networks [64]. These networks contain recurrent connections. That implies that a system using a Hopfield network is not a strictly reactive system, since information circling in the recurrent loops can in principle also influence the behaviour of the system.

However, none of the systems represented in this first group can be used for planning behaviour since information can only be retrieved from them, but cannot be manipulated. Systems with the ability to manipulate information were the core pursuit of classical AI researchers. This task was commonly approached by representing knowledge as symbols and defining rules to manipulate the information (for a prominent example see Newell [93] and Ernst and Newell [49] for a more recent outlook see Aleksander [1]). This approach was quite successful in some areas. Impressive results include modern chess computers and computer algebra systems.

In other areas – in particular in complex motor tasks and behaviour planning – the approach has only had limited success, which led Brooks to abandon it for the bottom up

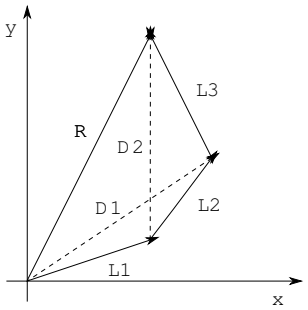

Figure 1.4: An arm consisting of three segments represented by vectors $\vec{L_1}$, $\vec{L_2}$ and $\vec{L_3}$. The position of the end effector is described by vector $\vec{R}$. Furthermore, two additional vectors $\vec{D_1}$ and $\vec{D_2}$, describing the diagonals, are shown.

approach of embodiment as discussed above.

In the spirit of the connectionists, Steinkühler and Cruse [122] proposed an artificial neural network that represents information in a manipulable form – the Mean of Multiple Computations (MMC) network. The first version of the MMC network is based on redundant calculations in a system of vectors. Fig. 1.4 shows a 2D manipulator, that consists of the three segments $\vec{L_1}$, $\vec{L_2}$ and $\vec{L_3}$. The segment $\vec{L_2}$ can for example be computed as $\vec{L_2} = \vec{D_1} - \vec{L_1}$, or as $\vec{L_2} = \vec{D_2} - \vec{L_3}$. Similar – redundant – equations exist for computing each vector shown in Fig. 1.4 from the other vectors. The redundancy is resolved by using the mean value of all computations for one value – hence the name "Mean of multiple Computations".

In fact, the MMC network can be considered to merge connectionist and symbolist approaches. The description of the MMC principle above uses symbols (vectors to represent manipulator segments) and rules (the equations) to process the symbols. However, Steinkühler and Cruse [122] showed that this particular system can easily be translated into the weight matrix of an artificial neural network.

Kindermann and Cruse showed that the MMC principle can also be used to model the complex kinematics of a six legged walker [76]. While these works of Cruse, Steinkühler and Kindermann can be said to have translated the symbolist approach into a connectionist approach, Roggendorf used the connectionist kernel of the MMC principle in a symbolic approach (see chapters 4 and 3).

By forgoing weight matrix based networks the latter approaches simplified the earlier applications of the MMC principle, improved the relaxation behaviour of the networks (chapter 3), and opened new fields of application for the MMC principle (chapter 4). The former work (chapter 3) presents a mature MMC based system that can serve as a manipulable body model. The latter work (chapter 4) presents an MMC based system that can solve the coordination problem. By mixing spacial and temporal aspects of the coordination problem, that system performs implicit planning. Future states are implicit part of the representation. The system relaxes to states that fulfill desired criteria.

2 Stance Trajectory Controllers for Hexapod Walking and Climbing

A system that controls the movement of legs during walking has to deal with two major problems: control of swing and stance motions.

This chapter introduces two stance controllers that combine conceptual simplicity with exceptional walking and climbing performance. One controller is extremely simple, decentralized and biologically plausible. The robustness and performance of this controller are surprisingly good. The other controller uses a basic architecture as derived from insect studies, but follows an engineering approach for solving crucial problems. Both controllers can successfully cross obstacles that are three times the height of the walker hips.

Furthermore, a simple method of enhancing attitude and height control is proposed, that uses information from rostrally neighbouring legs.

All controllers are tested on a dynamics simulation of an insect like hexapod.

2.1 Introduction

Most controllers for six legged walking are developed by mechanical engineers whose main interest is to construct a robot and only then as a second step to develop some controller software to make the main focus of their work walk. This approach considerably limits the ongoing research in controllers for six legged walkers, since current robots have (compared to insects) very limited mechanical capabilities. In particular the torques produced by their actuators are generally not sufficient for climbing.

Therefore insect-like climbing is rarely evaluated by mechanical engineers (for an exception see [8]), and most engineers limit their research on hexapod walking to plane ground (see for example [50]), sometimes with patches of the ground unsuitable for stepping onto (see for example [86, 94]). More recent work has considered low obstacles (see for example [109, 52]), in some cases even approaching walker hip height (see for example [124, 101]). Scorpion – currently the mechanically most advanced multi-legged walker (see [119]) – is even able to cross obstacles, that are somewhat higher (30 cm) than its hip height (26 cm, see [118]) .

Biologists develop controllers for hexapod walking, too. The most notable of these is the Walknet (see [27] and section 2.1.3), which is able to cover a broad range of problems associated to hexapod walking. However, the main focus of the Walknet development was to find models that explain biological data, not to find a controller that performs particularly well. Still, the Walknet's performance, as demonstrated in simulation (see e.g. [73]) and partly in robot (see [99]) studies, put it in league with the best controllers developed specifically for robots.

But even the Walknet is not able to cross obstacles that are much higher than the walker hips. For an overview of related works by mechanical engineers and biologists see [41].

This chapter introduces two novel biologically inspired controllers. Both are reflex/sensory driven (see [27]), i.e. they do not use central pattern generators (see e.g. [5]) and produce free gait walking patterns (see [80]).

Both controllers follow a decentralized architecture which is generally assumed to exist in insects (see [27]). The first type, Akin, consists of six leg controllers. Each explicitely computes the kinematics of the leg movement based on globally given requirements, e.g. body height or foreward velocity. Thus it uses engineering approaches where purely biologically inspired controllers may fail in difficult situations (as will be shown below).

Therefore Akin comprises a perfect decentral controller. It is well suited as a basic system that can be used as a tool to test the properties of newly introduced extensions. The application of such a tool is sensible because it is hard to study the properties of an extension when it is added to a complex and therefore not fully understood system. Of course, Akin can also be used to control a robot.

The other controller, Ejoin, is even more decentralized than biologically inspired controllers. Its architecture is extremely simple and easy to implement – it is probably the simplest controller devised, yet. Each joint of a stance leg is simply controlled by an independent proportional controller with fixed reference inputs. Therefore it might well be realized in biological systems.

The goal of this investigation is to test, to which extent these approaches are suited for the control of a hexapod walker in difficult situations. It is shown in dynamics simulations that both of these controllers perform very well, even when crossing very high obstacles.

As the general goal is to understand insect walking, in the simulations the legs are equipped with adhesive structures that fix stance legs to the ground. In other words, stance legs can develop forces pulling the body to the ground.

2.1.1 Problem Definition

The problem of controlling the gait of a six legged walker can be broken down into three subproblems:

1. Generating swing trajectories, in order to move legs from the liftoff point or posterior extreme position (PEP) to the touch down point or anterior extreme position (AEP) (see Fig. 2.1)

2. Generating stance trajectories, to move legs from AEP to PEP (see Fig. 2.1)

3. Determining the positions of AEPs and PEPs, i.e. deciding when to lift legs and where to put them. This includes the problem of how to coordinate the movement of different legs.

This chapter deals with the second issue, i.e. generating stance trajectories. Controlling leg trajectories comes down to controlling joints. A kinematics controller controls joint angles, a dynamics controller controls joint torques. The controllers described in this chapter are all kinematics controllers, but are tested in a dynamics simulation environment.

Stance trajectory control has to deal with special problems: retracting legs are parts of closed kinematic chains (see [2]).

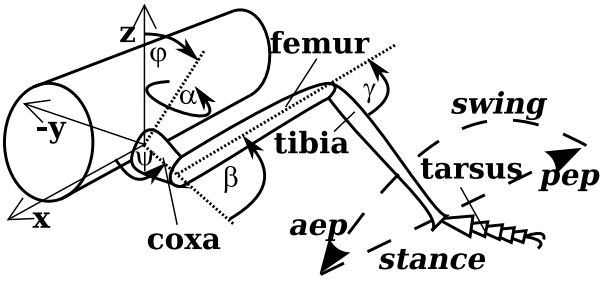

Figure 2.1: Leg geometry, angle and coordinate conventions, segment names, trajectory and extreme point name as used in the text.

That means the legs cannot move independently because they are mechanically coupled through the walker body and the ground. This mechanical constraint enforces all leg motions to be consistent with the motions of all other retracting legs. This results in a reduced number of degrees of freedom. The Kutzbach criterion (see [82]) can be used to calculate the remaining degrees of freedom. A comprehensive discussion of this topic can be found in [67]. The problem of finding consistent solutions in such constraint systems has been studied by several researchers (see e.g. [18, 77, 91, 92]).

In addition to this general requirement of performing consistent leg movements there are further specific constraints. The legs have to carry the body, which furthermore means that a stability criterion has to be fulfilled at all times. Static stability exists if the walker center of gravity is above the convex polygon enclosing the footholds (see [9, 85, 102]). The stability issue is however not crucial for most walking insects because they have adhesive structures on their tarsi (see [108, 79, 4, 3, 125]). In statically unstable situations though, adverse levers can result in excessive torques which increase energy consumption or might even exceed insect muscle capabilities.

2.1.2 Body Degrees of Freedom

The body of the walker has six degrees of freedom in space – three translational and three rotational degrees of freedom. These are coupled to different tasks in stance trajectory control (see Fig. 2.1 for conventions):

x Body motion along the x-axis determines walking speed. Control of that degree of freedom can therefore be regarded as velocity control.

y In order to maintain stability as described above, the walker body may be moved along the y-axis to be centered between the footholds or to maintain the maximal stability margin (see section 2.3.3).

z The walker position on the z-axis determines the body height above ground and might be adapted to ground properties.

roll, pitch Roll motions are rotations along the x-axis, pitch motions along the y-axis. Roll and pitch determine the attitude.

yaw Rotation around the z-axis changes the walking direction. This control variable can be determined by higher order behavioural controllers.

2.1.3 Walknet

The Walknet [27] may be the best established biologically inspired controller for a six-legged walker. It has however only been thoroughly tested in a kinematics simulation. One cursory experiment was done on a dynamics simulation [73]. The Walknet can only reliably cope with obstacles of about walker hip-height [27, 73]. It is however used

as one benchmark for the controllers introduced here. Furthermore, the Walknet's basic architecture is used as a template for the development of the other subsystems (subsystems concerning the generation of swing trajectories) of the walking controller used in the experiments presented in this chapter.

The Walknet breaks the task of controlling a six legged walker into three sub-tasks as mentioned above: Swing trajectory generation, stance trajectory generation and coordination which is defined as switching between the former two sub-tasks, i.e. determining the positions of AEP and PEP. This is illustrated in Fig. 2.2.

In the Walknet four subsystems control the six degrees of freedom of the walker body: The velocity subsystem controls x-translation, the q-subsystem controls y-translation, the height subsystem controls z-translation. The height subsystem only uses information that is local to the corresponding leg. The resulting equilibrium leads to a sensible body attitude. Thus the height subsystem can also be regarded as controlling roll and pitch. Yaw is controlled by the direction subsystem. α and γ joints (see Fig. 2.1) are controlled based on their angular velocity. The α joints are mostly responsible for protraction and retraction of the legs, the γ jonts are mostly responsible for lengthening the leg. The $\dot{\alpha}_{min}$ system enforces the $\dot{\alpha}$ value to stay above a given threshold. Without applying the $\dot{\alpha}_{min}$ subsystem the whole system is extremely unstable.

The Walknet's stance trajectory controller does not explicitly compute kinematics. It relies on the approximation that, in the leg coordinate system, x is about proportional to α and γ, y is about proportional to γ, and z is about proportional to β. The error of this approximation is mitigated by the positive feedback loop, which implements active

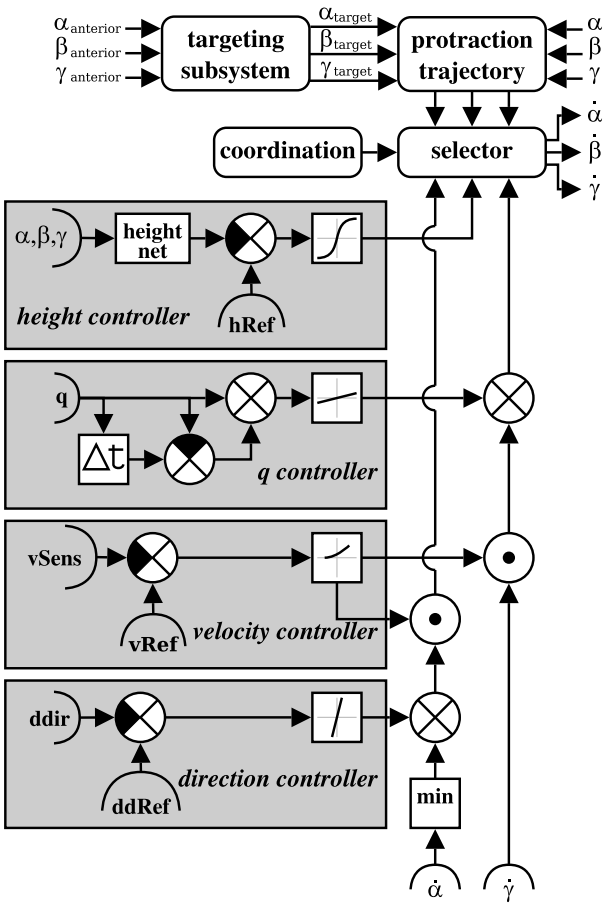

Figure 2.2: The Walknet controller for one leg. Targeting and swing trajectory subsystems generate the swing trajectory. Selector and coordination switch between swing and stance. All other elements are subsystems of the stance trajectory controller. Note that the sensed angular changes $\dot{\alpha}$ and $\dot{\gamma}$ are passed unchanged into selector if the error signals from the subsystems of the stance trajectory controller are zero. This is the positive feedback loop.

compliance (see [73]). Since this is insufficient in some walking situations, and as subsystems work against each other the walker can get stuck (see [73]): depending on the slope of the substrate, the different height controllers may work against each other. In particular, the height controller can work against the velocity controller when climbing up obstacles.

Targeting subsystem

The target positions for the front legs are given by fixed reference positions or reference angles. The other legs aim at points behind the current position of the anterior leg (see [20, 24, 39] for similar mechanisms in *Carausius*). All target positions including those for the front legs are adjusted to aim at the correct height (the information is taken from global knowledge about the environment simulation, but could potentially be retrieved from interpolation, or – in case of the front legs – from the antennae which are not simulated in the experiments presented here). That means all target points are always points on the ground.

2.2 Novel Models

Two controllers are introduced which are termed the Akin ("*a*nalytical *kin*ematics") controller and the Ejoin ("*e*ach *j*oint *o*n *i*ts ow*n*") controller.

Definitions

- Conventions for the global coordinate system: positive x values point to the anterior, positive y point to the left, and positive z point upwards.
- The leg coordinate system is rooted in the coxa (α joint) of the legs. x and z direction are equal to global coordinate system, y always increases toward the outside (i.e. left on the left body side and right on the right body side)
- A specific integrator is used, which is termed the auto reset integrator. This is an integrator that is automatically emptied when the sign of its input changes. When this special integrator is used in a negative feedback I-controller, the controller can have high gains without tending toward oscillations, combining advantages of proportional and integral controllers – i.e. it reacts as fast as a proportional controller, but corrects the error almost as completely as an integral controller. This graceful behaviour can however only be expected, when the controller controles the change rate of the actual value – as is the case with the controllers presented below – rather than its absolute value. If absolute values were controlled, the auto reset integrator would lead to major jumps of the controlled value, possibly resulting in instable behaviour of the system.

2.2.1 Akin Controller

The Walknet is an already complex system, able to control walking in disturbed situations. When more difficult tasks had to be solved (as are investigated in biological experiments like climbing over large gaps, negotiating sharp turns, climbing over narrow substrate), the controller had to be extended. To be able to test the properties of such extensions, a basic controller with predictable behaviour is required. Therefore, the core of the Akin controller is an

*a*nalytical *kin*ematics. For each leg this core has nine inputs – the six degrees of freedom of the body as explained in section 2.1.2 and the three current leg angles. The desired motions in these degrees of freedom are translated into the three new leg angles for each leg, the changes of the leg angles being the output of the subsystem. This means that information about body motion and rotation is translated into new local leg coordinates and that these are transformed into angles using the inverse kinematics of each leg.

The six body degrees of freedom are controlled by separate global subsystems.

Velocity control Velocity is determined by the coordination subsystem (see below, section 2.2.1.2). If all retracting legs are in the middle or in the anterior range of their work-spaces, the default velocity $v_{default}$ is used as the reference input for the velocity controller. If retracting legs do however approach the posterior end of their workspace, the reference velocity is lowered, eventually reaching zero.

The reference velocity is fed into a negative feedback integral controller to determine Δx: $\Delta x = \int k_x \left(v_{ref} - v_t\right) \mathrm{dt}$, with reference velocity v_{ref} determined by the coordination (see sec. 2.2.1.2) and current velocity v measured by an odometric subsystem (see [73]). k_x is the constant controller gain.

Controller Prescale Before entering the kinematics core, all six values are prescaled by a velocity influence so that fast movements are inhibited when the overall velocity is rather low. The control value Δx is used to determine the prescale p for all control values of the stance trajectory controller:

$$p = \begin{cases} \left|\frac{v_{ref}}{v_{default}}\right| & \text{if} v_{ref} < v_{default} \\ 1 & \text{else} \end{cases}$$

The velocity that is used as input for the leg kinematics is scaled by p as are all other control values (Δy, Δz, $\Delta roll$, $\Delta pitch$, Δyaw): $\Delta x_{scaled} = p\Delta x$. This method is responsible for adjusting the velocity of motions of all stance legs.

Q-Controller The q-controller centers the body between the footholds. The mean distance of the footholds from their corresponding coxae is calculated separately for the left and right body side. The mean left distance is then subtracted from the mean right distance. The resulting Δy_{pre} is fed through a derivative controller: $\Delta y = \frac{\mathrm{d}\Delta y_{pre} k_y}{\mathrm{d}t}$ with controller gain k_y.

Height Controller The height controller calculates the mean height (z coordinate in leg coordinate system) of all retracting legs and feeds this value through a negative feedback integral controller with auto reset (see definitions above):

$$\Delta z = \int_{\mathrm{auto-reset}} k_z \left(h_{t\,mean} - h_{ref}\right) \mathrm{dt} \quad (2.1)$$

.

Attitude is determined by roll and pitch. The first approach was to project the foothold points into two rotational planes (i.e. the x/z-plane for pitch and the y/z-plane for roll) and calculate an orthogonal regression line for those points. The control values were then the angles between the actual attitude and the regression lines, scaled by a gain factor and the confidence value computed from the regression. Preliminary tests showed that this approach only works for plane ground or moderately high obstacles. It occasionally fails in difficult situations. However, it turned out that using

height values yields much better results: The roll control value is the difference between the mean height of the left retracting legs and the mean height of the right retracting legs.

The pitch control value is derived from comparing mean heights of front, middle and hind retracting legs:

$$\Delta pitch = \sum_{i=f,m;j=m,h;i\neq j} \begin{cases} i-j & \text{if} \exists i \wedge \exists j \\ 0 & \text{else} \end{cases}$$

, with f, m, h = front mean height, middle mean height and hind mean height. The if statement tests if mean values could be calculated, i.e. if at least one leg of each leg pair i, j is retracting. That means that in typical walking situations all leg pairs are compared against each other. Note that this approach is similar to the one described above with linear instead of orthogonal regression.

Both roll and pitch control values are scaled by gain factors and fed through an auto reset integrator.

Yaw Controller The derivative of the measured yaw value is scaled by a gain and integrated to determine the control value. This constitutes a negative feedback integral controller based on Δyaw, that enforces straight walking paths. For curve walking a bias is added to the derivative before scaling and integration: $\Delta yaw = \int k_{yaw} \left(\frac{\mathrm{d}yaw}{\mathrm{d}t} + bias\right) \mathrm{d}t$. [73] used a proportional controller rather than an integral controller. Integral control does however work well with the kinematics core of the model and it ensures long term straight walking, which can simplify the experiment setup significantly.

2.2.1.1 Anticipation Extensions

In this section two simple extensions are introduced, one for the height-, the other for the attitude controller. Both extensions use information from the targeting subsystem (see Fig. 2.2 and subsection 2.1.3). Both controllers use height as their control values. In the attitude- and height controllers explained above the current height of the retracting leg is used. In the extended versions also the height values of the target points of legs are used that are yet swinging. This means that information from the rostrally neighbouring leg is used as well.

The effect of these extensions is an anticipative lifting and rolling/pitching of the body in front of obstacles. This behaviour is in accordance with biological data (see [47, 96, 115]).

The anticipative control was combined with an adjustment of the height controller to the ground properties: in addition to calculating the mean height $h_{t\,mean}$ of all stance legs as above, the variance of that mean value was also calculated. The square root of the variance was scaled with gain factor k_g. The result, limited to a maximal value $b_{g\,max}$, is the ground bias b_g:

$$b_g = \max\left(k_g \sqrt{\sum_{i=0}^{i<nstance} (h_{t\,mean} - h_i)^2}, b_{g\,max}\right)$$

b_g is added to the reference height $height_{ref}$ as used in equation 2.1. The effect of this procedure is that the walker body will maintain higher ground clearance on rough ground.

2.2.1.2 ThreshPorta Coordination

For testing the above stance trajectory controller (and the WalkNet), a complete walking controller including a coordination subsystem has to be provided. A modified version (see chapter 4) of a model proposed by Porta and Celaya [100] has been used for

solving the leg coordination problem. The ThreshPorta model has been proven to work well in experiments (see chapter 4) similar to the ones presented here.

In order to describe the model proposed by Porta & Celaya [100], the term of neighbouring legs has to be defined. According to the Porta & Celaya model, all adjacent legs are neighbouring with the exception of the two middle legs, which are *not* considered neighbours. Thus each leg has exactly two neighbours (the middle left and front right legs are the neighbours of the front left leg; the front right and hind right leg are the neighbours of the middle right leg). According to the Porta & Celaya model a leg is to be lifted, if it has a higher lifting priority than both neighbouring legs.

Lifting priority is defined in the following way: Protracting legs have the highest lifting priority (this implicitly means that neighbouring legs cannot protract simultaneously – the prime condition which Porta & Celaya wanted to fulfill). For retracting legs, the lifting priority is negatively proportional to the leg's distance from its physical PEP (the closer a leg is to its physical PEP, the higher its lifting priority). The physical PEP is determined by leg geometry (leg segment lengths). It is the hindmost point a leg can reach in a normal walking position.

In the original version of the model, the default stance velocity $v_{default}$ cannot be controlled: it has to be close to the swing velocity. If $v_{default}$ is lowered, coordination breaks down. The following extension has been introduced to solve this problem (see chapter 4).

The lifting rules are *only* applied, if the distance D of a given leg from its physical PEP is smaller than a preset threshold T. T was set to 5.0 mm in our experiments (compare to step lengths, being in the order of 20 mm).

These parameters are used for calculating the actual reference stance velocity v_{ref}. Each retracting leg proposes a "desired" global stance velocity accorded by $v_{ref} = \frac{D}{T} v_{default}$ if $D < T$, and $v_r = v_{default}$ otherwise. The leg with the smallest proposal determines the actual v_{ref}.

2.2.2 Ejoin Controller

In strong contrast to the mathematically exact yet decentral Akin controller, a controller with an extremely simple architecture is studied in this section. The Ejoin controller controls *E*ach *J*oint *O*n *I*ts ow*N*. Every joint angle θ has a simple negative feedback proportional controller with a fixed reference value θ_{ref} and a gain factor k_θ:

$$\Delta\theta = k_\theta \left(\theta_{ref} - \theta\right) \tag{2.2}$$

. Fig. 2.3 shows a diagram of the hexapod controller; it is divided into six partitions (boxes marked by thin lines), one for each leg; each partition contains three negative feedback controllers according to the equation above – one for each joint.

2.2.2.1 Anticipation Extension

The Ejoin controller is easily extensible, because it is very robust against additional influences and parameter changes. As an example an extension is introduced here, that behaves similarly to the anticipative attitude control introduced in section 2.2.1.1.

When approaching an obstacle, values describing the height of this obstacle are used to influence the β controllers of the front legs and of the hind legs in order to adjust the body pitch to the obstacle situation in front of the walker. The actual height of the front leg's target points are taken from

world knowledge in the simulation, but could be retrieved from antennae which were not simulated in the experiments presented here, see Fig. 2.2 and subsection 2.1.3.

The actual target height values of the target positions are translated into the front leg's coordinate system. By calculating the inverse kinematics of those legs, the according actual target β angles, β'_L and β'_R, are determined. The sum of these angles, $\beta' = \beta'_L + \beta'_R$, is used to influence the height of the front and hind legs. β' is compared to a fixed default value β'_0 that corresponds to an obstacle of height zero. The difference $\beta'_0 - \beta'$ is multiplied by a constant factor, $k_{\beta'}$, fed through an autoauto reset integrator (see sec. 2.2) and then added to the error of the front and hind leg's negative feedback controller. The equation for the front leg's β angles (see 2.2) is extended to

$$\Delta\beta = k_\beta \left(\beta_{ref} - \beta\right) + \int_{\text{auto reset}} k_{\beta'} \left(\beta'_0 - \beta'\right) \mathrm{dt}$$

, the equation for the hind legs is

$$\Delta\beta = k_\beta \left(\beta_{ref} - \beta\right) - \int_{\text{auto reset}} k_{\beta'} \left(\beta'_0 - \beta'\right) \mathrm{dt}$$

($k_{\beta'}$ is the constant gain of the proportional controller). See Fig. 2.3, circuit above and left of the walker.

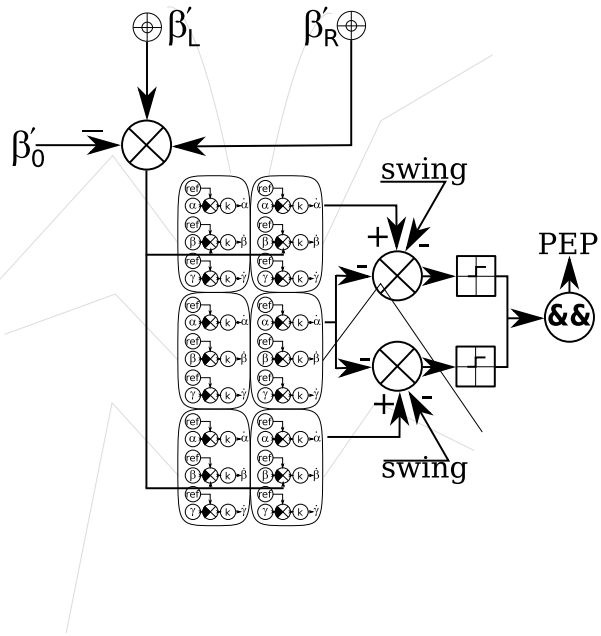

Figure 2.3: The complete Ejoin controller for all legs (hexapod body) with attitude anticipation (above and left of body) and coordination (right of body, coordination only shown for middle right leg).

2.2.2.2 Ejoin Coordination

A simple coordination mechanism that is modeled after the Porta coordination scheme was developed for the Ejoin controller. Neighbourhood in the Ejoin coordination is identical to neighbourhood in the Porta coordination. As in the ThreshPorta coordination, legs are lifted, when their lifting priority is higher than the lifting priority of their neighbouring legs. Swinging legs have highest priority – as in the Porta controller. The

lifting priority of stance legs is not determined by considering leg positions and work spaces as in the ThreshPorta variant, but is given by $\frac{1}{\Delta\alpha}$ (with the controller error $\Delta\alpha$ as calculated from eq. 2.2). This means the leg with the more posterior position has higher priority. Since the Ejoin controller cannot explicitly control stance velocity but does this implicitly, no stance velocity is calculated and no threshold is required.

Using the above rules, it is possible to design a simple circuit (see Fig. 2.3, right side of the body) to determine the PEP of a leg: The swing signal is one, if a leg is swinging and zero otherwise. $\Delta\alpha$ is subtracted from the $\Delta\alpha$ of the neighbouring leg and the swing signal of the neighbour is subtracted from the result. The resulting value will be positive only if $\Delta\alpha$ of the current leg is lower than $\Delta\alpha$ of the neighbouring leg and if the neighbouring leg is not swinging. This result is finally mapped to zero or one by a relay characteristic.

This is done for both neighbouring legs. The PEP is reached if both results are one.

2.3 Experimental Testbed

The two controllers and the Walknet were tested in a dynamics simulation of a six legged walker and its environment.

2.3.1 Swing Trajectory Controller

For better comparability, the same, simple swing trajectory controller was used for all leg controllers investigated. It moves the leg forward (in x direction) with a constant velocity of one millimeter per iteration cycle until it reaches the target position. The y-component of the motion was controlled by a negative feedback proportional controller, the height component was moved through a sawtooth trajectory between liftoff and touchdown.

If the leg hit an obstacle while still not close to the target position, an evasive motion was executed. The motion started in the opposite direction as that given by the torque vector in each leg joint when hitting an obstacle. The leg was then moved up a bit and finally continued its trajectory (see [111] for a similar reaction in the stick insect *Carausius*).

If ground contact was not registered at the position determined by the targeting system, the leg was moved straight down with a constant velocity. If still no ground contact occurred after a constant number of iterations, a search movement was generated by combining low (covering the whole workspace) and high frequency sine waves. Thus the whole workspace was searched with cirular patterns. Similar behaviour has been observed in *Locusta* [95] and *Carausius* [44, 7].

2.3.2 Dynamics Simulation

The dynamics simulation is based on the rigid body dynamics library ODE (Open Dynamics Engine, http://ode.org). The walker body is simulated as two bodies, one representing head/thorax, the other representing the abdomen. Both were linked by a passive hinge joint with elastic properties (see below).

Six legs are attached to the thorax, each leg consisting of three segments (coxa, femur and tibia) and linked to the body and one another through hinge joints (α, β and γ). The joints receive their reference values from the controllers.

Since the Walknet, the Ejoin controller and in some cases also the Akin do not

Segment	Length (mm)	Weight (mg)
Front		
Coxa	1.61	0.43
Femur	17.85	10.30
Tibia	17.10	3.30
Middle		
Coxa	1.57	1.00
Femur	13.47	8.05
Tibia	13.20	1.70
Hind		
Coxa	1.39	1.00
Femur	15.51	8.60
Tibia	16.51	2.70
Body		365.40

Table 2.1: Walker- length and weight parameters of dynamic simulation, see [48]

produce kinematically consistent configurations, the joints have to yield. Therefore the joints were implemented to be elastic. When pulled out of the reference position they produce a force proportional to the deviation. The joint behaviour is determined by spring constant and damping.

The spring constants were set as low as possible, but at least high enough to carry the walker weight even in adverse walking situations. The damping constants were then set as high as possible. The spring constant and damping was identical for all joints except for the thorax abdomen joint.

The segment lengths and masses were taken from [48], which partly relies on [19] (see tab. 2.1). Fig. 2.6 illustrates the proportions of the walker parts. It is an exact representation of the dynamics simulation.

The environment was simulated as a ground plane on which obstacle boxes could be placed.

Collisions were calculated between all bodies, including the ground plane and obstacle boxes. The contact points were also simulated as springs. Spring constants for contacts between walker parts and environment were three orders of magnitude higher than the joint spring constants. Spring constants for contacts between walker parts were three orders of magnitude lower than joint spring constants – this improves simulation stability because it prevents force buildup when walker parts collide. All contacts were frictionless.

When the tarsus (the lower part of the tibia segment) had ground contact and the controller set the according leg to stance mode, a frictionless ball joint was created between tarsus and environment. This joint was removed when the controller set the leg into swing mode. This simulated the adhesive properties of the tarsi of many insects (see [108, 79, 4, 3, 125]). Mechanical analogues for robots are currently investigated, prototypes have been shown to facilitate walking on vertical surfaces (see [104, 72]).

The simulation step was 0.05 s, i.e. one second took 20 iterations.

The walker was initialized with a leg angle configuration that resulted in all legs having ground contact and allowing to start walking without coordination problems. The walker was dropped from a height of maximum leg length, i.e. 36.56 mm. For the first 10 iterations, fixed angles – the start configuration – were set for the simulation, then the controllers determined the leg angles.

2.3.3 Stability

For evaluating the performance of the controllers the "stability margin" is introduced as a measure. The stability margin is based on the stability polygon (see. [9, 85, 102]) and a special characteristic.

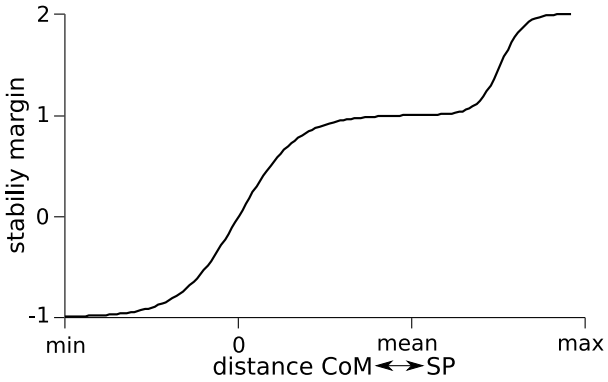

Figure 2.4: The stability characteristic for three or more ground contact points (CoM = centre of mass, SP = support polygon); distance CoM<->SP is the smallest distance between the center of mass to any boundary of the support polygon, see text for details.

The stability polygon is the convex hull around all ground contact points. A ground contact point is given by a leg in stance mode touching the ground. The centre of mass is assumed to lie between the middle leg coxae, projected into the ground plane (the plane that is orthogonal to the gravity vector). The distance of the centre of mass to the closest point of the stability polygon is measured. If the centre of mass lies inside the polygon, the distance value is set to be positive, otherwise it is set to be negative. If the centre of mass lies exactly on a corner or edge of the stability polygon, the distance is zero. [85] defines a similar concept, the longitudinal stability margin. For the longitudinal stability margin the distance of the centre of mass to the boundaries of the support polygon is only calculated in the direction of motion of the centre of mass. The stability margin used here gives a better measure since it calculates the smallest distance of the centre of mass to any boundary of the support polygon.

This distance value is mapped by a function that enhances resolution in critical situations and lowers resolution in "uninteresting" situations (Fig. 2.4). If all legs are fully extended and have ground contact, the distance reaches a maximum, which is mapped to a stability margin of 2 by the stability characteristic. The mean distance during walking on a plane is mapped to show a stability margin of 1. The minimum (i.e. the greatest negative) distance possible with three ground contact points is mapped to -1.

The following extreme cases are not depicted in Fig. 2.4: When only two legs have ground contact, the stability margin is determined by a linear function that maps the distance of the centre of mass to the line connecting the two ground contact points, to lie between -1 (if the centre of mass lies exactly between the two ground contact points) and -2. When only one leg has ground contact, the distance of the ground contact point to the centre of mass is linearly mapped to lie between -2 and -3. If no legs have ground contact, the stability margin is set to -4.

The stability margin is recorded along with the walker's body position (x/y position of point between the bases of the hind coxae, see Fig. 2.1) for each simulation iteration of each run.

2.3.4 Torque measurement

In one set of experiments the joint torques produced by the different controllers were measured. Phase changes of single legs produce relatively high torques in all legs (e.g. when protracting legs hit the ground and start retracting). In order to make the torque measurements independent from the frequency of phase changes, torque measurements were only updated when no phase changes occurred during the last five iter-

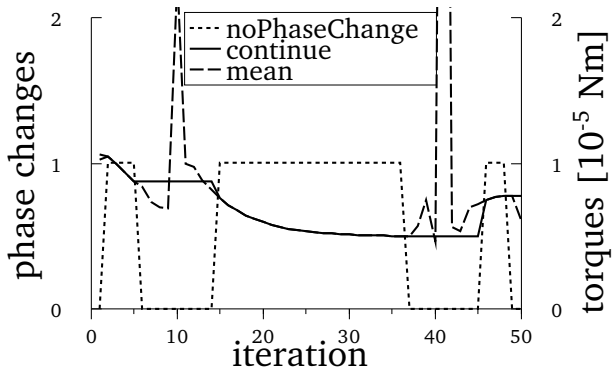

Figure 2.5: Algorithm for determining torques (in this case Akin controller). *noPhaseChange* is 1 if there was no phase change in any leg during the last five iterations, *mean* is the mean $|\beta|$ torque of all retracting legs, *continue* is the same as *mean*, but is only updated if *noPhaseChange*.

ation. This is visualized in Fig. 2.5, where *continue* shows the torque value that is actually used.

For better comparability of standing, tetrapod, and tripod walking, the mean absolute torques τ were then normed to six ground contacts, taking the actual number of legs with ground contacts gc into account: $\bar{\tau} = \frac{6\tau}{gc}$.

2.3.5 Parameter Optimization

The parameters of the Akin controller (as listed in sec. 2.2.1) and the Ejoin controller (gains and reference values, see sec. 2.2.2) were optimized by evolutionary programming.

For each controller a sensible start configuration was chosen. Each generation consisted of 100 individuals. The ten fittest were propagated to the next generation. Nine mutated copies were made of each of the ten fittest individuals to fill up the population to 100 again. In the mutated individuals each parameter had a 0.1 chance of being scaled by 0.6 to 1.4.

The fitness function $f(x)$ selected for high speed and low torques (see sec. 2.3.4). The torque and velocity values were long temporal mean values. v is velocity, t_i is the mean joint torque over all retracting legs filtered as described in section 2.3.4:

$$f(x) = \frac{v}{\sum_{i=\alpha,\beta,\gamma} t_i}$$

.

The environment in which the evolution proceeded contained one obstacle of height 20 mm, which the walker had to traverse. Evolution proceeded until no parameters changed in the best controller for ten generations.

2.3.6 Experiments

The following controllers were tested:

1. No controller (fixed angles)
2. Walknet
3. Ejoin without attitude anticipation
4. Ejoin with attitude anticipation
5. Ejoin B without attitude anticipation and reference angles adjusted for standing
6. Akin with neither anticipation nor ground clearance adjustment
7. Akin with attitude anticipation
8. Akin with height anticipation and ground clearance adjustment
9. Akin with attitude and height anticipation and ground clearance adjustment

Three tasks have been investigated: standing still on plane ground (torque measurements at models 1, 2, 3, 5, and 9), walking on plane ground (torque measurements at models 2, 3, and 9) and climbing over an obstacle (all models except 1, 2, and 5). Models 1 and 5 could not be used for walking experiments, because they are not designed to walk. The Walknet (#2) was not used in the obstacle crossing task, because it was shown in the torque experiments that it does not walk properly in a dynamics simulation. Those Ejoin and Akin controllers were selected for the torque experiments that performed best in the obstacle crossing task.

The following sections will refer to the models by using the numbers of the list presented above accordingly, e.g. (#1) for the uncontrolled system.

2.3.6.1 Standing on Plane Ground

The task was to stand on a plane for 5000 iterations (250 seconds). A starting posture different to that used for the walking experiments was chosen for this experiment: the posture was left/right symmetrical. To balance this posture, the velocity subsystem of the Akin controller was turned off. That partly disabled controller then controlled the walker while standing. The posture was balanced to neither shift forward nor backward. Resulting leg positions along the x-axis in leg coordinate system were: front 16.65 mm, middle 0 mm, hind -10 mm; y position was left/right 15/-15 mm; height was 6.5 mm.

Because of technical reasons various modifications had to be introduced to the controllers to make them stand still:

Walknet (#2) The Walknet contains the α-min module that assures that $\Delta\alpha$ is always at least 1° as it enters the positive feedback loop (see Fig. 2.2 and [73, 27]). This module was deactivated and the reference velocity was set to zero. Since the Walknet started showing erratic behaviour due to numeric effects when standing for more than 1000 iterations the test was limited to 1000 iterations in this case.

Ejoin (#3) Coordination was deactivated so that legs would remain glued to the ground.

Ejoin B (#5) Reference angles were adjusted to reflect the standing position (as defined above); this was done to reach smaller torques; coordination was deactivated.

Akin (#9) The default velocity was left unchanged, because otherwise the velocity prescale (see sec. 2.2.1) would deactivate all controllers. x-position was determined by a separate negative feedback controller with gain 1 and reference velocity 0.

The normed long temporal mean torque was recorded as described in section 2.3.4.

2.3.6.2 Walking on Plane Ground

The task was to walk on a horizontal plane for 5000 iterations in a line as straight as the controller would permit. During walking the normed long temporal mean torque was determined as described in section 2.3.4.

2.3.6.3 Climbing over an Obstacle

The task for all controllers was to cross an obstacle of varying height. It consists of an upward step and a downward step that are separated by a distance of 100 mm. For each obstacle height (see below) 150 trials were

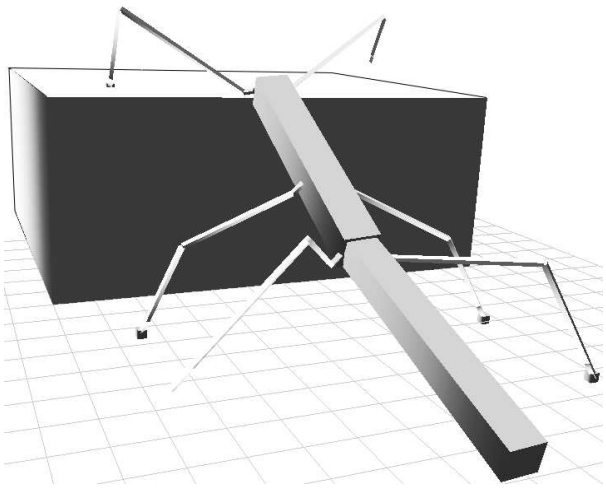

Figure 2.6: Screenshot of the dynamic simulation. The walker is ascending an obstacle of height 25. Small boxes between tarsi and ground represent contact points.

run. For each trial, the x-position of the obstacle is shifted by 0.1 mm. This has the effect that the legs make first contact with the obstacle at different points in their duty cycle, introducing randomness into the experiments.

26 obstacle heights (ranging from 0 - 25) were tested. Thus 3900 experiments were done with each of the six controllers. For an illustration of the highest obstacles see Fig. 2.6.

During the experiments the position of the walker's centre of gravity, walker velocity, and stability (see sec. 2.3.3) were recorded.

In the collected data each run was split into three parts according to the walker position. The first part was the area around the first obstacle cliff, where the walker was ascending, the second part was the area where the walker was descending and the third part was an area behind the obstacle, where the walker was walking on plain ground after traversing the obstacle. All areas had the same size. The data was sorted by obstacle height.

The mean velocity was calculated as the arithmetic mean of all data points from all x-shifted obstacles for each task (ascending, descending walking on a plane). To calculate the probability p_{unStab} of the walker to become unstable while performing a task, all runs for a given obstacle height were pooled. If the walker became unstable anytime in one run while performing a given task, p was 1 for that task in that run. If the walker never became unstable while performing the task in one run, p was 0 for that task in that run. The probability of becoming unstable, while performing a given task at an obstacle height, is the mean of all according p.

The resulting data was smoothed for the plots (Fig. 2.8): each data point in the plots is the arithmetic mean of seven surrounding points, including the according point itself. Three data points containing the value for height = 0 were prepended to the test data and three values containing the value for height = 25 were appended, to avoid the cropping that is necessary for the smoothing.

2.4 Results

2.4.1 Torques

2.4.1.1 Standing

The absolute torques produced by the uncontrolled system with fixed reference angles (#1 S) can serve as a benchmark for all other controllers. This benchmark was beaten by all controllers (see Fig. 2.7, top) except the Ejoin controller with standard reference angles (#3 S, see Fig. 2.7, bottom, right). The latter produced torques that are two orders of magnitude higher than the torques produced by any other controller while standing. The torques were even higher than

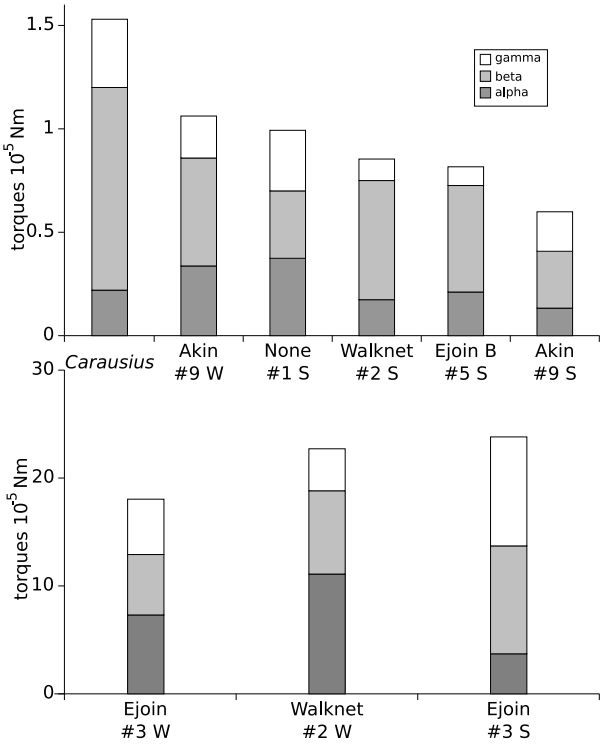

Figure 2.7: Joint-torques produced by various controllers when standing (marked S) and walking (marked W). Numbers for stick insect *Carausius* taken from [19].

those produced by any controller while walking.

The Akin (#9 S) controller produced the lowest torques, followed by the Ejoin B (#5 S) controller with reference angles adjusted for standing and by the Walknet (#2 S). In all controllers the torques are close to the torques of the uncontrolled system. While the torques in the uncontrolled system are about evenly distributed over the three joint types, all controlled systems – including Ejoin B (#5 S), but not Ejoin (#3 S) – put more load on the beta joints that carry the walker weight while relieving the gamma and alpha joints.

The Ejoin controller with standard reference angles (#3 S) produces the highest torques in β and γ joints indicating tension in these joints due to an inconsistent reference posture. This controller immediately shifted the standing walker far to the anterior direction into a crouching position with lowered head and elevated abdomen.

The Walknet started shifting the walker body backwards after about 1000 iterations. Later it would start oscillating.

2.4.1.2 Walking

The Akin controller (#9 W) produced torques (see Fig. 2.7) that were only slightly higher than the torques produced while standing, the Ejoin controller (#3 W) and the Walknet (#2 W) both produced much higher torques, the Ejoin controller (#3 W) performing slightly better.

The Walknet only achieved an average velocity of 0.3 mm per iteration – half of the default velocity. 0.3 mm per iteration is the value to be expected from the sole action of the $\Delta\alpha$min module. When this module was turned off, the walker would indeed not move at all. Or rather, it would start shifting erratically and oscillating.

2.4.2 Crossing an Obstacle

The results are shown in Fig. 2.8. The data shown in that figure was statistically analyzed. All data as shown was analyzed data point by data point. The probability of becoming unstable was analyzed with the χ^2 test, the velocity data was analyzed with the t-test. In the stability plots there is a one percent error probability where the differences between lines are about 0.06 for low values, 0.11 for mid range value and 0.13 for high values. In the velocity plots there is a one percent error probability where the differences between lines are about 0.02.

The general result is that all controllers can cross high obstacles with reasonable velocity and a low probability of becoming unstable. The Ejoin controllers perform better in most tasks than the Akin controllers. Attitude anticipation is advantegous as implemented in the Akin controller (#7, #9) and ambiguous in the Ejoin controller (#4). Anticipative height control yielded ambiguous results (#8, #9).

The Ejoin controller performs better without the attitude anticipation (#3) except for walking on plane ground after crossing the obstacle – there the attitude anticipation (#4) puts it in league with the Akin variants, while it performs much worse without the extension.

The probability of becoming unstable while ascending or descending an obstacle is lowest with the Ejoin controller without attitude anticipation (#3). Controller #3 performs the descending task on average a bit faster than the other controllers, but is slower when ascending or walking on plane ground (in these tasks it is slower still, when attitude anticipation (#4) is used).

The various Akin controllers cannot be discriminated by their performance on plane ground. However, both extensions (#7, #8, #9) improve the performance (mean velocity and stability) when ascending the obstacle. Attitude anticipation (#7, #9) also improves the performance when descending. The attitude anticipation (#7, #9) is apparently more important than the anticipative height adjustment (#8, #9), yet both improve the performance especially for ascending the obstacle. When descending the obstacle, the anticipative height adjustment (#8, #9) might even be disadvantageous.

The anticipative attitude controller in the Ejoin controller (#4) performs better on very high obstacles than on moderately high obstacles.

2.5 Discussion

Two novel controller designs for six legged walking were presented in this chapter. Both designs proved very performant at the difficult task of crossing very high obstacles, a task that was to our knowledge not yet solved by other hexapod walking controllers. The basic modular layout of the Akin controller is analogue to the layout of the Walknet. Therefore, parts of the Akin controller can be exploited as drop ins, when other parts of biological models are to be tested, since no *reliable* biologically inspired subsystems exist for practically any part of the walking problem.

The Ejoin controller may be the simplest controller yet, that reliably solves the six legged walking problem. It is therefore well possible that such controllers form the basis of biological solutions. It however differs from controllers assumed to exist in stick insects in two aspects.

First, height control was found to contain nonlinear characteristics [28], which are not included in the Ejoin controller but could easily be implemented. Second, stick insects, when walking, are assumed to use a mixture of positive and negative feedback in contrast to the simple negative feedback solution used here.

Implementing both aspects might lead to smaller torque values, and thus explain, why torque values produced by the Ejoin controller are higher than those found in free walking stick insects (see below and tab. 2.2). The Ejoin controller lends itself particularly well to such modifications because of its simplicity and robustness.

The general usefulness of anticipative

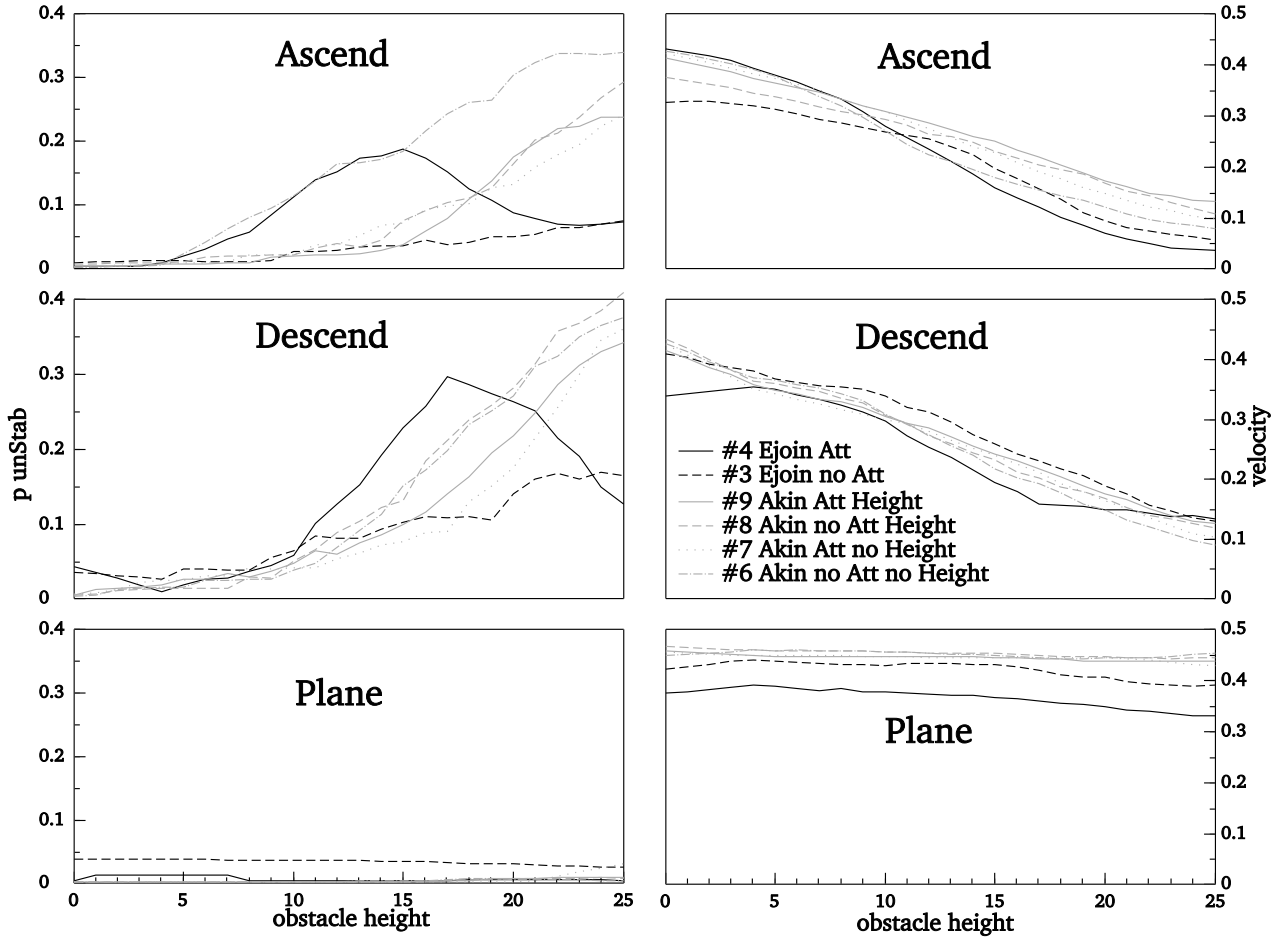

Figure 2.8: Performance of various controllers when climbing up ("Ascend", top plots) and down ("Descend", middle plots) an obstacle and for walking on plane ground after having negotiated the obstacle (bottom plots). p unStab is the probability of becoming unstable at least once when performing the according task (left plots), mean velocity is measured over the whole task (right plots). Note that for the left plots lower values correspond to better performance, whereas on the right side higher values mean better performance.
Error probability is 1% where the differences between lines are about 0.06 for low values, 0.11 for mid range value and 0.13 for high values on the left and where the differences are about 0.02 on the right (see text for details on statistical analysis).

height and attitude control was demonstrated for both controllers.

2.5.1 Torques

It is interesting to note that all controllers (except for Ejoin with standard references angles (#3 S)) optimize the torque distribution in the standing walker in comparison to a system with fixed angles (#1 S) – even though none of the controllers use any force information.

In fact the Ejoin controller with reference angles adjusted for standing (#5 S) is very similar to the system with fixed joints (#1 S). The fixed joint system can be regarded as an Ejoin controller where all gain factors are one. When the gain of a proportional controller (in this case the Ejoin controller) is lower than one, the controller's reference value will only be approximated. Thus the gains of the Ejoin controller introduce slack into the system. Physics then "exploit" this slack to minimize the system-energy. Therefore it is not absolutely necessary to use force feedback to optimize forces.

The same line of argument also covers the optimization going on in the other controllers: proportional controllers are used with few exceptions in the Walknet (#2) and the Akin controller (#9) as well.

The Akin controller (#9) constitutes a rather good benchmark for torques in walking systems. The torques produced by the Akin controller while walking are only marginally higher than the torques it produces while standing. This small rise in the torques can be explained by two factors:

1. The walker configuration changes continually during walking and will necessarily include stretched legs at certain times. These stretched legs contribute unfavourable levers which explain part of the increased torques – in particular in the beta joints which carry most of the walker's weight.

2. The reference velocity control used with the ThreshPorta coordination (sec. 2.2.1.2) continually changes the walking velocity. This continuous acceleration explains another part of the increased torques – in particular in the alpha joint which contribute most to acceleration along the walker x-axis.

$\frac{\text{leg}}{\text{joint}}$	front	mid	hind
α	0.2 - 8.5	0 - 2.6	1.8
β	7.8 - 11.3	14.4 - 21.9	35.6
γ	0.1 - 6.7	1.7 - 5.1	6.8

Table 2.2: Peak torques measured at a walking stick insect, reprinted from [19]. Unit is 10^{-6}Nm.

It should be noted that the torques the Akin controller (#9 W) produces while walking are a bit lower than the torques produced by real stick insects (see [19]), while the torques produced by the Ejoin controller (#3 W) are much higher. For better comparability the according results of [19] are reprinted here (Tab. 2.2). Note that the torques reprinted in Table 2.2 are peak and not mean values. The Akin controller (#9) may be even closer to the real insect when mean torques are compared to mean torques.

The most obvious mismatch between biological and simulation data for the Ejoin controller (#3 W) is in the alpha joints. The insects produce rather low torques in the alpha joints.

Another observation made with stick insects can also not be reproduced by the Ejoin controller (#3): The insects are assumed to

exhibit local positive velocity feedback in the α and γ joints [25, 73]. Superficial tests were made to introduce positive feedback to the Ejoin controller (#3) by simply adding positive and negative feedback signals for the error signal. Application of artificial evolution though, minimized the gains of the positive feedback loops, indicating that this approach is detrimental to walking velocity and/or produced torques.

More elaborate local positive feedback controllers (see [113, 112]) and a combination of positive and negative feedback (see [33, 51]), that were modeled after the behaviour of stick insects, have already been proposed. The Ejoin controller (#3) could serve as a platform to test such propositions in a six legged walker instead of the simple systems tested until now (single leg / cranking). If the α joint controllers (and maybe the γ joint controllers, as appears to be the case in stick insects) of the Ejoin controller (#3) were replaced with such more elaborate controllers, the torque discrepancy between Ejoin and real insect might also vanish. The Ejoin controller is particularly useful for such experiments because it is simple to implement, controls joints completely locally (allowing the test of different controllers for single joints instead of whole legs), and is extremely robust against parameter changes.

Problems with positive feedback as implemented in the Walknet

A close inspection of the torques produced by the Walknet (#2) – that was specifically designed to simulate the positive feedback observed in stick insects, see [73, 27] – while walking, revealed that the walking speed was solely maintained by the α_{min} module of the Walknet.

Without the α_{min} module, the simple positive feedback loop, as implemented in the Walknet, does not maintain the desired stance velocity. It has been shown that the angular velocity of an elastic leg joint under positive feedback control is proportional to the amount of bending (torque) that occurs in the joint [114]. This can be regarded as an active compliance behaviour. Yet, active compliance by definition relaxes the joint bending and therefore deprives the joint of the foundation for an ongoing stance movement.

In order to recover the ability of maintaining the initially imposed joint movement, the stance module of the Walknet can be equipped with a power controlled Local Positive Velocity Feedback (LPVF) controller as introduced in [114, 112]. As has been mentioned above, application of such a controller might lead to smaller differences between simulation and biological results.

2.5.2 Crossing an Obstacle

The most surprising result of the experiments presented here are the good obstacle crossing capabilities of the Ejoin controller. In this task it outperformed the Akin controller in all measured criteria except for the ascension velocity. The results concerning velocity are particularly significant since the Ejoin controller is notably slower on plane ground than the Akin controller. One disadvantage should however be noted: as shown in Fig. 2.8, bottom right, the velocity of the Ejoin controller after crossing the obstacle is more dependent on obstacle height than for the Akin controller. This indicates that the Ejoin controller takes longer to recover from a disruption of the leg phase relationships.

The anticipative attitude control extension of the Ejoin controller (#4) is not un-

ambiguously advantageous. The extension as implemented here is rather sensitive to optimizations. This is indicated by its better performance when ascending very high obstacle as compared to ascending moderately high obstacles: as in all other controllers it was optimized in an artificial evolution where it had to cross an obstacle of height 20 (see sec. 2.3.5). This is exactly the obstacle height where the controller performs best in the very high obstacle range.

However, the results of the Akin controller tests show that anticipative attitude control as observed by [115, 96, 45, 47] can indeed be advantageous in both tasks (ascending and descending), and that the problems of anticipative attitude control in the Ejoin controller are artifacts of the specific implementation in that controller.

The positive contributions of the anticipative height control to the ascending task are comparable to the anticipative attitude extension. But when descending, anticipative height control is of no advantage or even detrimental. This is due to the fact that this extension will raise the walker height before ascending as well as before descending. While ascending this lowers the remaining height between upper obstacle rim and walker body, but when descending that level difference is increased by the same mechanism.

This behaviour might be improved by changing the implementation of the anticipative height control. Instead of using the square root of the variance of the individual leg heights, one could for example use the difference between mean leg height and mean target height.

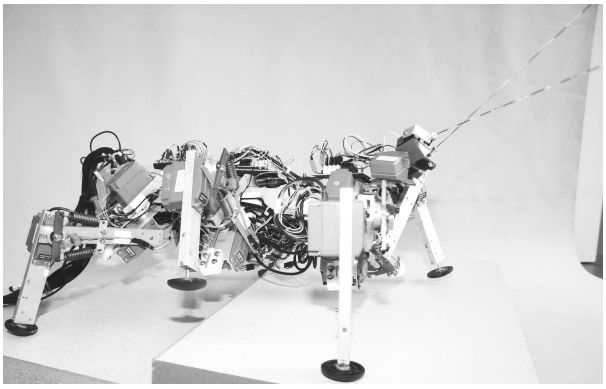

Figure 2.9: The hexapod robot Tarry II with elastic α and γ joints and antennae.

2.5.3 Application to robots

The Akin controller has been used to control the hexapod robot Tarry II that has no adhesive feet (see [112] and Fig. 2.9). In this case the Akin controller was used to test various coordination schemes (ThreshPorta as explained in sec. 2.2.1.2 and the Cruse rules as implemented in the Walknet, see sec. 2.1.3 and [27]) as well as the application of artificial neural network based swing trajectory generators. The Akin controller proofed to consistently produce sensible stance trajectories in these tests. It also turned out to be easily parametrizable. Due to mechanical constraints the tests were limited to plane ground and low obstacles, though.

As shown above, the Akin controller combined with ThreshPorta coordination and adhesive feet has a low probability of becoming unstable when crossing high obstacles. While this poses no problems to insects, it is critical for robots without adhesive feet: the robot would topple in such situations, potentially causing considerable damage to the hardware.

StabPorta, a more reliable alternative to ThreshPorta coordination, was presented in chapter 4. It does not use the threshold T

(see sec. 2.2.1.2) to determine, if a leg may be lifted, but lifts legs as soon as lifting will not lead to unstabilities. It also lowers the reference velocity or stops the walker when unstability is impending. Thus a walker using StabPorta coordination will never become unstable. However, using StabPorta coordination does in some cases result in lock up situations, where the robot would not walk any further.

This dilemma could be resolved in two ways: Either the robot is equipped with adhesive structures at its feet (see [104, 72]) or the StabPorta variant is used and combined with higher level control structures to resolve lock up situations. Such higher level structures could either plan ahead to find suitable footholds / gait patterns in advance (see [86, 94]) or it could kick in only after the lock up occurs and resolve it by reordering leg positions one by one or by walking backwards for one or two steps. The Akin controller lends itself well to the combination with higher level structures since all degrees of freedom of the walker body are separately controlled and could be interfaced with higher level controllers.

The Ejoin controller is currently not suitable for implementation on a robot. It produces excessive torques that may cause mechanical damage to current robots. This might be resolved by introducing positive feedback into the joint control loops as discussed above. Furthermore, the Ejoin controller will frequently produce forces that pull the walker toward the substrate. Without adhesive structures this would lead to lifting legs in stance mode. This would potentially disrupt walker stability and the function of the controller, because it relies on inconsistent effects of different legs cancelling each other out. This could only be addressed by adding adhesive structures to the robot feet (see [104, 72]) or by adding some kind of force control loops. The latter approach would however imply a thorough redesign of the controller.

2.6 Acknowledgements

- Software for programming, simulating, doing the math, the graphics and the text of this chapter and software for running above software was written and made available free – thanks to the people who did that.
- Holk Cruse inspired the work, kept it on track and got this chapter in shape.
- This work was funded by DFG grant Cr58/10 and EC-IST SPARK project.

3 Angular-, Momentum-, Non-relaxing- MMC; Dynamical Weighting

MMC networks [29] provide a general abstraction for solving problems with extra degrees of freedom. In this chapter I propose several extensions to the concept of "mean of multiple computations" (MMC). First, "angular" MMC networks which are based on trigonometry rather than vector math simplify many geometrical problems. Second, the relaxation behaviour of MMC networks can be tuned for better performance when a "momentum" term is introduced. This can be used to avoid local minima, has better damping properties than the traditional approach, and might also allow the simulation of dynamic processes with MMC networks. Third not least, a Posture Optimization MMC network is introduced, that simplifies the model proposed in [76] by orders of magnitude. The realization of the Posture Optimization MMC introduces three other novel concepts: (1) an MMC that never relaxes in the classical sense, (2) dynamical weighting of computations and (3) dynamical damping. All extensions are finally tested by applying them for a geometrical body model of a multi-legged walker.

3.1 Introduction

Consider a four or more legged walker walking over very rough terrain. If the walker is to climb up or down a high step it has to stretch all its legs to the limit to accomplish the task. Therefore, on sufficiently rough terrain it is necessary to operate the limbs and the actuators of the walker close to their kinematic limits. Such a walker might be endowed with sensors to identify terrain properties. Insects for example do have eyes and antennae and robots might have any kind of visual and/or range sensors.

The sensors can provide information about where possible footholds lie but not if these footholds can be reached. Deciding this question is not trivial: Either the foothold is in reach from the current position of the according leg's shoulder joint, or it might still be reachable if the body were shifted to another possible position. In this chapter a Posture Optimization MMC is proposed that is capable to solve this kind of task.

This model can not only be used to check if a valid walker configuration exists for given footholds, it can at the same time also be used to optimize postures for given footholds. Furthermore the results suggest that the model might also be usable as an indicator of dynamical aspects of such postures and of the process of assuming them.

The proposed model is based on a special type of recurrent neural networks, the so called MMC networks. MMC networks

have been used for the control of mechanical devices characterized by extra degrees of freedom [29, 34], for example redundant manipulators, or six-legged walkers [76]. Furthermore, MMC networks have been proposed to be applied for landmark navigation [32]. Those models are based on vector equations that can easily be translated to recurrent neural networks. This chapter introduces novel concepts that can simplify the models significantly. Furthermore additional concepts are introduced that modify the relaxation behaviour of MMC networks.

In the case of MMC networks being used as a kinematic simulation of a six legged walker [76], the model consisted of a computer simulation of the body and environment of a six legged walker. This simulation was used for the development of the controller of a six legged walker [74]. The controller – the "Walknet" – is essentially a feed-forward model (reflex system). While even a simple feed-forward model can show surprisingly complex behaviour [10, 98] – when the loop through the environment is utilized – these reflex systems are nevertheless limited to reacting to sensory input. When a reactive system has to make a decision between two or more paths of action, it cannot evaluate the consequences in advance. In finding a solution to complex problems feed forward systems are therefore limited to trial and error procedures.

This is fine as long as the cost of performing actions is low. However, when finding a solution to a problem would take too long, or when the mechanical system is in danger of being damaged during the trial and error process, the application of internal models of the manipulator and its environment constitute a significant advantage[23, 31]. The evaluation of different paths of actions using an internal model rather than the actual body might be much faster, is probably energetically cheaper and certainly less dangerous.

Kinematic MMC networks can be used to simulate states, i.e. given geometrical body configurations. They can be used to determine whether a given body configuration is consistent with the geometrical and mechanical constraints of the body. A major advantage of MMC networks is that the body configuration can be determined in joint space, Cartesian space or any mixed representation. MMC networks will find solutions of under-determined or determined problems and will find good approximations for overdetermined problems. Furthermore, MMC networks make it simple to describe mechanical or other geometrical constraints.

In the earliest version [29] a multitude of geometrical conditions have been applied to formulate an MMC network. The later version strictly relies on the application of vector equations. For the linear part of this version a convergence proof was possible[122]. This chapter proposes concepts to simplify these models. The simplified versions are used to build a complete geometrical body representation. The model is divided into substructures: the posture optimization network represents the overall body posture in relation to the available footholds and links the sub networks, which represent the manipulators (legs) and their constraints.

3.2 Classical MMC

The basic idea of MMCs is that various variables (e.g. vectors forming a closed chain, Fig. 3.1) can be linked by equations – the value of one variable results from the values of other variables. MMC networks exploit computational redundancy. An MMC

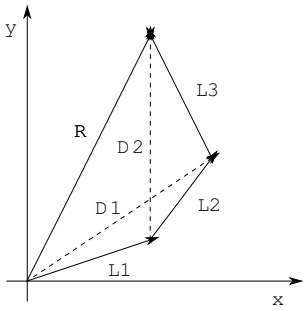

Figure 3.1: An arm consisting of three segments represented by vectors $\vec{L_1}$, $\vec{L_2}$ and $\vec{L_3}$. The position of the end effector is described by vector $\vec{R}$. Furthermore, two additional vectors $\vec{D_1}$ and $\vec{D_2}$, describing the diagonals, are shown.

variable has to be computable from multiple different equations. When the network is not relaxed, different ways of computing the value of a given variable can yield different results. This was the case if the vectors shown in Fig. 3.1 would not form a closed chain. To approximate a solution, i.e. a relaxed state, the *mean* of all results for one variable is computed, lending MMCs their name: "Mean of Multiple Computations".

In an MMC network variables are thus redundantly linked through a matrix of equations. The classical example [34] for this is an MMC network describing a three segment two dimensional manipulator (see Fig. 3.1). The model consists of 6 vectors. Two of these vectors, $\vec{D_1}$ and $\vec{D_2}$, serve the sole purpose of yielding redundant equations to get *multiple* computations for each of the other more meaningful variables, that represent the manipulator segments $\vec{L_i}$ and the vector $\vec{R}$ pointing to the position of the end effector. The variables in the MMC net are the coordinates (x/y) of the vectors depicted in Fig. 3.1. There are six vectors with two coordinate values each, making a total of 12 variables in the network.

To obtain multiple computations for each of these variables all closed paths (Fig. 3.1) with three edges[1] are expressed as vector equations:

$$\begin{array}{rcrcrcl} \vec{L_1} & + & \vec{L_2} & - & \vec{D_1} & = & \vec{0} \\ \vec{L_2} & + & \vec{L_3} & - & \vec{D_2} & = & \vec{0} \\ \vec{L_1} & + & \vec{D_2} & - & \vec{R} & = & \vec{0} \\ \vec{D_1} & + & \vec{L_3} & - & \vec{R} & = & \vec{0} \end{array} \tag{3.1}$$

Each of these equations can be rearranged to yield each of the variables that occur in the according equation (two equations $f_1$and f_2 for each variable):

$$\begin{array}{rcrcl} f_1 & \left(\vec{L_1}\right) & = & \vec{D_1} & - & \vec{L_2} \\ f_1 & \left(\vec{L_2}\right) & = & \vec{D_1} & - & \vec{L_1} \\ f_1 & \left(\vec{L_3}\right) & = & D_2 & - & \vec{L_2} \\ f_1 & \left(\vec{D_1}\right) & = & \vec{L_1} & + & \vec{L_2} \\ f_1 & \left(\vec{D_2}\right) & = & \vec{L_2} & + & \vec{L_3} \\ f_1 & \left(\vec{R}\right) & = & \vec{L_1} & + & D_2 \\ f_2 & \left(\vec{L_1}\right) & = & \vec{R} & - & \vec{D_2} \\ f_2 & \left(\vec{L_2}\right) & = & \vec{D_2} & - & \vec{L_3} \\ f_2 & \left(\vec{L_3}\right) & = & \vec{R} & - & \vec{D_1} \\ f_2 & \left(\vec{D_1}\right) & = & \vec{R} & - & \vec{L_3} \\ f_2 & \left(\vec{D_2}\right) & = & \vec{R} & - & \vec{L_1} \\ f_2 & \left(\vec{R}\right) & = & \vec{D_1} & + & \vec{L_3} \end{array} \tag{3.2}$$

Note that this equation matrix can be transformed into a weight matrix in this spe-

[1] There are 4 such paths in the graph, see Fig. 3.3. Note that in [35] the three possible closed quadrangle paths through the vector graph were also used. These paths are redundant though. The resulting weight matrix is identical whether triangle paths (Fig. 3.3), quadrangle paths or both are used together. Using quadrangles the equation matrices would become more complicated, but the behaviour of the network would be identical because the redundant equations cancel each other.

cial case of an MMC network: each component (x/y) of each vector has to be calculated separately, then. The resulting network will have twice as many equations (or weights respectively), the equation (weight) matrices for calculating x and y values are identical though. By adding for example $f_1\left(\vec{L_1}\right)$ and $f_2\left(\vec{L_1}\right)$ one gets $f\left(\vec{L_1}\right) = \frac{\vec{D_1}-\vec{L_1}+\vec{R}-\vec{D_2}}{2}$. Thus an equation computing $\vec{L_1}$ requires four inputs from $\vec{D_1}$, $\vec{L_1}$, $\vec{R}$ and $\vec{D_2}$ and each weight is 0.5 or -0.5. Corresponding equations can be determined for all other variables[121].

In the case of an equation matrix x/y components can however be disregarded because equations dealing with vectors make sense without disassembling them into their components. This chapter will not deal with weight matrices (as used in [34]) but only use equation matrices because concepts introduced below cannot be easily translated into weight matrices.

Running the network implies calculating the complete equation matrix 3.2. For each variable all equations are calculated and the mean is taken as the value used for the variable V in the next iteration:

$$V = \frac{\sum_{i=1}^{n} f_i(V)}{n}$$

(in our example n is $= 2$). If one wants to calculate vectors $\vec{L_1}$, $\vec{L_2}$, $\vec{L_3}$, that are consistent with a given end effector position $\vec{R}$, one can set $\vec{R}$ and iterate the network until $\sum_{i=1}^{3}\vec{L_i} = \vec{R}$. While iterating the network new values calculated for $\vec{R}$ by the net are overwritten by the preset value of $\vec{R}$. Or equations for computing $\vec{R}$ are not used at all, which is more efficient.

Additionally each equation i for computing variable V can be associated with a weight w_{Vi} so that some relations may be more influential than others:

$$MMC(V) = \frac{\sum_{i=1}^{n} f_i(V)\, w_{Vi}}{\sum_{i=1}^{n} w_{Vi}} \tag{3.3}$$

This is the general form for computing the MMC equation array of one MMC variable V. Note that the weights w_{Vi} are different from the weights referred to in the weight matrix discussion above.

Depending on the kind of MMC network and the problem to be solved, MMC networks might tend to oscillate [122]. Oscillations can be easily damped by adding equations that calculate new values for variable V at time t from V's old value V_{t-1}:

$$f_3(V_t) = V_{t-1} \tag{3.4}$$

Variables with such damping terms tend to keep their old values. This kind of damping is therefore called persistence throughout the rest of this chapter. Combined with weighting as explained above persistence can be arbitrarily strong.

The MMC network described so far comprises a linear system that has the property that the vector lengths for $\vec{L_1}$, $\vec{L_2}$, $\vec{L_3}$ are not fixed. That is a disadvantage when applied to real world manipulators with fixed segment lengths. To keep the length of segment i fixed to a given value l_i one can simply add another equation for each of $\vec{L_1}$, $\vec{L_2}$, $\vec{L_3}$:

$$f_4\left(\vec{L_i}\right) = \vec{L_i}\frac{l_i}{\left|\vec{L_i}\right|} \tag{3.5}$$

In addition, eq. 3.5 includes the damping property of eq. 3.4 and can therefore be used to replace it. Application of eq. 3.5 also

illustrates another reason to express MMC networks as equation matrices rather than weight matrices: In the MMC version described in [121] these nonlinear expansions were described in the following way. The network was divided into two parts, the linear vector part described above and a nonlinear part calculating direct and inverse kinematics. Nonlinear constraints were hidden in the nonlinear part. Instead, the equation matrix version provides a more compact and consistent description and is easily expandable since any constraints and transformations can simply be added to the equations of the according variable.

Another disadvantage of the network described so far is that the state of the segments $\vec{L}_1$, $\vec{L}_2$, $\vec{L}_3$ are expressed as Cartesian vectors while real world manipulators are usually controlled in angle space. Explicit representations of angular values can simply be achieved by adding variables for angles between the segments. These angles can be computed during each iteration. The angle α between vectors $\vec{L}_1$ and $\vec{L}_2$ could for example be calculated from

$$f_1(\alpha) = \arctan\left(\frac{\vec{L}_{1_y}}{\vec{L}_{1_x}}\right) - \arctan\left(\frac{\vec{L}_{2_y}}{\vec{L}_{2_x}}\right)$$

Additional equations for $\vec{L}_1$ would then result from

$$f_5\left(\vec{L}_1\right) = \begin{pmatrix} \vec{L}_{2_x}\cos\alpha - \vec{L}_{2_y}\sin\alpha \\ \vec{L}_{2_x}\sin\alpha + \vec{L}_{2_y}\cos\alpha \end{pmatrix} \frac{\left|\vec{L}_1\right|}{\left|\vec{L}_2\right|}$$

To constrain the workspace of α an equation can be added:

$$f_2(\alpha) = \begin{cases} min & if \alpha < min \\ max & if \alpha > max \\ \alpha & else \end{cases} \tag{3.6}$$

In Hopfield nets there is a simple and elegant way to describe the state of the network: the energy value[64]. Similarly, a scalar value can be introduced to describe the actual state of an MMC network. For this purpose, the network error, a measure to which extent an MMC network has reached its goal state after setting all target parameters and constraints can be calculated in the following way: For each variable V its error E_V is the standard deviation of all its equations: $E_V = \sum_{i=1}^{n} \{f_i(V) - M\}^2$, where M is the mean value calculated from eq. 3.3. The network error is the sum of all variable errors E_i:

$$E_{MMC} = \sum_{i=1}^{n} E_i \tag{3.7}$$

This is related to the calculation of a harmony value used earlier [32]. When the network error becomes zero all target parameters (e.g. a vector $\vec{R}$ in the above example) are consistent with the free variables and all constraints are met. If the error does not become zero but variables do not change anymore, the network has reached its final state allthough some constraints have nevertheless not been met.

3.3 Angular MMC networks

As described above, computing the MMC net with all constraints and explicit angle values requires the computation of the forward and inverse kinematics in each iteration in addition to all other equations. The same goal can however be achieved with a yet simpler network called angular MMC that is based on trigonometric relations instead of vector equations (eq. 3.1).

To this end triangles are used as illustrated in Fig. 3.2. The cosine rule defines

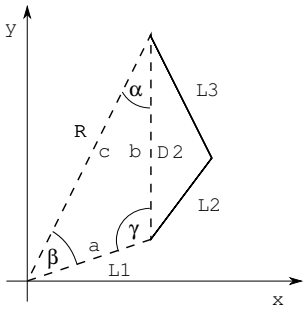

Figure 3.2: An arm consisting of three segments L_1, L_2 and L_3 (compare to Fig. 3.1). The position of the end effector is described by segment R. A triangle is drawn into the polygon enclosed by L_1, L_2, L_3 and R. Three more triangles could be drawn into the polygon accordingly.

the following relations in triangles:

$$\begin{aligned} a^2 = &\quad b^2 + c^2 + \quad 2bc\cos\alpha \\ b^2 = &\quad a^2 + c^2 + \quad 2ac\cos\beta \\ c^2 = &\quad a^2 + b^2 + \quad 2ab\cos\gamma \end{aligned}$$

These equations can be rearranged to yield the lengths of the sides of the triangle:

$$\begin{aligned} a = &\quad b\cos\gamma + c\cos\beta \\ b = &\quad a\cos\gamma + c\cos\alpha \\ c = &\quad a\cos\beta + b\cos\alpha \end{aligned} \tag{3.8}$$

, or to yield the angles of the triangle corners:

$$\begin{aligned} \cos\alpha &= \frac{\frac{b-a\cos\gamma}{c} + \frac{c-a\cos\beta}{b}}{2} \\ \cos\beta &= \frac{\frac{a-b\cos\gamma}{c} + \frac{c-b\cos\alpha}{a}}{2} \\ \cos\gamma &= \frac{\frac{a-c\cos\beta}{b} + \frac{b-c\cos\alpha}{a}}{2} \end{aligned} \tag{3.9}$$

Note that both eq. 3.8 and 3.9 could be simplified, but MMC is about having everything influence everything else, thus this form was chosen.

In the following, equations 3.8 and 3.9 are used as the base equations to construct

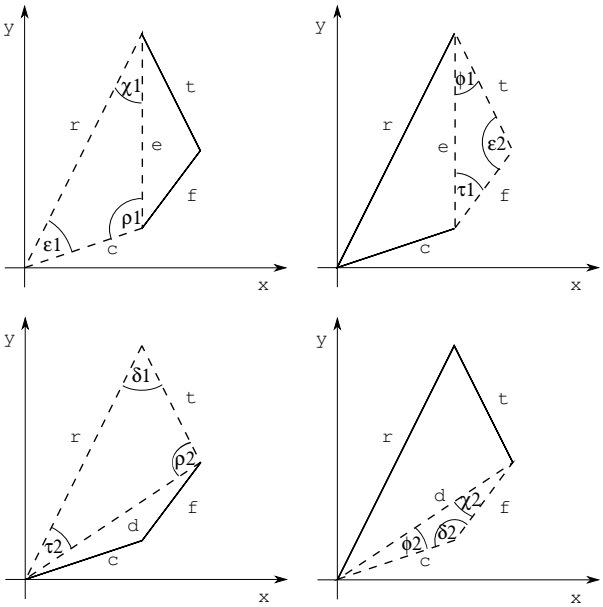

Figure 3.3: An arm consisting of three triangles (compare to Fig. 3.2). All four possible triangles are drawn within the polygon. Variable names are adapted to lowercase (triangle side lengths instead of vectors). Arm segment names are c(oxa), f(emur), t(ibia). Angle names are Greek letters of opposing triangle sides plus index because each triangle side has two opposing angles in two different triangles.

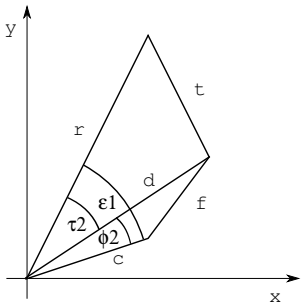

Figure 3.4: Three angles from three different triangles of Fig. 3.3 drawn into one diagram.

the angular MMC. As shown in Fig. 3.3 it is possible to define four triangles enclosed with the arm-target polygon. The target line and arm segments have been renamed to express the change from vector MMC to angular MMC. The whole MMC network consists of six length variables and twelve angle variables (Fig. 3.3). The complete equation matrix consists of twelve equations to compute lengths (eq. 3.8), since each length variable occurs in two triangles and twelve equations to compute angles (eq. 3.9).

The values calculated for the variables by an angular MMC do not have to be consistent with Fig. 3.3 even if r is kept constant accordingly. The reason is that the relation indicated in Fig. 3.4, $\varepsilon_1 = \tau_2 + \phi_2$ is not enforced. The triangles will be inconsistent in most cases and additional computations are required to interpret the output of such a network. It is however possible to simply add such relations to the MMC network for all according angles. In this example three equations result:

$$\begin{aligned} \varepsilon_1 &= \tau_2 + \phi_2 \\ \tau_2 &= \varepsilon_1 - \phi_2 \\ \phi_2 &= \varepsilon_1 - \tau_2 \end{aligned} \tag{3.10}$$

Similar equations result for the three other corners of the polygon. Note that arc cosines and cosines have to be calculated in each iteration if eq. 3.10 are to be used. In that case all triangles will be calculated by the network to be consistent. However, interpreting the results of the simpler version of the network (i.e. without eq. 3.10) once it is relaxed, is computationally simpler than calculating trigonometrical functions each iteration (as required for using eq. 3.10).

Fig. 3.5 shows two examples for the relaxation process of an angular MMC with eq. 3.10. Fig. 3.5 b, d depict a case where the target is positioned outside the workspace. This means that there is no solution to the problem. Nevertheless, the network "tries" to approximate the target as good as possible. Note that the relaxation process would be much faster with lower damping (see eq. 3.4). It was slowed down to give a better impression of the process.

3.3.1 Discussion of angular MMCs

The vector MMC was based on 12 vector equations (eq. 3.2). Apparently the angular MMC has more equations (12 length equations and 12 angle equations) than the vector MMC. In addition the equations of the vector MMC are computationally simpler: they are just sums as opposed to sums, products and fractions in the former.

This is however only true for the linear version of the vector MMC. If segment lengths have to be kept constant, the angular MMC can be simplified by simply neglecting the according length computations: the angular MMC then consists of only six length equations (two for each of r, d and e) and twelve angle equations, whereas the vector MMC becomes computationally much more complicated since vector length computation

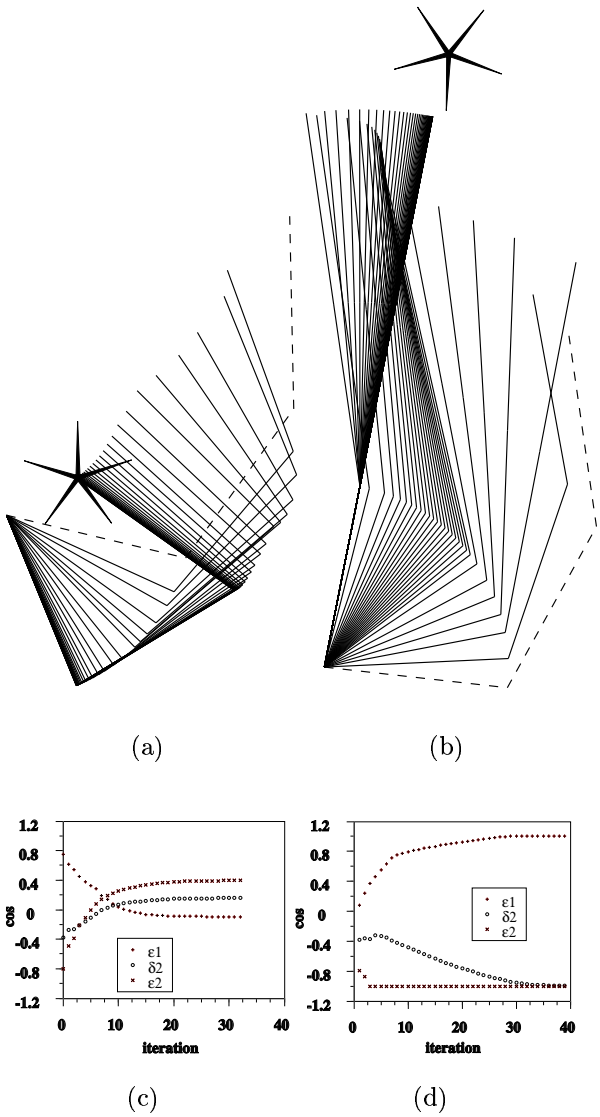

Figure 3.5: Two examples for relaxation processes of angular MMCs. a, b shows the positional changes of the three arm segments. Starting configurations are indicated by dashed lines, target position by stars. c, d the corresponding time course of the joint angles (see Fig. 3.3).

requires squares and roots.

A further advantage is that the angular MMC already operates in angle space: there is no need to compute kinematics each iteration as with the vector MMC. It is not even necessary to compute cosines during the iteration of the simple version (without eq. 3.10) of the angular MMC – every angle that occurs in the equations only ever occurs as $\cos\alpha$. If the network operates in a system where all angles are already encoded as cosines the transformation never has to happen. In biological systems this is rather plausible since the controlled parameter of the sensor/motor devices, i.e. the muscle length, is – in the ideal case – proportional to the cosine of the joint angle.

If signals on sensor or motor side are encoded differently, the transformation has to happen only for setting the network up before starting iterations – e.g. to define a goal state – and/or after the network is relaxed for interpreting the result.

In one experiment the angular MMC was also combined with the classical vector MMC. All equations of the simpler version of angular MMC (without eq. 3.10) were combined with all equations of the linear part of the classical MMC. The segment lengths were used to link the two separate networks: Length constraints (eq. 3.5) were used for all vectors of the classical MMC (not just the three manipulator vectors $\vec{L}_i$). The default lengths l_i were replaced by the length values taken from the angular MMC, which were variable for the diagonals and fixed for the other segments. The vector lengths of the diagonals in the classical MMC were accordingly used for calculating the lengths of the diagonals in the angular MMC. The performance of the network was similar to the performance of the nonlinear classical MMC

and the angular MMC with angle consistency equations (3.4).

Numerical tests indicate that the angular MMC has similar properties as the vector MMC. It apparently finds solutions for all points in the workspace. Workspace constraints can be simply added as equations (similar to eq. 3.6). It also seems to always approximate the best solution for points outside the workspace.

Other than the vector MMC the angular MMC does have singularities where triangle side lengths become zero. This can however be circumvented by a simple procedure deactivating the according equations. In that case no values are computed by the affected equation and the mean value is determined by accordingly fewer equations. Due to the redundancy built into the network it will thus find solutions even at singularities. The network can easily be extended to more segments by adding the according variables and triangle calculations.

Extending the angular network to the 3D case requires the computation of projections of the triangles into planes adding considerable computational complexity. However, since the calculation of 3D kinematics are also rather complex, the angular MMC is still simpler than the vector MMC.

3.4 Momentum

Eq. 3.3 is the general form of the computation of an equation array of one MMC variable. For better readability it is printed again[2] here:

$$MMC\left(v\right) = \frac{\sum_{i=1}^{n} f_i\left(v\right) w_{vi}}{\sum_{i=1}^{n} w_{vi}}$$

This equation is now extended to allow better control of the dynamical behaviour of the network:

$$\begin{aligned} \Delta_{v_{t0}} &= \left(1 - I\right)\left(MMC\left(v_t\right) - v_{t-1}\right) \\ \Delta_{v_t} &= \left(I\Delta_{v_{t-1}} + \Delta_{v_{t0}}\right)\left(1 - F\right) \\ v_t &= v_{t-1} + \Delta_{v_t} \end{aligned} \tag{3.11}$$

, where Δ_{v_t} is the change of variable v at time (iteration) t and $\Delta_{v_{t-1}}$ is the change of v during the previous time step. Two additional parameters are introduced: F and I. $0 \leq F \leq 1$ describes a velocity dependant friction. Δ_{v_t} decreases with higher F. The value $0 \leq I \leq 1$ corresponds to some kind of "inertia". With larger I, Δ_{v_t} depends more on $\Delta_{v_{t-1}}$. This is actually called momentum. v will be constant in systems with $\{I = 1 \wedge \Delta_{v_0} = 0\} \vee F = 1$. Systems with $I = 0 \wedge F = 0$ are identical to the classical MMC (eq. 3.3).

Fig. 3.6 shows the relaxation process of an angular MMC network with momentum.

3.4.1 Damping and stability

This section analyzes the damping properties of the momentum parameters I and F.

Completely linear MMC networks do not oscillate if equation weights are chosen appropriately. In nonlinear systems the oscillations depend on the kind of non-linearity. A minimal MMC network, where the sole MMC variable x is determined by $f_1\left(x\right) =$

[2] Variable name V was exchanged for v since upper case letters were used to indicate vectors.

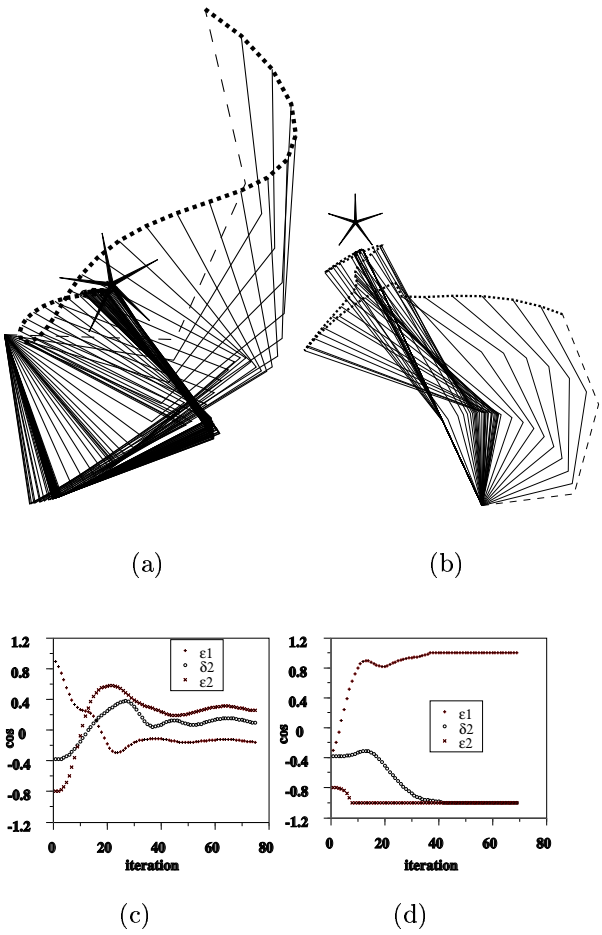

Figure 3.6: Relaxation process of angular MMC with momentum. Inertia I (see text) was set to 0.9 for all equations and friction F was set to 0.05. No damping terms (eq. 3.4) were used. Dotted lines in figure a and b show end effector trajectory (see Fig. 3.5 for details).

$-x^3$, illustrates the damping properties of I and F. The equation weight of f_1 is 1. This system will oscillate stably if $x^3 = x$. This is because the system will jump from $-x$ to x in that case. When $x^3 > x$ the system will build up oscillations. Otherwise it relaxes to $x = 0$. By solving the equation $x^3 = x$ it can be determined that the stable area is $-1 < x < 1$.

This area can be increased by applying persistence terms (eq. 3.4). Inserting the persistence term and $f_1(x)$ into eq. 3.3 we can write the limit case as

$$\frac{x^3 - dx}{1 + d} = x \tag{3.12}$$

, where d is the damping weight. Thus the stable area of the damped system is $-\sqrt{1+2d} < x < \sqrt{1+2d}$.

To calculate the stable area of the same system damped with inertia I and friction F, the values of the limit case are entered into eq.3.11:

$$\begin{aligned} x &= -x + \\ &\left(-2xI + (1-I)\left(x^3 + x\right)\right)(1-F) \end{aligned} \tag{3.13}$$

For this equation it is assumed that the system is currently (time t) at position $-x$. During the last iteration the system jumped from x to $-x$ thus $\Delta_{v_{t-1}}$ was replaced with $-2x$. Eq. 3.13 can be solved to

$$x = \sqrt{\frac{\frac{F(3I-1)-4}{I-1} - 3}{1-F}}$$

i.e. the stable area is $-\sqrt{\frac{\frac{F(3I-1)-4}{I-1}-3}{1-F}} < x < \sqrt{\frac{\frac{F(3I-1)-4}{I-1}-3}{1-F}}$. Fig. 3.7 visualizes the dependence of the stable area on inertia and friction. It is only plotted for $0 \leq I, F \leq 0.9$. The stable area is ∞ for $I = 1 \vee F = 1$.

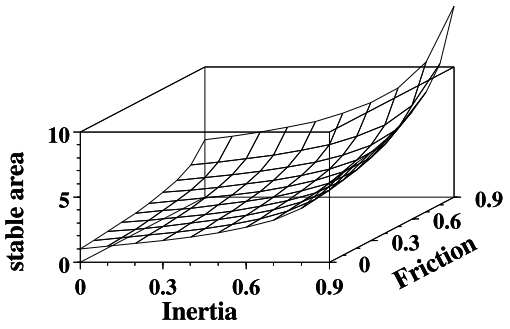

Figure 3.7: Dependence of stable area of simple momentum MMC system with $f_1(x) = -x^3$ on inertia and friction . Ordinate: size of stable area.

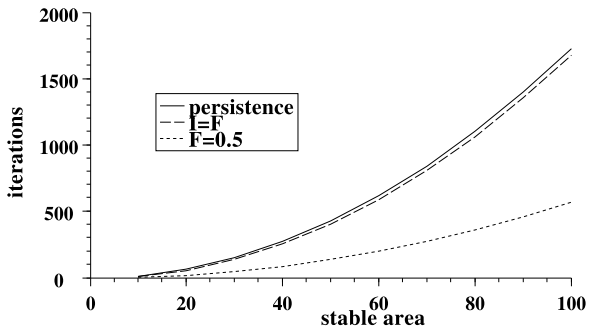

Figure 3.8: Comparison of relaxation speed of simple MMCs ($f_1(x) = -x^3$) with persistence and MMCs with momentum. Abscissa shows the parameter {-stable area $< x <$ stable area} for which the damping parameters were calculated. Ordinate shows the number of iterations it took the model to relax from $\frac{\text{stable area}}{2}$ to $|x| < 1$. For the I=F plot, inertia and friction had the same value, for the F=0.5 plot friction was always 0.5 and inertia was calculated.

Inertia contributes more to the stability than friction, which makes sense since friction lowers the velocity whereas inertia lowers acceleration. Thus while friction might be said to damp the velocity, i.e. it constitutes a first order damping, inertia damps acceleration, i.e. it is a second order damping. Both combined though increase the stable area more than either alone.

To compare damping using the momentum parameters I and F with damping using persistence (eq. 3.4), damping parameters for both were calculated from above equations (eq. 3.12 for persistence and 3.13 for momentum, each solved for the required parameters). It was then tested how long the network takes to relax from $x = \frac{\text{stable area}}{2}$. That value was chosen to prevent the network from oscillating, because in that case "lucky jumps" could take it quite close to $x = 0$. The network was considered to be relaxed if $|x| < 1$. Since damping parameters for MMCs with momentum cannot be calculated directly from eq. 3.13, two combinations were chosen: for one test inertia and friction always had the same value calculated from eq. 3.13, for the other friction was set to 0.5 and inertia was calculated from eq. 3.13. The results are shown in Fig. 3.8. For $F = I$, momentum damping is only slightly more efficient than persistence damping (both were >0.9 for all stable areas shown in Fig. 3.8). When F is kept at 0.5 and I is accordingly higher, momentum damping is much more efficient than persistence damping. Numerical tests not presented here indicate that persistence damping may be mathematically identical to friction damping when inertia is kept to 0. This was tested for the minimal MMC used above and for an MMC that had the additional equations $f_2(x) = -x$ and $f_3(x) = -x^5$.

3.4.2 Discussion of momentum in MMC networks

Momentum in MMC networks, i.e. the inertia parameter I, facilitates the adjustment of the dynamical behaviour. As in other artificial neural network architectures it can be used to heave MMC networks over local minima should they exist. As indicated in the next section, the momentum term can also be used to better simulate dynamic processes. It should be noted though, that such a network will not correctly reflect Newtonian dynamics. It is presumably possible to extend MMC networks to correctly simulate such processes, but that is out of the scope of this chapter.

The friction parameter F is just another way of expressing persistence damping (eq. 3.4) and provides another way to influence the dynamic properties of MMCs. Friction damping has two possible advantages over persistence damping: It is independent from weighting of MMC equations (as in eq. 3.3), whereas the value for persistence damping has to be adjusted when changing other equation weights. And it is expressed in the same way as inertia, which might render it more intuitive.

Together inertia and friction provide a more efficient way of damping MMC networks than friction or persistence damping alone. Lower relaxation times were measured when inertia was used while the stable area was identical. It is however not advisable to use inertia alone, because this will yield a system with slowly decaying oscillations. Friction can accelerate the decay of oscillations significantly, without negatively affecting stability. It indeed even raises stability.

3.5 A Posture Optimization MMC

3.5.1 The model

As mentioned in the Introduction, the main goal of this chapter is to develop a body model that can be used for posture optimization and to find feasible postures for extreme cases of foothold positions. The core of the model is the optimization algorithm which uses a built in optimal posture that it tries to approximate (see [102]). The capability of feasibility evaluation is achieved by introducing further constraints into the model.

Every foothold yields two equations for the MMC, one for body rotation and one for body translation (see Fig. 3.9). Two vectors are constructed to yield these equations. One vector $\vec{F}$ points from the current center of the walker to the according foothold. The other vector $\vec{O}$ points from the current center of the walker to the optimal foothold of that leg with respect to the current orientation of the walker body. The optimal position is defined by the dotted lines in Fig. 3.9 (see also Fig. 3.14) following [102]. Only the equations for the two dimensional case will be presented here, but the model was also tested in 3D. Body translation $\vec{T}$ (for both 2D and 3D cases) is given by:

$$\vec{T} = \vec{F} - \vec{O} \tag{3.14}$$

Body rotation $\dot{\alpha}$ (for 2D) is given by:

$$\begin{aligned} \dot{\alpha}'' = &\quad \arctan\left(\frac{\vec{F}_y}{\vec{F}_x}\right) - \arctan\left(\frac{\vec{O}_y}{\vec{O}_x}\right) \\ \dot{\alpha} = &\quad \arctan\left(\frac{\sin\dot{\alpha}''}{\cos\dot{\alpha}''}\right) \end{aligned} \tag{3.15}$$

.

To extend this equation for 3D cases, the three rotation angles roll, pitch and yaw

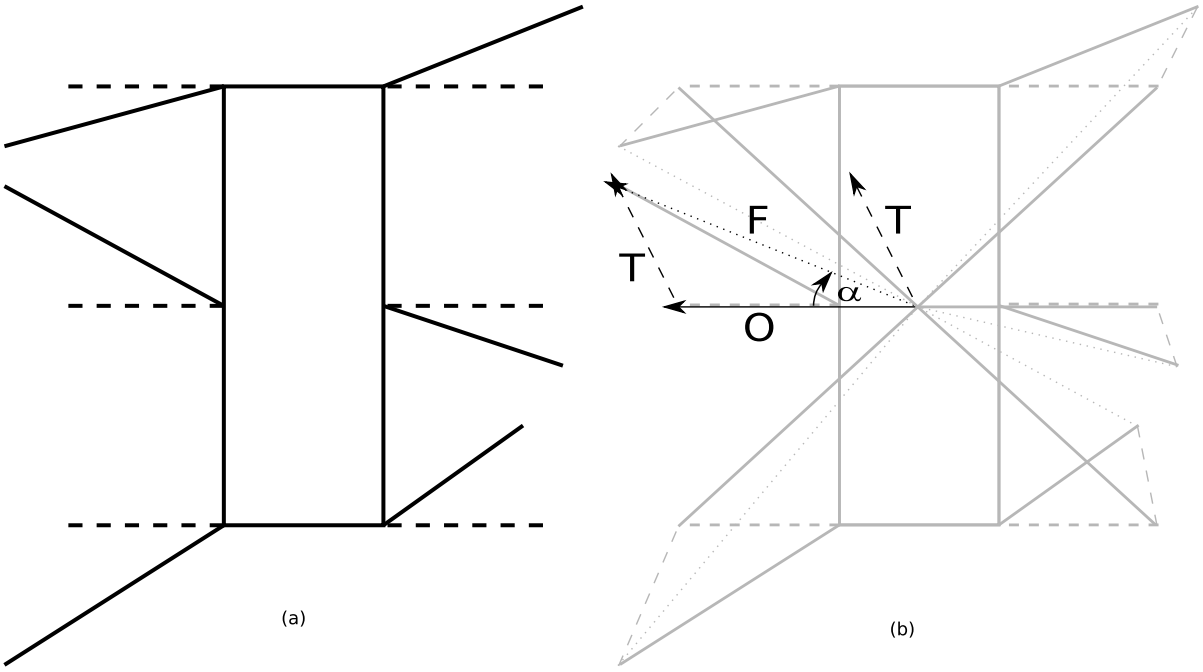

Figure 3.9: Posture optimization. (a) shows the walker body (central rectangle), the actual leg positions (solid lines) and optimal leg positions (dashed lines). (b) shows figure (a) (grey) and lines from the body center to the optimal leg positions (solid lines), lines from the body center to the actual leg positions (dotted lines) and lines from the optimal leg positions to the actual leg positions (dashed lines). For the middle left leg the line between optimal (solid line, vector $\vec{O}$) and actual (dotted line, vector $\vec{F}$) leg position is marked as vector $\vec{T}$ which is equal to $\vec{F} - \vec{O}$ (see text), the angle between optimal leg direction and actual leg direction is marked as angle α. Vector $\vec{T}$ is also shown shifted to the body center.

have to be calculated from the same equation (3.15) with the vector components $\vec{V}_x$ and $\vec{V}_y$ chosen from the three available vector components to reflect the plane in which the according rotation takes place.

Dealing with angular values requires a technical extension: at some point $\dot{\alpha}$ may jump from $-\pi$ to $+\pi$ or vice versa. This can lead to oscillations. To solve the problem a case decision is introduced, where $\dot{\alpha}'$ is the result of eq. 3.15:

$$\dot{\alpha} = \begin{cases} \pi - \dot{\alpha}' & if \dot{\alpha}' > \frac{\pi}{2} \\ -\pi - \dot{\alpha}' & if \dot{\alpha}' < -\frac{\pi}{2} \\ \dot{\alpha}' & else \end{cases} \tag{3.16}$$

Using equations 3.14 and 3.16 for each leg, an MMC network can be constructed for the complete walker (see Fig. 3.9). The resulting MMC network has only two variables being determined by multiple computations: body rotation[3] and body center translation vector[4] . For each of these variables one equation results per supporting leg.

To compute this MMC network the actual footholds $\vec{F}$ have to be computed every iteration according to the translation and rotation of the body. There are no multiple computations for these variables though. The foothold vectors are first shifted and then rotated:

$$\begin{aligned} \vec{F}_t &= \vec{F}_{t-1} - \vec{T}_t \\ \vec{F}_t &= \begin{pmatrix} \vec{F}_{xt}\cos -\dot{\alpha}_t & - & \vec{F}_{yt}\sin -\dot{\alpha}_t \\ \vec{F}_{xt}\sin -\dot{\alpha}_t & + & \vec{F}_{yt}\cos -\dot{\alpha}_t \end{pmatrix} \end{aligned} \tag{3.17}$$

For the 3D case $\vec{F}_t$ can be computed by rotation matrices:

$$\vec{F}_t = \vec{F}_{t-1}\mathrm{M}(\vec{r}) \tag{3.18}$$

[3] or a 3D rotation vector for 3D cases

[4] 2D or 3D depending on the dimensionality of the problem

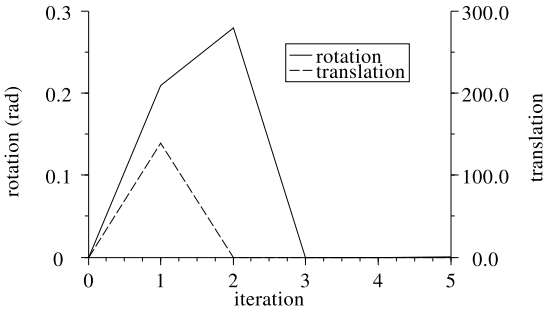

Figure 3.10: Relaxation process of a Posture Optimization MMC as described in the text. Rotation refers to $\dot{\alpha}$ translation is $\vec{T}$.

, where $\mathrm{M}(\vec{r})$ is the rotation matrix of the (roll, pitch, yaw) vector $\vec{r}$.

Fig. 3.10 shows the relaxation process of a 2D Posture Optimization MMC network with 4 footholds. No damping (eq. 3.4) was used. Body center translation ($\vec{T}$) takes only one iteration, body rotation stabilizes after 2 iterations.

3.5.2 Constraints: Dynamical Weighting

This network has a property that did not occur in any MMC network that has been proposed until now: for most problems it will never relax, i.e. the result of eq. 3.7 will never become zero. The network is nevertheless very robust and yields good results while being much simpler and faster to compute than previously proposed MMC networks for similar problems [76].

This model does however have the disadvantage that conventional MMC constraints (e.g. eq. 3.5 or 3.6) cannot be used. The reason for this is that the network does not relax in the sense that traditional MMCs relaxed. It only reaches an equilibrium and traditional constraints will shift that equi-

librium toward the desired state. They will however not enforce that state. To overcome this problem novel constraints are introduced that change equation weights (see eq. 3.3). A Posture Optimization MMC that operates on a real system will for example have to deal with finite leg lengths. In the context of the MMC described here, the length of leg i is given by the absolute value of the leg vector $\left|\vec{L}\right|$.

To calculate $\vec{L}$ a vector $\vec{S}$ describing the relative position of the according shoulder joint is required. $\vec{S}$ vectors are a predefined part of the model like the $\vec{O}$ vectors. $\vec{L}$ is computed as

$$\vec{L} = \vec{F} - \vec{S} \tag{3.19}$$

.

To enforce the limited leg length max (i.e. $\left|\vec{L}_i\right| < \max$), the weights of those equations, where $\vec{F}_i$ occurs[5], have to be dynamically changed to achieve the desired states:

$$w_i = \begin{cases} 1 & \text{if } \left|\vec{L}_i\right| < \frac{\max}{2} \\ \frac{\max}{\max - \left|\vec{L}_i\right|} & \text{if } \frac{\max}{2} < \left|\vec{L}_i\right| < \max \\ \max & \text{else} \end{cases} \tag{3.20}$$

This dynamical weighting must be applied to the corresponding equations for the calculation of $\vec{T}$ and it can also be applied to the computations of α. Dynamical weighting introduces non-linearities into the model that lead to oscillations when $|\vec{L}| > \frac{\max}{2}$. These oscillations can be successfully damped using persistence (eq. 3.4) or momentum terms (eq. 3.11) for the computations of $\vec{T}$ and α.

Fig. 3.11 shows the relaxation process of a length constrained MMC with dynamical weighting on $\vec{T}$ and α computations and persistence. The damping was the same for all computations: the damping weight was 50, all other weights were 1.

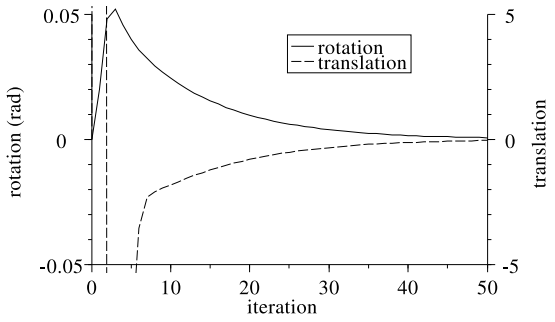

Figure 3.11: Relaxation process of a Posture Optimization MMC with a length constraint, dynamical weighting and persistence as described in the text. See Fig. 3.10 and main text for further explanations.

The relaxation process takes much longer than in an undamped Posture Optimization MMC. The rotation in particular changes significantly until iteration 30.

The optimal damping value depends on the problem at hand. The damping value for this experiment was hand picked. It is close to the optimum for the given problem. In picking a good damping value one has to make a trade off between an acceptable potential oscillation amplitude and network relaxation velocity. Sufficiently high damping values eliminate oscillations completely, but they also slow down the network significantly.

3.5.3 Dynamical Damping

To solve the problem of optimizing the damping values for non-linear problems as described above, damping values can be determined dynamically. The damping weight

[5] because $\vec{F}_i$ is the only *MMC* variable involved in the computation of $\vec{L}$

w_{dv} for a given MMC variable v (eq. 3.4) is determined during each iteration after the mean result for v was computed from eq. 3.3 using w_{dv} as determined during the previous iteration. The equations for calculating w_{dv} are

$$sc(x_t) = \text{sign}(-x_t x_{t-1}) \quad (3.21)$$

$$hr(x) = \text{sign}(x)\, x \quad (3.22)$$

$$li(x_t) = \sum_{i=1}^{t} x_i \quad - hr(li(x_{i-1})\, l) \quad (3.23)$$

$$w_{dv} = li(sc(MMC(v_t) - v_{t-1}))\, k \quad (3.24)$$

, where

$sc(x_t)$ is a sign change detector function that returns 1, if the sign of iteration t is different from the sign of iteration $t-1$, and 0 otherwise;

$\text{sign}(x)$ is the signum function;

$hr(x)$ is a half wave rectifier;

$li(x_t)$ is a leaky integrator that looses $0 \leq l \leq 1$ times its value during each iteration – though the computation of $li(x_t)$ appears to take longer for each iteration, it can easily be implemented to take the same time for each iteration;

k is an amplification scale that determines the strengths of the dynamic damping effect.

Eq. 3.24 increases the damping when sign changes in the derivative of the calculated value are detected – such sign changes indicate oscillations. The damping will be increased immediately, there is no time delay as with low pass filtering. The longer the system oscillates, the stronger the damping will become – independently from the amplitude of the oscillations. Due to the leak in the integrator the effect will degrade over time. This has the advantage that the system can become fast again after damping oscillations. It does however have the disadvantage that the system is frequency dependent and only partly damps oscillations with a sufficiently low frequency.

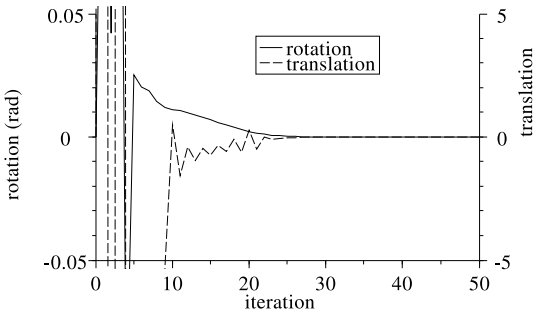

Figure 3.12: Relaxation process of a Posture Optimization MMC with a length constraint, dynamical weighting and dynamical damping as described in the text. See Fig. 3.10 and main text for further explanations.

Fig. 3.12 shows the relaxation process of a Posture Optimization MMC with dynamical weighting and dynamical damping. The value of the parameter k was 15, the value of l was 0.05 in this example.

This system starts with high amplitude oscillations which are quickly damped – after about 5 iterations. The system relaxes notably faster than a system with fixed damping optimized for the same task.

3.5.4 The complete model

The posture optimization MMC network only represents the body posture in relation to the footholds. Angular MMCs (section 3.3) can be used to represent the legs and for example introduce angular workspace constraints into the complete body model.

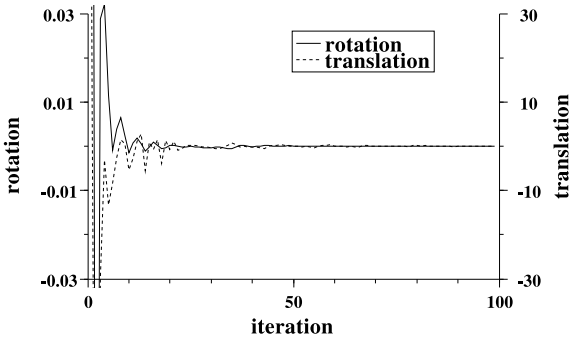

Figure 3.13: Relaxation process of the complete model with four angular leg MMCs (simple version without angle consistency, i.e. eq. 3.10), a central posture MMC, dynamical weighting and damping. Abscissa shows iteration number, ordinates show rotation and translation (differently scaled).

Each leg network is linked to the posture network by two variables. One variable – leg length – passes information from the posture network to the leg network. The other variable – MMC error – passes information from the leg network to the posture network. Leg length in the posture MMC is $\left|\vec{L}\right|$ as computed from eq. 3.19. In the angular MMC it is the length of the radius r that enters into equations 3.8 and 3.9. The MMC error of the leg sub-network can be computed from eq. 3.7. It enters the posture MMC as a dynamical weight for computation of $\vec{F}_i$ and replaces eq. 3.20.

By using the MMC errors of the leg sub-networks as dynamical weights in the posture network, effects of, for example, angular workspace constraints in the legs will automatically propagate into the posture net, which mediates between the legs. Fig. 3.13 shows a relaxation process of the complete model. For this plot simple versions of the angular MMC (without eq. 3.10 for angle consistency) were used. The whole network was dynamically damped ($l = 0.1$ in eq. 3.23 and $k = 20$ in eq. 3.24). The complete model is much less susceptible to oscillations, when the leg networks are faster than the posture network, thus the leg networks were iterated twice for each iteration of the posture network.

3.5.5 Discussion

A Posture Optimization MMC is introduced that solves a similar task as the one proposed in [76], but is orders of magnitude simpler and relaxes much faster. The network proposed here solves a slightly different problem than [76], though. The latter would only relax to postures that are consistent with given constraints (e.g. leg lengths and/or joint work-spaces) and not optimize the posture as the network proposed in this chapter does. The network presented here can however be used to solve the constraint problem too, without optimization. To this end one simply has to change eq. 3.20 to:

$$w_i = \begin{cases} 0 & \text{if } \left|\vec{L}_i\right| < \frac{\max}{2} \\ \frac{1}{\max - \left|\vec{L}_i\right|} & \text{if } \frac{\max}{2} < \left|\vec{L}_i\right| < \max \\ 1 & \text{else} \end{cases}$$

, and not use computations of unconstrained variables. The behaviour of the system will then be similar to the one presented in [76].

The use of dynamical weighting shows how computation weights can be used to serve different purposes. Unsymmetrical equation weights have been proposed together with the earlier MMC networks [120]. The proposition of a Posture Optimization MMC is however the first work to demonstrate the usefulness of computation weights in MMC networks.

The dynamical damping is in fact just another instance of dynamical weighting since the variables controlled by dynamical damping are computation weights – weights of damping computations respectively. As indicated by Fig. 3.12, dynamically damped MMC networks are much more susceptible to oscillations than systems with a high, but fixed damping value. The reason for this is that dynamically damped systems start off *undamped*. The damping only starts when oscillations are detected (eq. 3.21). The damping weights increase fast and are then lowered (eq. 3.23) until a stable state is achieved or until an equilibrium is reached between oscillation induced damping buildup (eq. 3.21) and the damping decrement (eq. 3.23). The overall effect is a system that relaxes faster than systems with fixed damping. At the same time dynamically damped systems can be parametrized to stabilize MMC networks against a broader range of possible oscillations since dynamically damped systems can successfully damp oscillations of any amplitude (eq. 3.22). The only problem that remains to be solved are low frequency oscillations. These however occurred rarely in the experiments and had a low amplitude.

All concepts introduced in section 3.5 were tested in 2D and 3D. 3D problems are more susceptible to oscillations than are 2D problems. The reason is that oscillations in one rotational degree of freedom can propagate to other rotational degrees of freedom when the roll, pitch, yaw convention is used. These oscillations can however be tackled with fixed or dynamical damping. All of these concepts were tested each by itself and in different combinations – without apparent problems.

The Posture Optimization MMC was also tested with momentum terms (section 3.4)

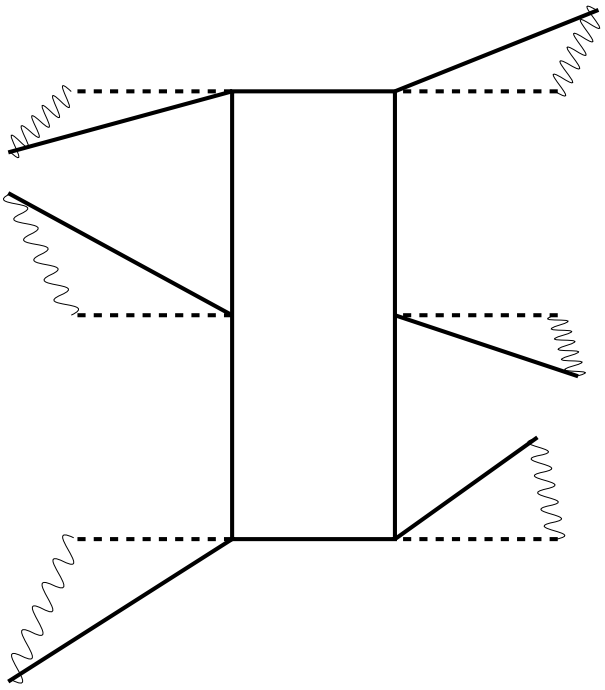

Figure 3.14: A mechanical posture optimization system. The central rectangle depicts the walker body. Actual leg positions are indicated by solid lines and optimal leg positions by dashed lines. This can be considered as a wire frame mechanical model where actual leg positions are connected to optimal leg positions by springs, which move the body to a "most comfortable" position.

alone and in combination with other concepts introduced in section 3.5; if e.g. dynamical damping is to be combined with momentum, eq. 3.24 controls the friction parameter F of eq. 3.11. Such a system behaves remarkably like a real world physical model as depicted in Fig. 3.14. This indicates that MMC networks with momentum might also serve as models for dynamical processes. To verify this, one would have to compare the proposed system with a real mechanical model or at least with a dynamics simulation of such a system.

The equation based nature of the model may make it implausible that such a model could also be used in biological systems. However, in [110] it was shown that the computations necessary for the basic model – vector translations and rotations, see sec. 3.5.1 – can be implemented in artificial neural networks using population coding and possibly also in real neural systems.

3.6 Conclusions

Several new concepts were introduced in this chapter that extend the theory of MMC networks significantly. All of these extensions were demonstrated by specific examples. They are however not constricted to these examples. Using these concepts MMC networks can be implemented for solving completely different problems.

The momentum concept can be used to solve any problem that has local minima in its state space. Dynamical weighting can be used to introduce constraints into any MMC system that does not relax in the classical sense and probably has other applications, too. Dynamical damping can be applied in any MMC network that suffers from oscillations. Angular MMC networks can be used for simulating any multi segment manipulator.

This chapter introduced novel approaches to problems that have previously been modeled with MMC networks. The novel approaches simplify the problems and extend the available tools to model various problem with MMCs.

3.7 Acknowledgements

- Software for programming, simulating, doing the math, the graphics and the text of this chapter and software for running above software was written and made available free – thanks to the people who did that.
- Malte Schilling helped with some equations.
- Holk Cruse inspired the work, kept it on track and got this chapter in shape.
- This work was funded by DFG grant Cr58/10 and EC-IST SPARK project.

4 Comparing different controllers for the coordination of a six legged walker

This chapter compares four models for coordination: the Cruse rules [27], two extended versions of a model based on ideas by Porta & Celaya [100] and one model that is based on the idea of MMC networks [122]. All models are capable of coordinating a six legged walker on level terrain and of crossing low obstacles. The experiments presented here show that an MMC network for coordination is feasible and that the Porta variants, which are based on rather intuitive ideas, show the best performance.

4.1 Introduction

4.1.1 Problem Definition

The problem of controlling the gait of a six legged walker can be broken down into three subproblems:

- Generating swing trajectories, in order to move legs from the liftoff point or posterior extreme position (PEP) to the touch down point or anterior extreme position (AEP) (that movement is hence called protraction, see Fig. 4.5)
- Generating stance trajectories, to move legs from AEP to PEP (that movement is hence called retraction, see Fig. 4.5)
- Determining the positions of AEPs and PEPs, i.e. deciding when to lift legs and where to put them

As the last issue of this list contributes to the coordination between the different legs, this will be called the coordination problem in this chapter. The solution of the coordination problem can become anything from trivial to extremely complex, depending on intended walking path (straight or more or less curved), ground properties (even, sloped, rough and/or patchy) and desired walking performance (e.g. velocity and stability).

On even ground already very simple coordination schemes can cope with straight paths and all sensible stability and velocity requirements [54, 16]. An often used example is a simple oscillator that alternatingly lifts three of the six legs [16], generating what is called a tripod walking pattern [126, 56], where three non-neighbouring legs are lifted simultaneously. However, tripod gaits could also be regarded as a special case of wave gaits [126]. The latter are described as a wave of swing movements running along the body from rear to front [9].

The solutions to the coordination problem, that have been proposed in the past, can be broken down into two broad groups: fixed gait approaches and free gait approaches [80]. Gaits are referred to as "fixed", if internally produced recurring patterns are

used for coordination.

The coordination schemes tested in this chapter all belong to the free gait group – coordination is generated by local rules, which may lead to somewhat irregular patterns. Free gaits have the advantage that they are more flexible and can in principle cope with more difficult situations. On the downside the generation of free gaits is an accordingly more difficult problem, because it is not obvious how to guarantee body stability in every situation.

Coordination schemes can also be classified based on their internal signal flow organisation rather than by their use of pattern generators - or lag thereof. The most common classification of implementations divides coordination schemes into local rules and global/central systems. If a system is based on local rules, legs are lifted solely based on locally available information - e.g. the state of the affected leg itself, or at most the states of adjacent legs. Global systems can take the states of all legs into account, when deciding about lifting one leg.

Advantages of local rules are their superficial simplicity and - leaning from that - their simple implementation. The behaviour of a system of local rules can however only be predicted with difficulty. Therefore it is also hard to design such a system to meet certain criteria (e.g. body stability). Systems of local rules commonly generate free gaits.

The best tested system of local coordination rules is the one devised by Cruse [27], which is based on six-legged insect walking. Calvitti & Beer conducted a detailed analysis of the system from a dynamical systems perspective [17]. Due to the complexity of the interactions between different local rules *and* the mechanics of the walker, forming a loop through the environment, only single, isolated coordination rules could be analysed as yet [17]. Kindermann tested the model extensively in a simulation of a six legged walker [75]. This allowed him to evaluate the performance of all coordination rules being active at the same time plus the coupling with the mechanics of the walker through the external world as well as the dynamics of other controller components.

In this chapter I adopt Kindermann's ethological approach to system analysis, i.e. the behaviour of the complete, rather complex system is analysed and quantified. The systems are run in a complex simulation and the behaviour of the systems is observed. This approach yields a statistical description of aspects of the observed behaviour rather than a complete formal (mathematical) analysis. Instead of the thorough investigation of only one system, here I compare the performance of four different coordination schemes, using the same testbed:

1 Cruse's coordination rules,

2 and 3 two extended versions of a model proposed by Porta & Celaya [100] and

4 a newly developed model, called the MMC model.

All four belong to the class of free gait controllers. The Cruse rules can be implemented as local rules. The other coordination schemes can be mostly implemented as local rules, but use and control a few global parameters like retraction velocity that affects all legs or global stability.

4.1.2 The Cruse model and the Porta & Celaya model

Both the Cruse model and the Porta & Celaya models determine the position of the

PEP of a given leg by taking only parameters of neighbouring legs into account. In all models (including the MMC model) only the x-component of PEP positions is used. Note that the coordinate system for PEPs (and other coordination parameters) is rooted in the shoulder joint of the according leg.

Cruse's model [27] uses default positions for all PEPs. The local coordination rules shift these default PEPs along the x-axis (see Fig. 4.5). Cruse's model as implemented here introduces three coordination rules (see Fig. 4.1):

1. The PEP of the affected (e.g. middle left) leg is shifted *backwards* along the x-axis (by 15 to 20 units depending on leg, see tab. 4.1 for scaling of values), if the *ipsilateral posterior* (e.g. hind left) leg is protracting. This *prolongs* retraction in the affected leg.

2. The PEP of the affected (e.g. middle left) leg is temporarily shifted *forward* (by about 3.5 units), if the *ipsilateral posterior* (e.g. hind left) leg or the *contralateral* (e.g. middle right) leg just changed from protraction to retraction. This *shortens* retraction in the affected leg.

3. While the *ipsilateral anterior* (e.g. front left) *or the contralateral* (e.g. middle right) leg is retracting, the PEP of the affected (e.g. middle left) leg is shifted *forward* proportionally (by about -1 to 6 units depending on state of influencing leg). This *shortens* retraction in the affected leg.

Legs are lifted, if the current position lies behind the PEP thus determined.

In order to describe the model proposed by Porta & Celaya [100], the term of neighbouring legs has to be defined. According to the Porta & Celaya model, legs connected by arrows in Fig. 4.1 (a) are neighbouring - with the exception of the middle legs, which are *not* considered neighbours. Thus each leg has exactly two neighbours. According to the Porta & Celaya model a leg is lifted, if it has a higher lifting priority than both neighbouring legs.

Lifting priority is defined in the following way: Protracting legs have the highest lifting priority (this implicitly means, that neighbouring legs cannot protract simultaneously – the prime condition which Porta & Celaya wanted to avoid). For retracting legs, the lifting priority is negatively proportional to the leg's distance from its physical PEP (the closer a leg is to its physical PEP, the higher its lifting priority). The physical PEP is determined by leg geometry (leg segment lengths). It is the hindmost point a leg can reach in a normal walking position.

In [100] Porta & Celaya propose an optimisation on that model: As a shortcut for "leg a has higher lifting priority than leg b" we say a>b. Then if a>b>c>d, change priorities to a>b<c>d - leg c can be lifted in spite of the original rule. Vice versa: If a<b<c<d, change priorities to a<b>c<d - leg b can be lifted. Since protracting legs always have higher lifting priorities than *both* neighbours, the rule not to lift neighbouring legs simultaneously is not violated. However, since farther neighbourhood relations have to be taken into account, the model is less local when using this optimisation procedure. Without the optimisation, only immediate neighbourhood relations are required for coordination. The optimisation was used in the experiments presented here.

Note that in [102] Porta & Celaya proposed to determine the lifting priority by temporal parameters. In that paper they also describe how they determine the ref-

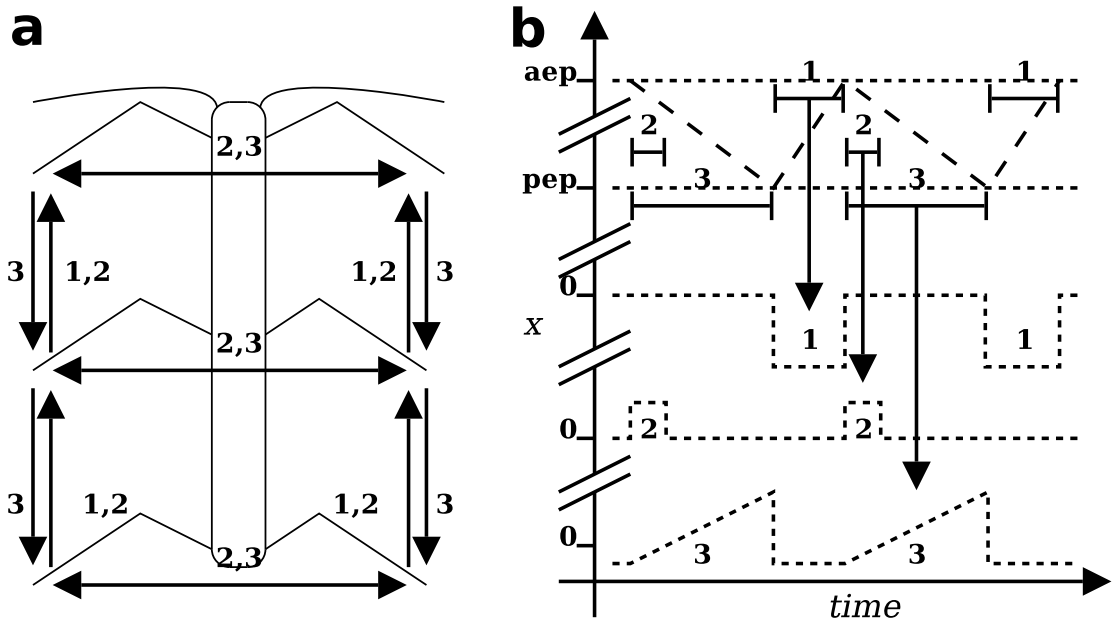

Figure 4.1: This figure illustrates Cruse's coordination rules. a) shows which legs influence each other: arrows point from influencing to affected legs. The numbers refer to the enumeration of coordination rules explained in the text (section 4.1.2).
b) illustrates how each influence works. Time is on the abscissa, the ordinate denotes x-positions (see Fig. 4.5) of the tips of the legs relative to the long axis of the body. Higher x-values indicate anterior positions. The dashed zig-zag line at the top shows the presumed position of an influencing leg. During protraction the leg is moved forward relatively to the body - the line ascends. During retraction the line descends. The solid lines below the numbers show the periods during the cycle in which the coordination rules are active. The three dotted lines below illustrate, how each coordination rule shifts the PEP of the affected leg away from the default PEP (marked 0 at the ordinate).

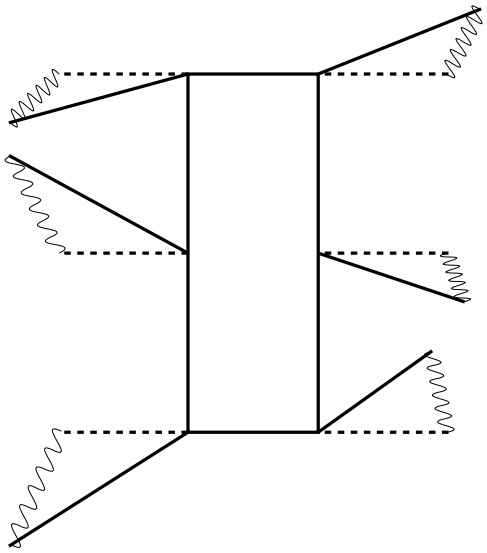

Figure 4.2: Physical analogue of retraction trajectory control according to Porta & Celaya [102]. An optimal posture of the legs is defined (dotted lines). Each retracting leg tries to minimise the distance from its actual position (solid lines) to its position in the optimal posture. The minimisation follows a gradient (indicated by springs in the figure). The resulting translational and rotational vectors are averaged to determine the global movement vector for all legs. Retraction in a given leg is achieved by protracting other legs to new positions: legs that step (protract) will shift their positions forward (in front of their optimum), "dragging" the body with them by minimising their distances to their according optimal positions once they reach the ground. Retraction velocity is thus determind by protraction trajectory length. Curved walking can be achieved by making protraction tractories on one side of the body longer than on the other.

erence velocity ($v_{r\,ref}$, for a discussion on retraction velocity control in walking sticks see [37]). Their method is however closely linked to their method for leg trajectory generation which I did not reproduce (see Fig. 4.2). Instead I propose an even simpler approach: as legs get close to their physical PEPs, the retraction velocity v_r for all legs is lowered. The retraction velocity can reach zero. This approach is somehow similar to the one discussed in [102]. In the Porta model one leg that approaches its physical extreme will "stretch its spring" (Fig. 4.2) exerting a backward force on the body, eventually halting retraction altogether.

Porta & Celaya designed their model for the walker to achieve the highest possible v_r. I tested the models with different reference retraction velocities $v_{r\,ref}$. The original model performs very poorly with low retraction velocities, since it lifts legs as soon as possible. With low retraction velocity, the PEPs are thus shifted very far forward - eventually to positions shortly behind the AEPs. This can lead to grave instabilities (the centre of mass lies outside of the support polygon which would let the walker topple in real world, see detailed discussion of stability in section 4.3.2). Therefore I applied two different extensions to the model.

4.1.2.1 ThreshPorta

According to this version of the Porta & Celaya model, its lifting rules were *only* applied, if the distance (D) of a given leg from its physical PEP was smaller than a preset threshold (T). T was set to 5.0 length units in our experiments (compare to mean step size of 20 units, see table 4.1).

These parameters are used for calculating the actual retraction velocity v_r. Each retracting leg proposes a "desired" global re-

traction velocity accorded by $v_r = \frac{D}{T} v_{r\,ref}$ if $D < T$, and $v_r = v_{r\,ref}$ otherwise. The leg with the smallest proposal determines the actual $v_{r\,ref}$.

4.1.2.2 StabPorta

The second extended version of the Porta & Celaya model made use of the threshold T for calculation of v_r only. Instead of a simple application of the lifting rule of Porta & Celaya, it was checked whether the walker would become instable if that leg was lifted: A leg is only lifted if the walker retained static stability. Instability in this context was reached if the stability margin (see section 4.3.2) became < 0.5. Note that this extension implies some planning ahead.

4.2 The MMC Model

4.2.1 Introduction

In addition to the above mentioned reactive control systems, yet another approach to coordination is introduced here that is based on a "mental model" [31, 23, 75]. Certain aspects of the coordination problem are modelled by linking essential parameters of coordination by various equations. High level constraints are then applied to some of these parameters. Through the network of equations all parameters are calculated to be consistent with the constraints and with each other. The essential parameters, i.e., the degrees of freedom in the coordination problem considered here, are the positions of AEPs and PEPs and the protraction (swing) and retraction (stance) velocities, v_p and v_r.

In this chapter, the MMC model is used with high level constraints that prevent neighbouring legs from swinging at the same time. Theoretically the model should also be flexible enough to comply with other requirements of coordination and requirements made by higher level structures [55] (e.g. AEPs determined by patchy ground, predetermined protraction velocity, desired walking direction ...). The MMC model exploits all remaining degrees of freedom (i.e. all parameters – AEPs, PEPs, velocities ... – that are not determined by constraints as above) to produce parameters that are consistent with the requirements, regardless of whether the problem at hand is under-determined or not.

4.2.2 Introducing MMC Networks

The problem of leg coordination is modelled using "mean of multiple computations" (MMC,) nets [122, 76, 35]. A mathematical proof exists that MMC nets reliably produce consistent results for under-determined linear systems [122]. Anecdotal evidence from computer simulation suggests that MMC nets can also cope with nonlinear systems [76].

The MMC nets established and tested by Cruse and collaborators, were modelling kinematic systems, especially three degrees of freedom actuators [122, 35] and an 18 degrees of freedom kinematic model of a stick insect [76]. Recently, this type of network was applied to simulate landmark navigation [32]. These MMC nets could be represented by asymmetric weight matrices of recurrent neural networks with optional constraint characteristics in the recurrence loop. The weight matrices were derived from vector equations describing the kinematic system.

However, the MMC principle [35] does not necessarily imply weight matrices. It re-

quires a consistent set of equations that describe the subject-matter to be modelled. Each equation is then calculated in each iteration. If several equations yield results for the same variable, the mean is used. The result is fed back to all equations for the next iteration. Optionally the results can be passed through constraint characteristics [35], before being fed back. If all equations for a given variable yield the same result, the error for that value is zero. If the errors for all variables computed by the network are zero, the net is "relaxed" [35] and iteration stops.

When calculating the mean of different results for the same value, each result (i.e. equation) can be weighed [35]. If the sum of all such weights is 1, then the weighed mean of the different results is the sum of the weighed results. To prevent oscillations, the network can be damped. To damp oscillations of a given variable, last iteration's value for that variable also enters into the mean of results for that variable [35]. Using weighting of equations, heavy damping is possible. MMC nets can easily be implemented in the usual form of recurrent artificial neural networks, if all equations can be reorganised to be sums of factors (i.e. weighed sums) of the results of the other equations. This is not true for the MMC-net introduced here, since products and ratios are used.

4.2.3 Basic Equations

Coordination can be considered to be a problem determined in space and time. Many coordination schemes explicitly determine spatial parameters [100, 27, 50, 94, 85, 109]: the positions of AEPs and PEPs. But one of the most basic requirements on any coordination scheme is that it prevents adjacent legs from swinging at the same *time*, because this would almost certainly let a walker topple [100]. Therefore the coordination model proposed here explicitly represents time (see also [55]). Nevertheless, coordinates for AEPs and PEPs are determined in space *and* time. Spatial representation is convenient, since it allows calculation of stability measures.

The MMC system introduced here establishes equations to calculate AEPs and PEPs and other important parameters of coordination. All equations are linked in a MMC network for the coordination of the whole animal where the parameters determining the state of one leg are propagated to both neighbouring legs. Coordination is achieved by imposing constraints on certain parameters that relate neighbouring legs (Δij, see below). This MMC network is related to the kinematic MMC network described in [122, 76] only in its use of the MMC principle. It constitutes a novel application of the MMC principle in a rather different context. It is also the first time an MMC network has been devised that cannot be represented as a weight matrix, because products and ratios are used in the equations. The following relations (see Fig. 4.3) are applied to provide the basic equations that are used to construct the MMC system (to simplify the problem only the x-component (see Fig. 4.5) of coordinates was used):

For a given leg, the

- time to AEP ta and
- time to PEP tp

can be determined from

- the current position x,
- the aep coordinate,
- the pep coordinate,

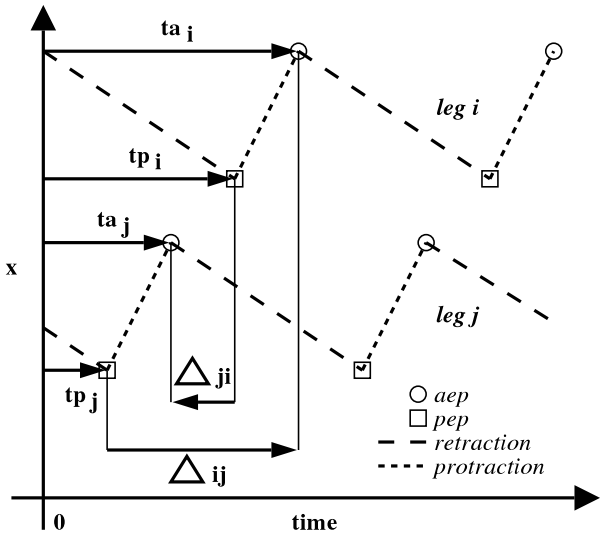

Figure 4.3: Illustration of ta, tp and Δij. Coordinate system analog to Fig. 4.1 b. Current leg position x (see Fig. 4.5) is shown at time = 0 (ordinate). Dashed and dotted lines show presumed positions of leg i and leg j through time.

- the protraction velocity v_p and
- the retraction velocity v_r

The equations for computing ta and tp depend on whether a leg is currently protracting or retracting. As illustrated in Fig. 4.3, we obtain

$$ta = \frac{aep - x}{v_p} \tag{4.1}$$

$$tp = \frac{aep - x}{v_p} + \frac{aep - pep}{v_r} \tag{4.2}$$

for protraction and

$$ta = \frac{aep - pep}{v_p} + \frac{x - pep}{v_r} \tag{4.3}$$

$$tp = \frac{x - pep}{v_r} \tag{4.4}$$

for retraction. In Fig. 4.3, v_r and v_p are represented by the slopes of the dashed and dotted lines.

4.2.4 Relating Legs

Since coordination is about the relation between legs, a variable to relate two legs has to be established: Variable Δij denotes the time that passes between the occurrence of the next AEP in leg i (at time ta_i) and the occurrence of the next PEP in leg j (at time tp_j; see Fig. 4.3):

$$\Delta ij = ta_i - tp_j \tag{4.5}$$

Any too legs could be thus related, e.g. by calculating $\Delta ih = ta_i - tp_h$. However Δ values are only calculated for neighbouring legs in this model (h and i are neighbours as are i and j but not h and j, i.e. $\Delta ij = ta_i - tp_j$, $\Delta ji = ta_j - tp_i$, $\Delta ih = ta_i - tp_h$ and $\Delta hi = ta_h - tp_i$ are calculated but not $\Delta jh = ta_j - tp_h$ or $\Delta hj = ta_h - tp_j$); neighbourhood is defined as in the Porta & Celaya model: the middle left and right legs are not considered neighbours, i.e. each leg has exactly two neighbours (see section 4.1.2 and [100]). The calculation of Δij depends on whether the legs are protracting or retracting. Inserting equations 4.1 - 4.4 in equation 4.5, one obtains:

1. if both legs i and j are retracting:

$$\begin{aligned}\Delta ij &= ta_i - tp_j \\ &= \left(\frac{aep_i - pep_i}{v_p} + \frac{x_i - pep_i}{v_r}\right) - \frac{x_j - pep_j}{v_r}\end{aligned} \tag{4.6}$$

2. if leg i is retracting and leg j is protracting:

$$\begin{aligned}\Delta ij &= ta_i - tp_j \\ &= \left(\frac{aep_i - pep_i}{v_p} + \frac{x_i - pep_i}{v_r}\right) \\ &\quad - \left(\frac{aep_j - x_j}{v_p} + \frac{aep_j - pep_j}{v_r}\right)\end{aligned} \tag{4.7}$$

3. if leg i is protracting and leg j is retracting:

$$\Delta ij = ta_i - tp_j = \frac{aep_i - x_i}{v_p} - \frac{x_j - pep_j}{v_r} \tag{4.8}$$

4. The case that both legs i and j protract simultaneously is not considered, because it should not occur, if the controller is operating appropriately.

4.2.5 Multiple Computations

The equations 4.6 to 4.8 as formulated above can be used to calculate Δij. They can however be reorganised to calculate any other variable that occurs in these equations. Thus the model could control any variable that occurs in its equations.

In order to decrease the number of free variables, the following assumptions were made. The protraction velocity v_p is assumed to be constant. The retraction velocity v_r could be variable, but was determined separately (see section 4.2.7). The extreme positions aep_i and aep_j could be controlled by the model, but in our setup they were determined by a targeting mechanism that was also used with the other models tested (see section 4.2.7). The current positions of the legs, x_i and x_j, are given values. Thus, only Δij, ta_i, tp_j, pep_i and pep_j remain as variables that are to be calculated by the MMC model.

For the MMC principle to apply, multiple ways to compute each variable are required. The complete set of all equations used in this model is documented in appendix A. The equations for each variable depend on the actual protraction/retraction states of the according legs, and have to be determined by reorganising equations 4.6 - 4.8. For example, the PEP of leg j, pep_j, can be calculated by the following two equations, if leg i is protracting and leg j retracting (base equation 4.8):

$$pep_j = \left(\Delta ij - \frac{aep_i - x_i}{v_p}\right) v_r + x_j$$

$$pep_j = (\Delta ij - ta_i)\, v_r + x_j$$

pep_jin turn is used to calculate pep_i: Reorganise equation 4.6 or 4.7. Since pep_i does not occur in equation 4.8, pep_j does not influence pep_i through this equations when leg i is protracting and leg j is retracting. In that case pep_i is (among other relations) determined by equation 4.7 with indices swapped for the calculation of Δji:

$$\begin{aligned} \Delta ji &= ta_j - tp_i \\ &= \left(\frac{aep_j - pep_j}{v_p} + \frac{x_j - pep_j}{v_r}\right) \\ &\quad - \left(\frac{aep_i - x_i}{v_p} + \frac{aep_i - pep_i}{v_r}\right) \end{aligned} \tag{4.9}$$

. Further, similar equations for computing pep_j (and all other variables) are available by swapping indices accordingly for the calculation of Δji in equations 4.6 and 4.8 and from the relations of leg j to its other adjacent leg h: Δjh and Δhj (swap indices accordingly in equations 4.5 - 4.8). Thus pep_j also enters the calculation of pep_h – and vice versa. That means each variable shows up in the equation set of three legs. The actual number of equations for each variable depends on the protraction/retraction states of the according legs, but redundancy for each variable occurs in all possible combinations. Please refer to appendix A for the complete set of equations.

This set of equations allows to describe the time of occurrence and position of AEPs and PEPs from any given starting position. Furthermore the delays between AEPs and PEPs of neighbouring legs are given.

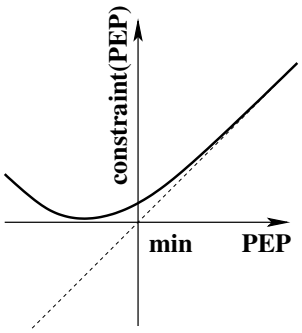

Figure 4.4: Constraint function of exponential extremes. Coordinate origin is min/min. See main text.

4.2.6 Constraints

As long as no constraints are introduced, the MMC model will relax to the closest consistent parameter set without respect to for example kinematic stability of the walker. Stability can be ensured in most cases if two neighbouring legs are prevented from protracting at the same time. Using the MMC model this can be achieved by constraining Δij values: A negative Δij implies that the AEP in leg i occurs before the pep of leg j - leg j begins protraction after leg i finishes it. Thus Δij values are constrained to be negative in most situations (see below).

These high level constraints on Δij values implement the actual coordination between legs. Physical limitations concern the positions of peps. The peps must lie within reach of the leg and – retraction being directed backwards – should be behind the current position of the leg.

To make the model fulfil these requirements, hard wired constraints and other mechanisms were implemented. With these constraints and mechanisms, the MMC model should consistently produce coordination parameters, that lead to a stable gait of the walker even in difficult situations. Different types of constraints were plugged into the recurrence loop of the MMC-network for the PEPs and the Δij variables as described in detail below.

The range of possible PEP-values is determined by anterior and posterior extreme values. The anterior extreme is the current position x of the according leg. The posterior extreme of the PEP range is a fixed value determined by leg lengths:

Range of anterior extremes of PEP:
"Hard" limits are used. If the mean value of all computations for a variable exceeds this limit, the value fed back to the equations is set to that limit.

Range of posterior extremes of PEP:
Soft limits formed by exponential functions are used: the constrained PEP value cannot fall below its physical posterior extreme (min), only converge to it: $constraint(PEP) = PEP + e^{-k(PEP-min)}$ (see fig 4.4), where k is an appropriately chosen constant. This approach yields a system that can be regarded as a system of pseudo springs, which looks for a good compromise solution [76].

The constraints on the Δij values implement the actual coordination mechanism for testing the model. It forced Δij to be < 0, unless

1. both legs i and j were in stance mode, and
2. the condition $(x_i - pep_i) > (x_j - pep_j)$ applied.

A negative Δij implies $ta_i < tp_j$, thus leg i reaches its AEP before leg j reaches its PEP (Fig. 4.3). If this condition is always

enforced, adjacent legs are prevented from swinging at the same time.

For Δij values a "safe extreme" is used. "Safe extremes" work much like hard limits with an additional safety margin: Δijs are set to -7 time units if they exceed -7 (compare to tab. 4.1, protraction/retraction times). As the iteration cycles of the MMC network proceed, the safety margin is decreased, until the hard limit -2 is hit – if the network does not relax beforehand. This "melting" of the safety margin is active from iteration 100 until iteration 300.

4.2.7 Retraction velocity

The protraction velocity is kept constant at $v_p = 1$. The retraction velocity v_r is set to the default value v_{rref} (see section 4.3.3), before starting to iterate the MMC net. This value is maintained fixed during the first 400 iterations. v_r is then decreased as iterations proceeded, until the lower limit $0.1v_{rref}$ is hit - if the network does not relax beforehand. v_r is decreased during iterations 400 to 900.

If the network does not relax after another 100 iterations (1000 cycles total), iteration is stopped. Whatever values are assigned to the variables at that point are used for coordination. The global reference retraction velocity (*vRef* in Fig. 4.6) is $0.1v_{rref}$ in that case.

4.3 Experimental Testbed

The MMC model for coordination was tested on a simulation of a six legged walker and its performance was compared to three other coordination schemes.

front leg length	32.9
middle leg length	25.6
Δmiddle/hind legs	11.3
hind leg length	29.0
body length	27.1
body width	1.0
mean body height	≈ 6.5
mean step size	≈ 20.0
v_p	≈ 1.0
mean t_p	≈ 20.0
mean t_r	$\approx \frac{t_p}{v_r}$

Table 4.1: Geometrical measures in distance units and iteration cycles of the simulated six legged walker I used for testing the models. $\approx$ indicates approximate equality. Leg lengths is given for fully extended legs from shoulder joint to the tip of the tibia. Δ middle/hind legs is the distance from middle shoulder joint position to hind shoulder joint position. Body length is distance from front to hind shoulder joint positions, width is distance from left to right shoulder joint positions (see Fig. 4.5). v_p is protraction velocity, t_p is protraction time. The mean retraction time t_r depends on the retraction velocity v_r which is varied between experiments. Protraction/retraction trajectory lengths (step size) and velocities and other mean values vary during walking and are only approximations.

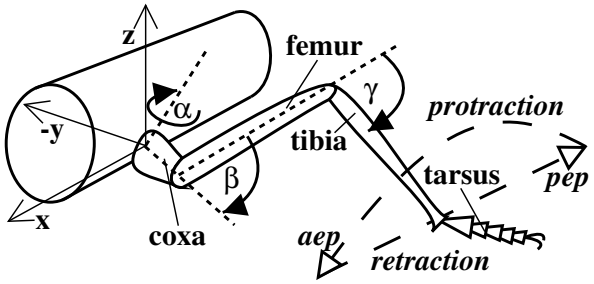

Figure 4.5: The figure illustrates an insect leg and denotes the angular and coordinate conventions used in this paper. Protraction and retraction trajectories are schematically shown by dashed arrows.

4.3.1 Controllers and Simulation

Since the four models discussed here (Cruse's coordination rules ([27] and section 4.1.2), ThreshPorta and StabPorta ([100] and section 4.1.2) and the MMC model described above (section 4.2)) only control coordination, additional systems are required to determine the AEPs, and to generate protraction and retraction trajectories.

To control retraction velocity, the retraction trajectory controller compares the actual retraction velocity to a reference velocity (see Fig. 4.6). In the Porta & Celaya models and the MMC model, the retraction velocities are calculated by these models and these values are entered into the retraction trajectory controller as the reference velocity. Since the Cruse coordination mechanisms do not influence retraction velocities, fixed default velocities v_{rref} as defined in section 4.3.3 are used as reference velocities.

Protraction and retraction trajectories are generated by analytical kinematic modules (see chapter 2), i.e. delta angles are calculated from the current angles and from additional information (target angles in swing or delta z (=height) and delta x (=velocity) in

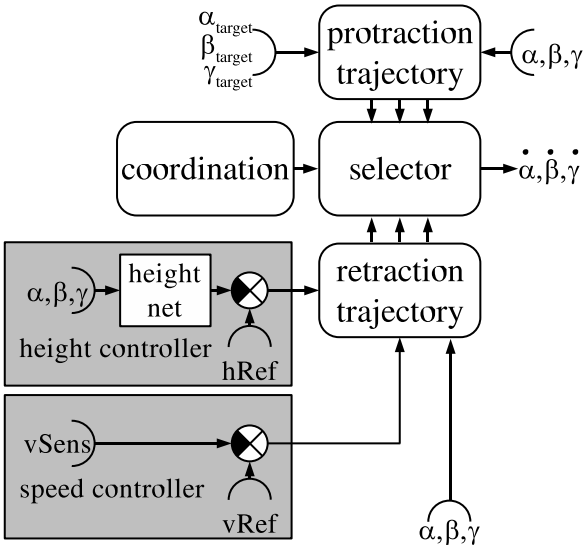

Figure 4.6: Common controller components used with all coordination models. For angle conventions see Fig. 4.5. vSens is the sensed velocity, hRef the reference height.

stance). The AEPs are determined by a targeting mechanism that is used for the generation of protraction trajectories (see Fig. 4.6): Hind and middle legs are aiming at the current position of the according anterior leg, the AEPs of the front legs are constant. This method was first introduced by Cruse [20] as a fourth coordination mechanism, and is justified through ethological studies on stick insects [20]. Furthermore, legs that hit the cliff of an obstacle during protraction perform an avoidance reflex [38]: the leg moves backward and up before continuing it forward motion.

To simulate the kinematics of the walker, an MMC network is used which adopts a completely different approach than the MMC network implementing the internal coordination model [76]. The kinematic simulation is embedded in a simple environment simulation, that allows walking over plane ground and obstacles.

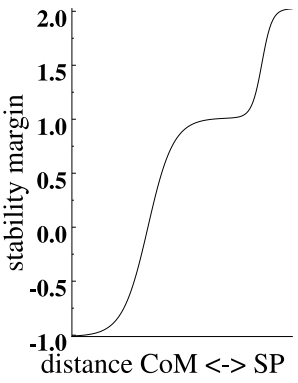

Figure 4.7: The stability characteristic for three or more ground contact points (CoM = centre of mass, SP = stability polygon); see text for details.

4.3.2 Stability margin

For evaluating the performance of the four coordination models I introduce the "stability margin" as a measure. The stability margin is based on the stability polygon [9, 85, 102] and a special characteristic.

The stability polygon is the convex polygon around all ground contact points. A ground contact point is determined by a leg in retraction mode touching the ground. The centre of mass is assumed to lie between the middle leg coxae, projected into the ground plane (the plane that is orthogonal to the gravity vector). The distance of the centre of mass to the closest point of the stability polygon is measured. If the centre of mass lies inside the polygon, the distance value is set to be positive, otherwise it is set to be negative. If the centre of mass lies exactly on a corner or edge of the stability polygon, the distance is zero. [85] defines a similar concept, the longitudinal stability margin. For the longitudinal stability margin the distance of the centre of mass to the boundaries of the support polygon is only calculated in the direction of motion of the centre of mass. The stability margin used here gives a better measure since it calculates the smallest distance of the centre of mass to any boundary of the support polygon.

This distance value is mapped by a function that enhances resolution in critical situations and lowers resolution in "uninteresting" situations (Fig. 4.7). If all legs are fully extended and have ground contact, the distance reaches a maximum which is mapped to a stability margin of 2 by the stability characteristic. The mean distance during walking on a plane is mapped to show a stability margin of 1. The minimum (i.e. the greatest negative) distance possible with three ground contact points is mapped to -1.

The following extreme cases are not depicted in Fig. 4.7: When only two legs have ground contact, the stability margin is de-

termined by a linear function that maps the distance of the centre of mass to the line connecting the two ground contact points, to lie between -1 (if the centre of mass lies exactly between the two ground contact points) and -2. When only one leg has ground contact, the distance of the ground contact point to the centre of mass is linearly mapped to lie between -2 and -3. If no legs have ground contact, the stability margin is set to -4.

The stability margin is recorded along with the walker's body position (x/y position of point between the bases of the hind coxae, see Fig. 4.5) for each simulation iteration of each run.

4.3.3 The Tasks

4.3.3.1 Climbing an Obstacle

The four controllers (Cruse, ThreshPorta, StabPorta and the MMC model) are each tested on the simulation of a six legged walker. The task for all controllers is to cross an obstacle that is approximately as high as the shoulder joints (coxae, see Fig. 4.5) of the walker. It consists of an upward step and a downward step that are separated by a distance of 100 length units (this corresponds to about 4-5 normal walker steps, see tab. 4.1). For each default retraction velocity (see below) 10 trials are run. For each trial, the x-position of the obstacle is shifted by 2 length units. This has the effect that the legs make first contact with the obstacle at different points in their duty cycle, introducing randomness into the experiments.

The default retraction velocity v_r is varied systematical from $v_r = 0.05v_p$ to $v_r = 0.6v_p$ in steps of $0.05v_p$. This means that the duration of retraction is between 20 and 1.67 times longer than the duration of protraction. Thus 12 default velocities are tested, with 10 trials with shifted x-positions of the obstacle each, summing to a total of 120 trials for each controller type. Since the walker takes obviously longer to cross the obstacle at lower velocities, the number of controller/simulation iterations for each trial depended on v_r.

4.3.3.2 Curve Walking

Since the performance of the two versions of the Porta & Celaya model can not be discriminated in the obstacle climbing setup an additional simple experiment is applied (both remain statically stable at all times; mean stability of ThreshPorta variant is higher, but it turns out to be inferior to StabPorta variant in the additional experiment): The walker has to run on plane ground with default $v_r = 0.7v_p$ in a circle with a diameter of about 3–4 body lengths. When walking in a circle with such a diameter, the retraction trajectories of all legs are still directed mostly backward, but not necessarily parrallel to the body long axis. Since all models use x-coordinates of tarsi (in local coordinate systems – see beginning of section 4.1.2), curve walking should pose no problems as a matter of principle.

4.3.4 Processing Time

In order to evaluate the computational complexity of the models, each model was run separately without any other controller parts and without the kinematic MMC simulation (see section 4.3.1). Inputs required by each model was supplied by fixed reference values which did not change during the whole test, i.e. the walker did not move at all. For the tests, parameters of a reasonable walker configuration were chosen where all six legs had ground contact.

Two tests were run for each model. One test for each model lasted zero iterations. This is required to account for the time needed for module initialisation. The other test lasted one billion iterations for the ThreshPorta and Cruse models, 10,000 iterations for the StabPorta model and 1000 iterations for the MMC model. The numbers of iterations were chosen to make the tests significantly longer than initialisation time while keeping all tests in a reasonable time frame (10 to 30 seconds).

All tests were run on Pentium IV system with 2.66 GHz clock cycle. The operating system was Linux version 2.4.22 (gcc version 3.3.2).

4.4 Results

4.4.1 Crossing an Obstacle

In these tests lock-up situations occurred with all models at times. With all models, lock-ups could for example occur when a protracting leg hit the cliff of the obstacle and perpetually performed an avoidance reflex, without ever reaching the top of the obstacle. Since lock-ups occurred rarely and with all models, such trials were excluded from the results. For a discussion on avoiding lock-ups see e.g. [85, 94].

The mean stability margin for the different default velocities was determined for all four models in three distinct situations: While ascending the obstacle, while descending from the obstacle and while walking on plane horizontal ground some distance behind the obstacle (Fig. 4.8). All measurements from about one step before the walker reaches the obstacle, until it is well above it, were pooled in the "ascend" data. The "descend" data were pooled accordingly. The

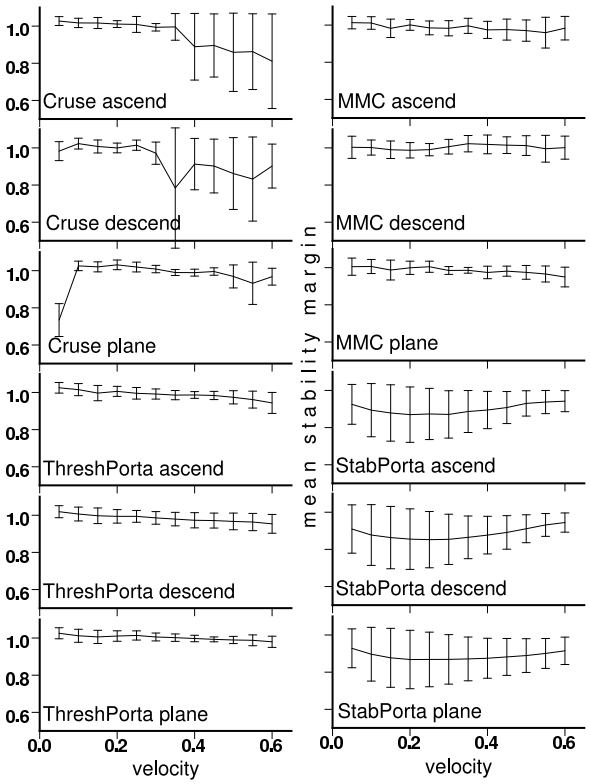

Figure 4.8: Mean stabilities for all four models, while ascending the obstacle, descending from the obstacle and walking on plane ground at different default velocities. The standard errors of the mean values are given as error bars.

"plane" data were pooled from an equally sized stretch behind the obstacle. Mean values and standard errors are presented in Fig. 4.8.

All models except the StabPorta model tend to perform better at lower default velocities in all situations. The StabPorta model shows the worst mean performance. The Cruse model also shows performance problems when ascending or descending the obstacle. With the exception of the Cruse model all models show roughly the same mean stability in different walking situation. However, in all cases the mean stability was positive.

Considering the performance of coordination models, apart from considering the mean stability, it is useful to know how likely they are to fail when performing a given task. Toppling of the walker is a good indicator for failure. In real world the walker would topple when it becomes instable. Thus the probability of becoming instable when performing a given task was also determined (see Fig. 4.9). For each run data were pooled separately into three tasks ("ascend", "descend", "plane" according to mean stability calculation above). If the stability margin became negative (i.e. the walker became instable and would topple in real world) for at least one iteration cycle during one task of one run, that task was marked as "failed". Each task was performed 10 times for each model and each default retraction velocity. If the model failed a task in one out of ten trials at a given default retraction velocity, a probability of 0.1 results for that model to fail the task at that default retraction velocity.

The Cruse model only performs reliably when walking slowly on plane ground. Its performance worsens with higher walking speeds. The MMC model has problems with

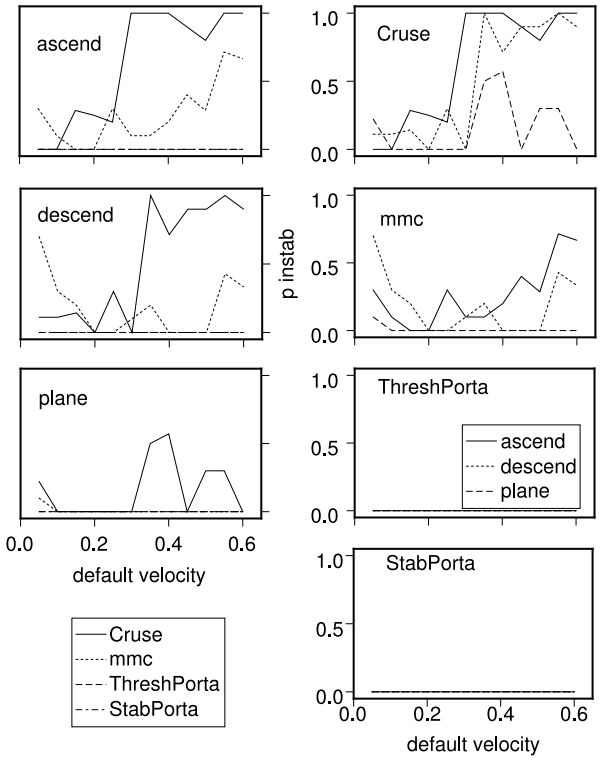

Figure 4.9: Probability of becoming instable while performing different tasks. The plots in the left and right column visualise the same data, grouped differently. Thus the performance of different models in the same situation (left), and for one model in different situations (right), can be better compared.

ascending or descending the obstacle, particularly at low and high velocities. Neither variant of the Porta & Celaya model ever became instable.

The data plotted in Fig. 4.9 were also used in Fig. 4.10 (left), but now showing on the abscissa the spatial position of the walker relative to the obstacle (x-positions were measured at hind left shoulder joints). For that figure the x-coordinates for each walk were plotted in such a way that the obstacle always begins at position zero and ends at position 100 (abscissa). Each run was then partitioned into x-position bins (bin size 10 length units - about half a mean step - see tab. 4.1 and Fig. 4.5). The left side of Fig. 4.10 shows the probability for the walker to become instable (compare to Fig. 4.9 left).

If the stability became negative at any point in a bin, that bin was assigned value 1.0, otherwise 0.0. For each default velocity each bin was set to the arithmetic mean of all 10 runs of that velocity. The resulting 2D-distributions were smoothed with a Gaussian-filter of width 10 bins (with an x-position bin size of 10 length units this corresponds to 100 units in Fig. 4.10, about 5 mean steps or three to four walker lengths), and then used in the plots (Fig. 4.10). No smoothing was done in y-direction.

Note that the walker position was measured between the hind coxae (see end of section 4.3.2). Thus instabilities can occur at position before the obstacle (when the front legs are climbing the obstacle).

Fig. 4.10, right shows the same stability data with a different criterion applied: a value of 1.0 was assigned where the stability reached values $<$ -1.0, i.e. where the walker has ground contact with only two or less legs (otherwise 0.0 was assigned). Having two or less legs on the ground is here defined as a coordination error. Fig. 4.10 right thus shows the probability to make coordination errors.

The performance of both variants of the Porta & Celaya model is far better than that of the other models under this measure. Both Porta & Celaya models did not make any coordination errors in these tests, and retained stability all the time. With MMC and Cruse model, instabilities mostly occur while ascending the obstacle and while descending it.

The MMC model also has some problems at low velocities while the Cruse model tends to fail at high velocities. The performance of the MMC model is much better than that of the Cruse model, but not quite as good as both variants of the Porta & Celaya model.

Both with the Cruse and with the MMC model instabilities seem to cluster at certain positions in relation to the obstacle and independently from phase and velocity. The first such cluster occurs when the front legs hit the obstacle upward cliff. The second cluster occurs when the middle legs hit the obstacle upward cliff. The third cluster occurs when the front legs step into the gap after the downward cliff, the fourth and last cluster occurs when only the hind legs remain on the obstacle. With the MMC model the second and third clusters are very week and do not occur at all retraction velocities.

4.4.2 Curve Walking

To discriminate between the two variants of the Porta & Celaya model another experiment was applied: Walking in circles on plane ground at $v_{rref} = 0.7v_p$. Qualitative inspection shows the StabPorta variant which uses stability evaluation to determine PEPs to perform far superior as compared to the ThreshPorta variant. Fig. 4.11 shows the results of this experiment. The

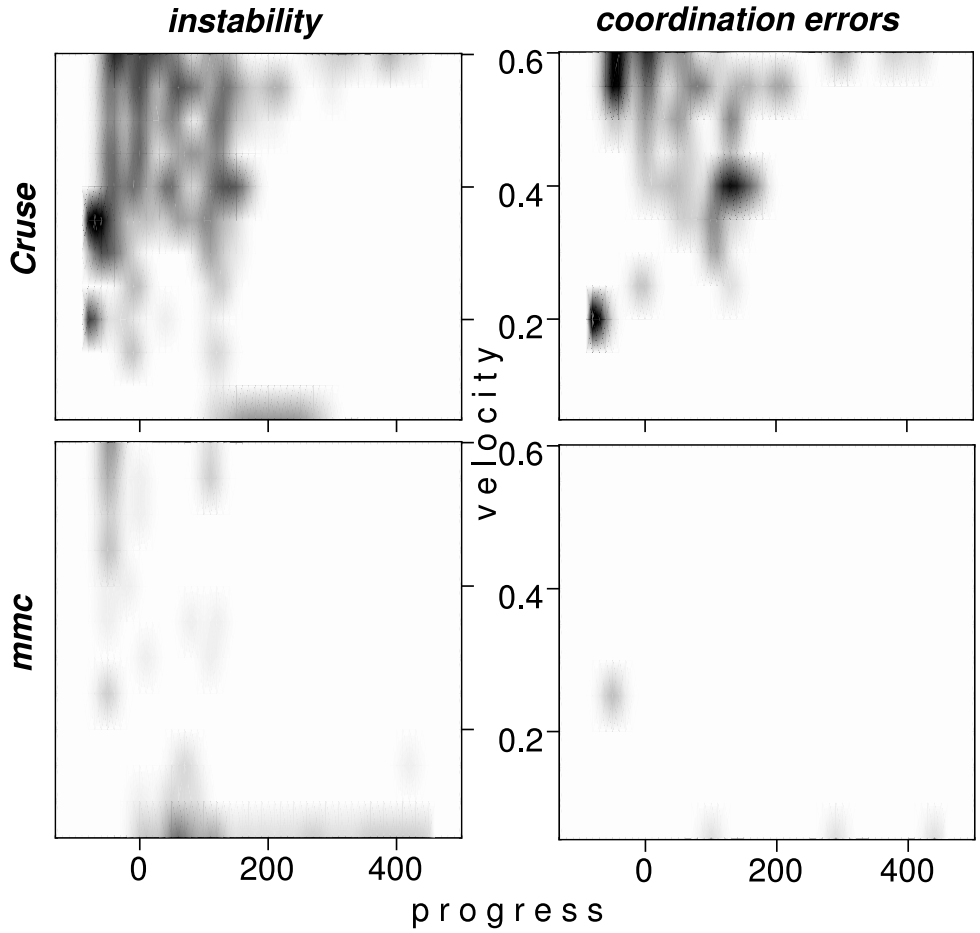

Figure 4.10: Probability for the walker to topple over (left), or make coordination errors (right). The upper plots show the performance of the Cruse model and the lower showsthe performance of the MMC model. The results for the two variants of the Porta & Celaya model are not displayed since they did not make any errors. The x-position of the walker is denoted on the abscissa. The obstacle begins at position 0 and ends at 100. The default velocity is denoted on the ordinate. Grey level shows the probability of becoming instable (left: white = 0, black = 0.5), or making coordination errors (right: white = 0, black = 0.25). Body length corresponds to about 27 units (see table 4.1).

Cruse	$2.5 * 10^{-5}$
ThreshPorta	$1.7 * 10^{-5}$
StabPorta	$0.84 * 10^{-3}$
MMC	$2.7 * 10^{-2}$

Table 4.2: This table gives the time in seconds each model needs to complete one iteration.

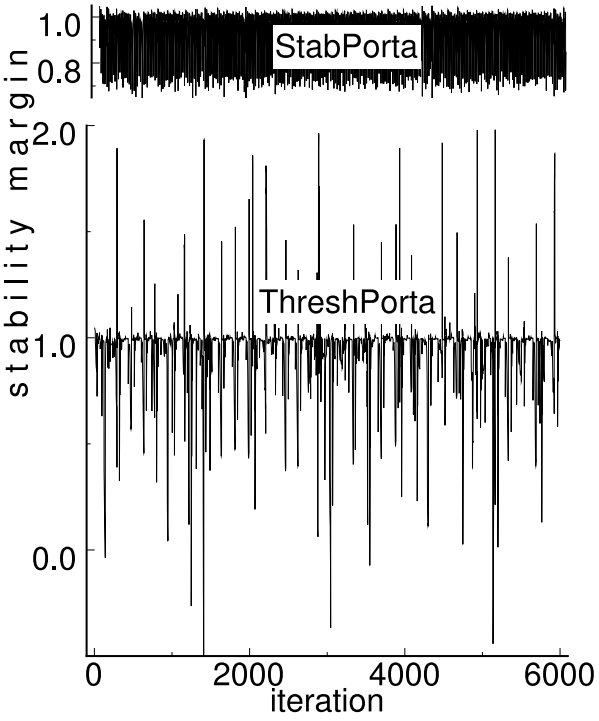

Figure 4.11: Stability margin for the StabPorta (top) and ThreshPorta (bottom) models while running in circles for 6000 iterations (one trial shown).

ThreshPorta variant was instable during 19 of 6000 iterations, while the StabPorta variant never became instable. Yet the mean stability was about the same in both cases (0.93) – compare this result to Fig. 4.8 where the StabPorta variant is shown to have lower mean stabilities than the ThreshPorta variant without ever becoming instable (Fig. 4.9). The StabPorta variant keeps the stability margin in a relatively narrow band as compared to the simpler variant. It also achieves a much higher actual walking speed (mean $0.62v_p$) than the ThreshPorta variant (mean $0.33v_p$).

Curve walking was also tested with the other models. The MMC model produced instable walker configurations in 64 out of 6000 iterations, the Cruse rules 1065. Mean retraction velocity was $0.53v_p$ with the MMC model and $0.69v_p$ with the Cruse rules (note that the Cruse rules do not control retraction velocity). It is interesting to note that with the MMC model all instances of instable walker configurations occurred during the first 1400 iterations while they were evenly distributed over all 6000 iterations with the ThreshPorta model and the Cruse rules. This indicates that the MMC model can reliably solve the curve walking problem once it falls into an appropriate "rhythm".

4.4.3 Processing Time

The different values for processing time are given in table 4.2. The ThreshPorta model is the fastest. The Cruse model takes a bit longer to compute but is still in the same order of magnitude as the ThreshPorta model. The StabPorta model takes two orders of magnitude longer to compute and the MMC model takes another order of magnitude longer to compute.

4.5 Discussion

4.5.1 Model Comparison

The extended versions of the Porta & Celaya model yield the best results. Compared to these versions, the original variant as published by Porta & Celaya [100] is considerably less modular and less reliable. However, with the extensions proposed here (StabPorta variant with stability evaluation) it presents a reliable yet still simple and efficient solution. In further experiments similar to the ones described here, I determined the impact of important concepts of these models on performance. Both Porta & Celaya models were tested with/without the optimisations proposed in [100] and with/without reference velocity control. With both features turned off both models still outperform the Cruse rules and the MMC model. The optimisation has a minor yet measurable effect on static walker stability, the reference velocity control has a major effect. The optimisation yields greater walking speeds - as claimed by Porta & Celaya [100].

The performance of the new MMC model by far exceeded that of the Cruse model. This result was acquired with quite simple, hard wired, hand tuned constraints, while the Cruse mechanisms used long tested characteristics and parameters that were optimised using genetic algorithms. This comparison is not really fair though, because the Cruse model does not control retraction velocity. In the introduction (section 4.1.1) I defined coordination as the problem of determining the positions of AEPs and PEPs. The results indicate that the determination of an appropriate walking speed should be considered another important aspect of the coordination problem.

The MMC model was outperformed by the Porta & Celaya variants. This was unexpected because the MMC model has the capability to look ahead for one complete step – a feature that sets it apart from the other models. Certain aspects of the MMC model might be improved, but only by introducing more complexity. Everything that could possibly be achieved by the MMC model is already achieved by the two variants of the Porta & Celaya model tested here.

The StabPorta model which showed the best over all performance is prone to a special kind of lock-up: When a hind leg is near its physical PEP and the neighbouring hind and middle legs are close to their AEP, the hind leg cannot be lifted without loosing stability. To cope with all possible lock-ups purely reactive models will probably not do. One has to either plan ahead quite some time into the future (see e.g. [94]), or change tactics when the lock-up occurs. If the latter approach is to be taken the system has to be quite flexible and be able to try many different tactics – e.g. moving sideways or stepping back and changing stepping order and so on.

4.5.2 Model Complexity

The results concerning the computational complexity should be regarded qualitative results rather than quantitative, since the implementation of the models was not at all focused on execution speed. The ThreshPorta and Cruse models should be considered to be of roughly equal complexity, the ThreshPorta variant being slightly more simple. Both the StabPorta model and the MMC model are much more complex, the MMC model probably being the most complex.

Obviously the MMC model has to iterate until it relaxes. For each controller iteration, the MMC net has to compute dozens of equations for tens or even hundreds of iterations. The processing time for the MMC model varies with the number of iterations. The model tends to take longer the more restricted a situation is. This is not reflected in the processing time tests.

The StabPorta variant also requires iterative processes for determining the convex polygon from the cloud of (ground-contact-) points. The C++ code dealing with the geometric and kinematic calculations in the implementation of the StabPorta model, which was used in the tests presented here, are particularly inefficient. The code also performs more stability calculations than are strictly required for the StabPorta model, because the code is designed to be of general use. In an implementation focused on execution speed the StabPorta model would become exponentially faster with each lifted leg. The StabPorta variant requires all code from the ThreshPorta variant be executed too. But compared to the StabPorta model the execution time for ThreshPorta code is insignificant.

The Cruse model has to perform a few simple calculations for each retracting leg. Coordination rule 1 is a simple product, coordination rule 2 consists of a delay, two low pass filters and two products in the implementation used in this test and coordination rule 3 consists if four thresholds and a couple of sums and products.

The ThreshPorta model only has to compare three values for each retracting leg and determine the lowest value from all retracting legs to determine the retraction velocity with one multiplication.

In an implementation focused on execution speed both the Cruse model and the ThreshPorta model scale linearly with the number of retracting legs.

4.6 Conclusions

Several points could be made in this chapter. It could be demonstrated that retraction velocity control is an essential part of leg coordination. A novel application of the MMC principle on coordination is introduced and demonstrated to perform more or less reliably. Two coordination models were shown to excel in the coordination model comparison performed in this chapter: the ThreshPorta variant is the simplest model tested and performs reliably under most conditions. The StabPorta variant is still relatively simple (an inefficient implementation taking a millisecond to compute on a contemporary personal computer) while performing reliably in all tested situations.

I thank the Deutsche Forschungsgesellschaft for funding this work by the graduate program Strukturbildungsprozesse (grant number GK 231) and DFG grant no. Cr58/9-3.

Appendix

A MMC Equations

This appendix contains all equations that define the MMC model for coordination. The equations listed below are all equations for relating leg i and leg j (indices).

A.1 Base equations

The following four equations are base equations that occur in most other equations. Index r indicates that the respective leg is retracting, index p indicates it is protracting. For saving space I only list the equations. The concerning phase combinations can be determined from the indices r and/or p.

$$ta_{ir}() = \frac{aep_i - pep_i}{v_p} + \frac{x_i - pep_i}{v_r} \quad (4.10)$$

$$ta_{ip}() = \frac{aep_i - x_i}{v_p} \quad (4.11)$$

$$tp_{jr}() = \frac{x_j - pep_j}{v_r} \quad (4.12)$$

$$tp_{jp}() = \frac{aep_j - x_j}{v_p} + \frac{aep_j - pep_j}{v_r} \quad (4.13)$$

A.2 Δij

Base relation between legs i and j (all phase combinations):

$$\Delta ij = ta_i - tp_j \quad (4.14)$$

Inserting equations 4.10 to 4.13 in equation 4.14 for *ta and/or tp* (according to phase constellation, see indices p and r):

$$\Delta ij = ta_{ir}() - tp_{jr}() \quad (4.15)$$

$$\Delta ij = ta_{ir}() - tp_{jp}() \quad (4.16)$$

$$\Delta ij = ta_{ip}() - tp_{jr}() \quad (4.17)$$

$$\Delta ij = ta_{ir}() - tp_j \quad (4.18)$$

$$\Delta ij = ta_{ip}() - tp_j \quad (4.19)$$

$$\Delta ij = ta_i - tp_{jr}() \quad (4.20)$$

$$\Delta ij = ta_i - tp_{jp}() \quad (4.21)$$

All of the following equations were deduced by reorganising the above equations.

A.3 ta and tp

ta and tp can be calculated from the base equations above. They are also stored in variables that are used directly in some equations. More equations for ta and tp result from reorganising the Δij equations above:

$ta_i = \Delta ij - tp_j$ from 4.14
$ta_i = \Delta ij - tp_{jr}()$ from 4.20
$ta_i = \Delta ij - tp_{jp}()$ from 4.21
$ta_i = ta_{ir}()$ from 4.10
$ta_i = ta_{ip}()$ from 4.11
$tp_j = ta_i - \Delta ij$ from 4.14
$tp_j = ta_{ir}() - \Delta ij$ from 4.18
$tp_j = ta_{ip}() - \Delta ij$ from 4.19
$tp_j = tp_{jr}()$ from 4.12
$tp_j = tp_{jp}()$ from 4.13

A.4 pep_i

To simplify the equations the following helper function is used (for leg i in retraction). From 4.18 (with 4.10 inserted; 4.11 does not contain pep_i):

$$h_{ipr}(tp) = \frac{aep_i v_r + x_i v_p - (\Delta ij + tp)\, v_r v_p}{v_r + v_p} \tag{4.22}$$

pep_i does not occur in equations where leg i is in protraction. Thus no equations for pep_i are listed for the according phase combinations.

Inserting the established expressions for tp_j we get:

$pep_i = h_{ipr}(tp_{jr}())$ 4.12 in 4.22
$pep_i = h_{ipr}(tp_{jp}())$ 4.13 in 4.22
$pep_i = h_{ipr}(tp_j)$ 4.22

A.5 pep_j

Two helper functions (one for leg j in protraction and one for leg j retraction) are used for the calculation of pep_j. These are derived from 4.13 inserted in 4.21 (4.23) and 4.12 inserted in 4.20 (4.24):

$$h_{jpp}(ta) = (\Delta ij - ta + (aep_j - x_j)\, v_p)\, v_r + x_j \tag{4.23}$$

$$h_{jpr}(ta) = (\Delta ij - ta)\, v_r + x_j \tag{4.24}$$

Inserting established expressions for ta_i we get:

$pep_j = h_{jpr}(ta_{ir}())$ 4.10 in 4.24
$pep_j = h_{jpp}(ta_{ir}())$4.10 in 4.23
$pep_j = h_{jpr}(ta_{ip}())$4.11 in 4.24
$pep_j = h_{jpepp}(ta_i)$ 4.23
$pep_j = h_{jpepr}(ta_i)$ 4.24

B aeps and v_r

Equations for calculating aeps and v_r are a logical part of the model therefore they are also listed here. They were implemented and tested. They were however not used in the experiments presented in this chapter.

B.1 aep_i

Two helper functions (see equations for calculation of peps) are used for the calculation of aep_i, derived from 4.11 inserted in 4.19 (4.25) and 4.10 inserted in 4.18 (4.26):

$$h_{iap}(tp) = (\Delta ij + tp)\, v_p + x_i \tag{4.25}$$

$$h_{iar}(tp) = \left(\Delta ij + tp - \frac{x_i - pep_i}{v_r}\right) v_p + pep_i \tag{4.26}$$

Inserting established expressions for tp_j we get:

$aep_i = h_{iap}(tp_{jr}())$ 4.12 in 4.25
$aep_i = h_{iar}(tp_{jp}())$ 4.13 in 4.26
$aep_i = h_{iar}(tp_{jr}())$ 4.12 in 4.26
$aep_i = h_{iap}(tp_j)$ 4.25
$aep_i = h_{iar}(tp_j)$ 4.26

B.2 aep_j

One helper function is used (for leg i in retraction) for calculating aep_j, derived from 4.13 in 4.21:

$$h_{jar}(ta) = \frac{(ta - \Delta ij)\, v_r v_p + x_j v_r + pep_j v_p}{v_r + v_p} \tag{4.27}$$

aep_j does not occur in equations where leg i is in protraction. Thus no equations for aep_j are listed for the according phase combinations.

Inserting the established expressions for ta_i we get:

$aep_j = h_{jar}(ta_{ir}())$ 4.10 in 4.27
$aep_j = h_{jar}(ta_i)$ 4.27

B.3 v_r

Since no retraction/protraction indices occur in the equations for calculating v_r the

according phase combinations are given for once:

leg i protracting / leg j retracting

- $v_r = \frac{pep_j - x_j}{\Delta ij - ta_i}$ 4.12 in 4.20
- $v_r = \frac{(pep_j - x_j)v_p}{\Delta ij v_p - aep_i + x_i}$ 4.11, 4.12 in 4.17

leg i retracting / leg j protracting

- $v_r = \frac{(x_i - pep_i)v_p}{(\Delta ij + tp_j)v_p - aep_i + pep_i}$ 4.10 in 4.18
- $v_r = -\frac{(aep_j - pep_j)v_p}{(\Delta ij - ta_i)v_p + aep_j - x_j}$ 4.13 in 4.21
- $v_r = \frac{(x_i - pep_i - aep_j + pep_j)v_p}{\Delta ij v_p - aep_i + pep_i + aep_j - x_j}$ 4.10, in 4.13 in 4.16

leg i retracting / leg j retracting

- $v_r = \frac{(x_i - pep_i)v_p}{(\Delta ij + tp_j)v_p - aep_i + pep_i}$ 4.10 in 4.18
- $v_r = -\frac{pep_j - x_j}{\Delta ij - ta_i}$ 4.12 in 4.20
- $v_r = \frac{(x_i - pep_i - x_j + pep_j)v_p}{\Delta ij v_p - aep_i + pep_i}$ 4.10, 4.12 in 4.15

C More Redundancy

The above appendices list all equations resulting from $\Delta ij = ta_i - tp_j$. The inverse relation $\Delta ji = ta_j - tp_i$ yields another set of equations like the one above (to get this set of equations, swap all indices i and j). That doubles the number of ways each variable can be calculated. Still only the relations between legs i and j are accounted for. But all variables of leg leg j also occur in equations derived from the relation between j and its other neighbour, k: $\Delta jk = ta_j - tp_k$, $\Delta kj = ta_k - tp_j$ (to get the derived equations, swap indices accordingly). That means, the above appendices list one quarter of the equations that concern variables of one leg.

5 Discussion

In this thesis several concepts have been developed that allow to improve the control of a kinematically complex body. The properties of these newly developed systems will be discussed by describing their possible applications to future problems.

The systems presented in this thesis provide a viable base for a controller that combines low level reactive modules for hexapod walking (see chapter 2) with higher level modules for motion planning (see chapter 3). For the coordination problem the integration of the reactive controller with a module that performs implicit planning has already been demonstrated (see chapter 4).

5.1 Future Works

With this solid development base several possibilities for future research are simplified or opened.

5.1.1 Reactive Systems

One important aspect of stick insect walking has been completely ignored in Walknet studies until now – the role of the antennae. Biological studies by Krause and Dürr [78, 46] have provided sufficient data for integrating antennae into the Walknet simulations. The Walknet is however prone to complex interactions between its parts. When introducing new modules, unexpected effects that are very hard to interpret are frequently observed. The integration of antennae would be vastly simplified, if the development started with integrating the antennae with the Akin (analytical kinematic stance trajectory generation as proposed in section 2.2.1) controller and ThreshPorta coordination (section 4.1.2). Once the antennae simulation works as intended with these controllers, the various submodules can be replaced one by one by their Walknet equivalents.

The Ejoin controller – stance trajectory generation with local joint controllers as proposed in section 2.2.2 – can be used to test the advanced positive velocity feedback mechanisms proposed by Schneider [113, 112] and the integrated positive/negative feedback mechanisms proposed by Fischer and by Cruse et al. [51, 33]. Both approaches constitute models for the behaviour of standing or retracting stick insect legs.

Since such tests would only concern stance trajectory generation, instead of using original Walknet modules, it would be sensible to use the more reliable alternatives presented here – e.g. ThreshPorta for coordination as proposed in chapter 4 and kinematical targeting and swing trajectory generators based on analytical kinematics as proposed in chapter 2. This would considerably reduce the complexity of the development task. Positive feedback as proposed by Schneider only concerns the α and γ joints. Therefor the Akin (analytical kinematic stance trajectory generation as proposed in section 2.2.1) controller could still be used to control the β joints. Thus, the interference between retraction and height

control as observed in the Walknet (see chapter 1) could be minimized. As the development of alternative positive feedback mechanisms proceeds, the kinematic controllers could be replaced by Walknet variants once the development has stabilized.

Research of stick insects frequently leads to novel observations. Almost as frequently the models derived from these observations are tested in simulation studies. Several of these studies could profit from the development of the Akin and Ejoin controllers as explained in the examples above.

5.1.2 Posture MMC as Attitude Controller

The primary goal of the Akin development was to have a reliable and predictable controller which might be extended with planning modules, though. The integration of these reactive and "cognitive" modules could be approached in one of two ways: The Akin controller could be used for all tasks until problems are detected that cannot be solved by the reactive controller. The planning modules could then take over.

However, one of the most important arguments in Cruse's hypothesis about the evolution of cognition [31] was that cognitive properties must not necessarily have implied the – evolutionary – development of completely new modules. Instead, Cruse postulated that existing modules could be modified to use them for planning.

The posture MMC network as proposed in section 3.5.1 constitutes the first MMC network that could be used for control as well as for modelling. In the proposed form it optimizes the walker posture and could as such already be used to replace the stance modules of the Akin controller for all but the x-component, which controls the actual retraction. In terms of implementation, both systems – Akin controller and posture MMC – would run in parallel in such a system. The Akin controller would only be used to control retraction, the MMC network would be used to control all other degrees of freedom of the walker body.

Retraction can in principle also be controlled by the posture MMC (section 3.5.1). The system might not even have to be changed to achieve this goal. Porta and Celaya proposed an alternative posture optimization algorithm [102]. In contrast to the posture MMC the algorithm of Porta and Celaya does not stem from the connectionist approach, they do not explain if or how constraints could be introduced to their approach and it could not serve as a model for sensor fusion and sensory motor integration as the MMC network can. However, they show that their approach can still be used for controlling retraction: while walking, legs are continuously moved from PEP to AEP during the swing phases. Once these legs reached their anterior positions, the posture optimization takes these legs into account and shifts the body forward to optimize the posture, thus retracting the legs.

Since the MMC can replace the posture optimization approach of Porta and Celaya, it should be possible to use it for retraction in the same manner without having to modify the MMC. It might however be advantageous to shift the "optimal" posture backward for stance trajectory generation. Without such a shift the resulting gait pattern might dominantly use the anterior part of the leg work space. To achieve this shift with the posture MMC network, one would simply have to change the vectors defining the optimal posture ($\vec{O}$ in section 3.5.1) accordingly. Anticipative attitude and height

control could be implemented in a similar manner. In that case the actual target positions as determined by antennae or anterior legs are taken into account to determine the optimal posture in the anticipated situation.

When the posture optimization MMC network is thus embedded into the reactive part of the controller, it could not only serve as a first model for how motion planning could be achieved, it could then also account for the above mentioned argument of Cruse – namely that novel developments might not be necessary in all cases, when evolving planning capabilities [31].

Furthermore it could account for the empirical evidence that thinking is neurally similar to acting (see chapter 1 or [68, 53, 71]) and for the inhibitory structures that seem to block action execution while thinking (see chapter 1 or [106, 71, 59]).

5.1.3 Further Development of Coordination MMC

Several new mechanisms for influencing the dynamical behaviour of MMC nets were proposed in chapter 3. This work has been performed after developing the generalized MMC approach which dispenses with weight-matrix-based networks and after using that approach for the development of the coordination MMC (chapter 4). Therefore the coordination MMC network does not yet use these novel mechanisms.

Instead, customized characteristics were used to control various aspects of the dynamical behaviour of the network. Various "hard" and "soft" extreme values were imposed onto the system to constrain the leg workspace. The dynamical weighting of MMC variables as introduced in section 3.5.2 might constitute a better alternative to implement such constraints. That section (3.5.2) explicitly demonstrated how to implement constraints on MMC variables with dynamical weighting. Dynamical weighting would work similarly to the "soft extreme" proposed in section 4.2.6 and could be used to replace it.

To keep the dynamic behaviour of the network under control in all situations, the coordination MMC uses a mechanism that linearly raised the network damping while oscillations proceeded. Dynamical damping (section 3.5.3), which constitutes a significantly more elaborate control mechanisms for the network dynamics, will likely yield better results.

Since coordination is a highly nonlinear problem due to the binary nature of leg mode (swing versus stance), the state space of the coordination problem will likely contain local minima. The momentum mechanisms proposed in section 3.4 might thus help to further improve the performance of the coordination MMC.

Finally, the constraints that are used in the coordination MMC network for introducing some kind of coordination ("safe extreme") are extremely simple. They merely prevent simultaneous swinging of adjacent legs and exceeding of the leg workspace. Using dynamical weighting might open the possibility to implement more elaborate coordination rules, like the ThreshPorta variant (sections 4.1.2.1 and 2.2.1.2).

5.1.4 Complete Integration

All systems discussed above should finally be integrated. Since accurate motion execution is essential after motion planning, the analytical kinematics controllers as proposed in chapter 2 should be used as the reactive base system where posture MMC (chapter

3) and coordination MMC (chapter 4) cannot be used for motion control (i.e. for swing trajectory generation).

The determination of suitable target positions for swinging legs could then be facilitated by evaluating the resulting postures for possible footholds using the posture MMC before executing the motion. When it becomes necessary to shift several legs for assuming another posture, the coordination MMC can be used for evaluating different sequences of the rearrangement in advance.

With such a system, elaborate planning of most aspects of hexapod walking would become possible. Once such a system is implemented, two possible lines of further research are possible.

The first way concerns the generation of swing movements, at this point the only part of hexapod walking not covered by "internal or mental models". However, the serial complexity of stick insect legs is rather low. Cruse argued that serial complexity might be the main factor that has led evolution to evolve mental body models [31]. This thesis demonstrated that it can be sensible to model parallel complexity as well, though. However, swinging legs are not kinematically coupled to the other legs – they constitute open kinematic chains. Thus it might make little sense to model aspects of swing trajectory generation in detail.

The other possible line of research considers higher levels of behaviour. It has already been demonstrated by Cruse that MMC networks can be used for solving navigation problems [32]. Other behavioural problems that could be approached in a similar manner are for example decisions whether an obstacle should be crossed or circumvented.

As mentioned above, all models of stick insect behaviour currently implemented, model aspects of parallel complexity. Serial complexity might however also be modeled in the form of sequential behaviour. Such a sequence might for example regard the path the simulated walker takes from one point to another. Modelled aspects might contain various local landmark situations and obstacles. Kühn and Cruse proposed an MMC network that can model sequential behaviour [81]. The actual behaviour considered in that work is language, though. However, the approach might also be suitable for modeling other kinds of sequential behaviour.

5.2 Cognition

Discussion of problems of cognition on the one hand and control of stick insect walking on the other hand appears to comprise very different domains. This thesis is not actually about cognitive capabilities of stick insects, though. The work on motion planning was inspired by research on human arm motion (see Cruse et al. [22, 26]). The choice to implement Cruses's ideas about motion planning [31] on top of the Walknet rather than on top of an analogous controller for human arm movement is a practical one: While work on modeling human arm control has just begun (see Hartmeier [62]), research on modeling stick insect walking is going on for decades and has already resulted in advanced controllers like the Walknet.

However, if the propositions of the first part of the Discussion were to be realized, what would this imply? Would such a project actually be the implementation of a cognitive entity? Before attempting to answer that question, a couple of theoretical concepts have to be introduced:

5.2.1 Models

Modelling may address different levels and aspects. The Walknet is a computer model of the controller supposed to enable stick insects to walk. A dynamics simulation of a stick insect and its environment is used to test the Walknet. This dynamics simulation is a model of the stick insect's mechanics and its interaction with its environment.

On the one hand, when analysing the stick insects mechanics, physical laws have to be applied. On the lowest level one might have to analyze the forces between atoms, applying laws of relativity and quantum mechanics, presumably though, Newtons laws will suffice for the purpose of this work. When the original stick insect controller was analyzed, one had to deal with chemical and electrochemical phenomena.

On the other hand, when analyzing the Walknet and the dynamics simulation, one has to deal with mathematics and program code. Everything comes down to zillions of transistors having one of two possible states. How are these two phenomena – the real insect and its simulation – related?

If the simulation is indeed a valid simulation of the stick insect, the short answer is: Insect and simulation share a similar causal topology as will be explained below.

5.2.1.1 Causal Topology

Causal relationships have two or more elements: at least one cause, the subject of the relationship, and one effect, the object. A billiard player hits the white ball with a cue. In this case the system billiard player/cue become the subject of a causal relationship with the white ball, the object. The billiard player is, presumably, not utterly inapt and the white ball will consequentially hit another ball, thus becoming a subject in a causal relationship itself ... and so on. Each element of the billiard game is a node in the network of the causal relationships inherent in the game. A formal description of this network is given by the causal topology: analyzing the connections between the nodes.

A good computer simulation of a billiard game will have a similar causal topology as the game itself. Hopefully the Walknet has a similar causal topology as the neural networks in the stick insect that are responsible for generating its gait. As long as the causal topology is similar, it is irrelevant if the actual implementation relies on artificial networks – which in the Walknet are for a good part trained by back propagation to emulate kinematics computations – or directly on the computation of kinematics. The implementation as artificial networks may have the advantage of being more convincing to other researchers, in particular to neurobiologists. But it has numerous disadvantages, too: it is very hard to adapt the artificial networks to new walker geometries; it is not easily testable, whether the implementation is correct, since weight matrices are not analyzable with simple mathematics; the functionality of the networks is very hard to extend.

Since the implementation as artificial neural networks has already been demonstrated to be feasible, there is no reason to choose such an implementation for further research. The development of the Akin controller would have taken much longer, had it been done using artificial neural networks. The same is true for the posture- and the coordination MMC networks.

5.2.1.2 Systems

There is a fundamental difference between a billiard ball, the game of billiard, the Walknet and the posture MMC network. This difference is found in the very nature of the causal topology of each of these phenomena.

The causal topology of a billiard ball is very structured, very constant and very simple. When a billiard ball is hit by another ball, the shock wave will pervade the hit ball in easily predictable ways. The ball reacts similarly to identical inputs and its causal topology is unchanged, if it is not hit too hard. Note that the actual structure of the ball (the exact arrangement of its constituting atoms) will probably change a bit over time. However, the actual structure of its causal topology will be rather constant. The possible range of the ball's reactions to variable inputs is very limited and governed by few simple rules, i.e. its causal topology is rather simple.

Like its balls, the game of billiard has a relatively simple causal topology – it can be completely described by a few basic laws and the coordinates of the constituting balls. But quite opposed to its balls the game's causal topology is volatile, i.e. the causal topology is unstructured (at least after the first shot) and changes from shot to shot, because the coordinates of its balls change. The game will never (or rather extremely rarely) react similarly to the same input (cue motion).

The causal topology of the Walknet, though, is very constant. It will always react similarly to identical inputs in most cases, since previous states of the Walknet play only a very minor role. The reason for this is that information in the Walknet flows mostly into one direction: from sensors to motors. The Walknet's reactions to variable inputs follow relatively complex rules, i.e. it has a complex causal topology. I call phenomena with relatively complex and constant causal topologies "systems".

The posture MMC network also has a rather constant causal topology. However, its actual reactions to inputs are governed by internal states as well as by inputs. The reason for this is that information is flowing into all directions in this system, not just from inputs to outputs. Such loops in the causal topology are a prerequisite for a system to establish a model about itself. Like the Walknet, the posture MMC has a complex causal topology. The posture MMC has another distinguishing property: Its causal topology can change due to varying internal states. But it has attractor topologies to which it will always return to when disturbed.

5.2.2 Perspective

At any position in space there is only a limited set of information available. Examples are: properties of a medium at the position (temperature, sound waves, chemical texture) and electromagnetic waves crossing the according position. The actual set of available information at that position determines the respective perspective. This also means that a perspective must exist for any position in space and thus for any system.

5.2.2.1 Egocentric versus Allocentric models

Any system as defined in section 5.2.1.2 (excluding the universe as a whole which might or might not be a system according to that definition) must have a given perspective. As discussed above, "perspective" is defined

by the set of information that is available to the system. According to the definition of "system" given in section 5.2.1.2, models of systems are themselves systems (or subsystems respectively). When considering the perspective of models, two forms of representation can be distinguished. These forms are called "egocentric" and "allocentric" model throughout this discussion.

Egocentric models have also been called "implicit models", "lookup tables", or "reactive models" (see Cruse [31]). Egocentric models do not abstract the underlying mechanisms of the modeled systems from the sensory information that is used to construct the model. They will usually merely model the stochastics of sensory information. That means that stimulus configurations are mapped to certain responses, without abstracting the causal topology of the phenomenon that causes the stimulus. Examples are

- associative memory modules. In a given stimulus situation such modules will retrieve information that was previously recorded in similar stimuli configurations.
- heteroassociators that will produce a previously learned response to a given stimulus configuration. Like associative memory, heteroassociators can for example be implemented as Perceptrons (see Rosenblatt [107]) or Hopfield networks (see [64, 65]).
- Tani proposed networks that contain specialized sub-networks to predict the output of other sub-networks [123] (Tani uses Jordan networks that might be described as asymmetric Hopfield networks with refined architecture). These predictors then gate the activity of the other sub-networks and can thus be regarded to make behavioural decisions. The predictors can be regarded as egocentric self models.

In contrast allocentric models model the actual rules governing the observed behaviour of the modeled phenomena. They have a similar causal topology as the modeled phenomenon. An example for an allocentric model is the classical MMC network that models a three segment arm (see Steinkühler and Cruse [122]). This network models the kinematics of the manipulator independently from the observer's perspective. The causal topology of the underlying vector system is expressed in the equations that are underlying the network's weight matrix.

One application of this network is the translation of the joint angles of the manipulator into the Cartesian coordinates of its tip. The same transformation can for example be computed by the height-net subsystem of the Walknet. The height-net is a feed forward network that is trained by back propagation to approximate that exact transformation. However, the height-net has just the three angles as inputs and only the z-coordinate as an output. It constitutes a heteroassociator, that was trained to approximate the according mapping function. The training is susceptible to over-training and other adverse effects known to complicate the application of artificial neural networks. Only when the training is very adequate, the mapping will be a good approximation. With its defined inputs the height-net can be said to be adapted to a certain perspective from which it can not generalize. More importantly, the weight matrix of the height net merely reflects the stochastics of the training stimuly rather than the underlying mechanisms.

The MMC network can be used to calculate the inverse transformation as well as any mixed form, and it can even model constraint angle- or Cartesian work-spaces and approximate solutions, when the requested transformation is impossible under the given constraints. The MMC network can thus be said to be a general – allocentric – model of the manipulator, while the height-net is an egocentric model that can only yield the stimulus response for which it was trained.

This example was chosen to make the difference as clear as possible. The dichotomy into egocentric and allocentric models might be artificial, though, or rather the boundary might be blurred. It seems likely that the evolutionary oldest models were egocentric. Evidently though, allocentric models exist, at least in humans – most prominently demonstrated in science. According to evolution theory any development is continuous. Therefore egocentric and allocentric models are probably the extremes of a continuum.

5.2.2.2 Manipulable Allocentric Models and Cognition

Allocentric models have a big advantage over egocentric models: Any of their parameters can be manipulated and the remaining parameters adjust into a configuration that reflects the underlying rules. While egocentric models can only be used to test rather specific stimuli that have to be in accordance with the perspective for which the model was build, allocentric models model the actual causal topology and can therefore be regarded as general problem evaluators for the phenomena they represent.

Thus allocentric models are an important prerequisite for cognition (see Cruse [31]). Yet allocentric models like the three segment manipulator MMC network (see Steinkühler and Cruse [122]) are not sufficient for cognition. Such systems can merely be used to evaluate the effects of given parameter configurations. According to Cruse such evaluation takes place in the course of probehandeln (i.e. imagining an action without actually performing it, see chapter 1).

There have to be additional mechanisms that choose parameters to test. One approach to achieve this has been proposed in this thesis: MMC networks can be designed in such a way that they either fulfil certain constraints (as demonstrated in chapter 4 with the coordination MMC network, also see Cruse et al. [122, 76]) or they can be designed to completely optimize a problem as demonstrated in chapter 3 with the posture MMC network (also see Hopfield and Tank [66] for another approach to decision making using attractor networks). The posture MMC network would not, however, produce novel, "creative" solutions, it would always produce the same result. Therefore sensibly constraining the workspace might be a more promising approach for designing cognitive systems. MMC networks are particularly suitable for allocentric representations with constrained parameter spaces as for example demonstrated by Kindermann and Cruse [76].

Parameters for probehandeln could then be generated by various well established algorithms like random search, evolutionary strategies or pruned search trees as proposed by Aleksander [1]. Note that all of these strategies also require memory to remember tested parameter sets and some algorithm to compare the stored results.

An integrated system with such manipulable models and according parameter generators for probehandeln could indeed be called cognitive according to McFarland and Bösser [84] or Cruse [31].

Bibliography

[1] I Aleksander. Brain inspired computing: The next 50 years of artificial intelligence. *RSA Journal*, 4:74–78, 2000.

[2] J. Angeles. *Fundamentals of robotic mechanical systems: theory, methods and algorithms.* Springer, Berlin, Heidelberg, New York, 1997.

[3] J. W. Arnold. Adaptive features on the tarsi of cockroaches (*Insecta*: *Dictyoptera*). *Int. J. Insect Morphol. and Embryol.*, 3:317–334, 1974.

[4] R. E. Baier, E. G. Shafrin, and W. A. Zisman. Adhesion: Mechanisms that assist or impede it. *Science*, 162:1360–1368, 1968.

[5] U. Bässler and A. Büschges. Pattern generation for stick insect walking movements - multisensory control of a locomotor program. *Brain Res. Rev.*, 27:65–88, 1998.

[6] Bettina Bläsing. Adaptive locomotion in a complex environment: Simulation of stick insect gap crossing behaviour. In A Billard S. Vijayakumar J. Hallam J.-A. Meyer S. Schaal, A. Ijspeert, editor, *From animals to animats 8.*, pages 173–182, Cambridge, MA, 2004. Proceedings of the Eighth International Conference of the Simulation of Adaptive Behaviour, MIT Press.

[7] Bettina Bläsing and Holk Cruse. Stick insect locomotion in a complex environment: climbing over large gaps. *J Exp Biol*, 207(8):1273–1286, 2004.

[8] P. Blazevic, A. Iles, D. E. Okhotsiminsky, A. K. Platonov, V. E. Pavlovsky, and A. V. Lensky. Development of multi-legged walking robot with articulated body. In *CLAWAR 1999*, pages 205–212. Proceedings of the CLAWAR 1999 Conference, 1999.

[9] J. Alphonsius Borellus. *De Motu Animalium. Pars Prima.* 1685.

[10] V. Braitenberg. *Vehikel. Experimente mit kybernetischen Wesen.* Rowohlt Taschenbuch, Deutschland, 1993. ISBN: 3499195313.

[11] Valentino Braitenberg. *Vehicles: Experiments in synthetic psychology.* MIT Press, Cambridge, MA, 1984.

[12] R. A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[13] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[14] Rodney A. Brooks. Integrated systems based on behaviors. *SIGART Bulletin*, 2(4):46–50, 1991.

[15] Rodney A. Brooks. New approaches to robotics. *Science*, 253:1227–1232, 1991.

[16] W. W. Buddenbrock. Der rhythmus der schreitbewegung der stabheuschrecke dixippus. *Biol. Zblatt*, 41:41–48, 1921.

[17] Alan Calvitti and Randall D. Beer. Analysis of a distributed model of leg coordination. I. individual coordination mechanisms. *Biological Cybernetics*, 82:197–206, 2000.

[18] F.-T. Cheng and D. E. Orin. Efficient algorithm for optimal force distribution – the compact-dual lp method. *IEEE Transactions on robotics and automation*, 6(2):178–187, 1990.

[19] H. Cruse. On the function of the legs in the free walking stick insect *Carausius morosus*. *J. Comp. Physiol.*, 112:235–262, 1976.

[20] H. Cruse. The control of the anterior extreme position of the hindleg of a walking insect. *Physiological Entomology*, 4:121–124, 1979.

[21] H. Cruse. A new model describing the coordination pattern of the leg of a walking stick insect. *Biological Cybernetics*, 32:107–113, 1979.

[22] H. Cruse. Constraints for joint angle control of the human arm. *Biological Cybernetics*, 54:125–132, 1986.

[23] H. Cruse. Feeling our body - the basis of cognition? *Evolution and Cognition*, 5(2):162–173, 1999.

[24] H. Cruse, C. Bartling, G. S. Cymbalyuk, J. Dean, and M. Dreifert. A modular artificial neural net for controlling a six-legged walking system. *Biological Cybernetics*, 72:421–430, 1995.

[25] H. Cruse, C. Bartling, and T. Kindermann. High-pass filtered positive feedback for decentralized control of cooperation. In F. Moran, A. Moreno, J. J. Merelo, and P. Chacon, editors, *Advances in Artificial Life*, pages 668–678. Springer, Berlin, Heidelberg, New York, 1995.

[26] H. Cruse and M. Brüwer. The human arm as a redundant manipulator: the control of path and joint angles. *Biological Cybernetics*, 57:137–144, 1987.

[27] H. Cruse, T. Kindermann, M. Schumm, J. Dean, and J. Schmitz. Walknet - a biologically inspired network to control six-legged walking. *Neural Networks*, 11:1435–1447, 1998.

[28] H. Cruse, D. Riemenschneider, and W. Stammer. Control of body position of a stick insect standing on uneven surfaces. *Biological Cybernetics*, 61:71–77, 1989.

[29] H. Cruse and U. Steinkühler. Solution of the direct and inverse kinematic problem by a unique algorithm using the mean of multiple computation method. *Biological Cybernetics*, 69:345–351, 1993.

[30] Holk Cruse. Feeling our body – the basis of cognition? *Evolution and Cognition*, 5(2):162–173, 1999.

[31] Holk Cruse. The evolution of cognition – a hypothesis. *Cognitive Science*, 27:135–155, 2003.

[32] Holk Cruse. A recurrent network for landmark-based navigation. *Biological Cybernetics*, 88:425–437, 2003.

[33] Holk Cruse, Simone Kühn, S. Park, and Josef Schmitz. Adaptive control for insect leg position: controller properties depend on substrate compliance. *Journal of Comparative Physiology A*, 190:983–991, 2004.

[34] Holk Cruse, Ulrich Steinkuehler, and Christian Burkamp. Mmc - a recurrent neural network which can be used as manipulable body model. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S. W. Wilson, editors, *Proc. of the 5th Int. Conf. on Simulation of Adaptive Behavior 1998: From Animals to Animats 5*, pages 381–389, 1998.

[35] Holk Cruse, Ulrich Steinkühler, and Christian Burkamp. Mmc - a recurrent neural network which can be used as manipulable body model. In R. Pfeifer, B. Blumberg, J. A. Meyer, and S. W. Wilson, editors, *Proc. of the 5th Int. Conf. on Simulation of Adaptive Behavior 1998: From Animals to Animats*, volume 5 of *From animals to animats*, pages 381–389. MIT Press, 1998.

[36] A. R. Damasio. *The feeling of what happens. Body and emotion in the making of consciousness.* Harcourt, Brace & Company, New York, San Diego, London, 1999.

[37] J. Dean and H. Cruse. Evidence for the control of velocity as well as position in leg protraction and retraction by the stick insect. In H. Heuer and C. Fromm, editors, *Generation and modulation of action patterns.*, volume 15 of *Exp.BrainRes.Series*, pages 263–274. Springer, Berlin, Heidelberg, New York, 1986.

[38] J. Dean and G. Wendler. Stick insects walking on a wheel: Perturbations induced by obstruction of leg protraction. *J. Comp. Physiol.*, 148:195–207, 1982.

[39] J. Dean and G. Wendler. Stick insect locomotion on a walking wheel: Interleg coordination of leg position. *Journal of Experimental Biology*, 103:75–94, 1983.

[40] J. Decety and J. Grezes. Neural mechanisms subserving the perception of human actions. *Trends in Cognitive Sciences*, 3(5):172–256, 1999.

[41] Fred Delcomyn. Insect walking and robotics. *Annu. Rev. Entomol.*, 49:51–70, 2004.

[42] Rene Descartes. *Meditationes de Prima Philosophia.* Amsterdam, 1642.

[43] M. Desmurget and S. Grafton. Forward modeling allows feedback control for fast reaching movements. *Trends in Cognitive Sciences*, 4(11):423–431, 2000.

[44] V. Dürr. Stereotypic leg searching movements in the stick insect: Kinematic analysis, behavioural context and simulation. *The Journal of Experimental Biology*, 204:1589–1604, 2001.

[45] V. Dürr and C. Brenninkmeyer. The use of the antennae in tactile obstacle localisation in the walking stick insect. page 263, 2001.

[46] Volker Dürr. Biomechanics of an active tactile sensor: the flagellum of carausius morosus. 2005.

[47] Volker Dürr, Andre Krause, Josef Schmitz, and Holk Cruse. Neuroethological concepts and their transfer to walking machines. *Int. J. Robotics Res.*, 22(3):151–167, 2003.

[48] Ö. Ekeberg, M. Blümel, and A. Büschges. Dynamic simulation of insect walking. *Arthropod Structure and Development*, 33:287–300, 2004.

[49] G. W. Ernst and A. Newell. *GPS: a case study in generality and problem solving.* Thesis/dissertation, Carnegie Institute of Technology, New York, 1969.

[50] M. R. Fielding and R. Dunlop. Exponential fields to establish inter-leg influences for omni-directional hexapod gait. In Karsten Berns and Ruediger Dillmann, editors, *Clawar 2001*, pages 587–594, London, 2001. Proceedings of the Fourth International Conference on Climbing and Walking Robots, Professional Engineering Publishing Limited.

[51] Björn Fischer. *Implementierung eines adaptiven Reglers modelliert nach dem Verhalten des Femur-Tibia-Gelenks der Stabheuschrecke.* Diplomarbeit, University of Bielefeld, 2004.

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

[53] J. M. Fuster. *Memory in the cerebral cortex. An Empirical approach to neural networks in the human and nonhuman primate.* MIT Press, 1995.

[54] Vitus Graber. *Über den propulsatorischen Apperat der Insekten.* 1872.

[55] D. Graham. Simulation of a model for the coordination of leg movement in free walking insects. *Biological Cybernetics*, 26:187–198, 1977.

[56] D. Graham. Pattern and control of walking in insects. *Adv. Insect Physiol.*, 18:31–140, 1985.

[57] R. L. Gregory. *The Oxford Companion to Mind.* Oxford University Press, New York, 1987.

[58] R. L. Gregory. *Eye and Brain: The Psychology of Seeing.* Oxford University Press, New York, 1990.

[59] Sten Grillner, Jeanette Hellgren, Ariane Menard, Kazuya Saitoh, and Martin A. Wikström. Mechanisms for selection of basic motor programs - roles for the striatum and pallidum. *Trends in Neurosciences*, 28(7):364–370, July 2005.

[60] Corporate PDP Research Group. *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations.* MIT Press, Cambridge, MA, USA, 1986.

[61] Corporate PDP Research Group. *Parallel distributed processing: explorations in the microstructure of cognition, vol. 2: psychological and biological models.* MIT Press, Cambridge, MA, USA, 1986.

[62] Sven Hartmeier. *Simulation der Pfadplanung von Handbewegungen beim Menschen.* Thesis/dissertation, Diplomarbeit, Bielefeld, 2005.

[63] P. Hoffmann. Beitrag zur Kenntnis der menschlichen Reflexe mit besonderer Berucksichtigung der elektrischen Erscheinungen. *Arch Anat Physiol.*, 1:223–246, 1910.

[64] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci.*, 79:2554–2558, 1982.

[65] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci.*, 81:3088–3092, 1984.

[66] J. J. Hopfield and D. W. Tank. "neural" computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[67] K. H. Hunt. *Kinematic geometry of mechanisms.* Oxford University Press, New York, 1978.

[68] Masao Ito. Movement and thought: identical control mechanisms by the cerebellum. *TINS*, 16(11):448–450, 1993.

[69] M. Jeannerod. The representing brain: Neural correlates of motor intention and imagery. *Behavioural Brain Sciences*, 17:187–245, 1994.

[70] M. Jeannerod. *The cognitive neuroscience of action.* Oxford: Blackwell, 1997.

[71] M. Jeannerod. The 25th bartlett lecture. to act or not to act: Perspectives on the representation of actions. *The Quarterly J. of Experimental Psychology*, 52A(1):1–29, 1999.

[72] Sangbae Kim, Alan T. Asbeck, Jonathan E. Clark, and William R. Provancher. Spinybotii: Climbing hard walls with compliant microspines. In *ICAR 2005 (in press)*. Proceedings of the IEEE ICAR 2005 Conference, 2005.

[73] T. Kindermann. *Positive Rückkopplung zur Kontrolle komplexer Kinematiken am Beispiel des hexapoden Laufens: Experimente und Simulationen.* Thesis/dissertation, Universität Bielefeld, Fakultät f. Biologie, Abt. Kybernetik, 2003.

[74] T. Kindermann, H. Cruse, and J. Dean. A biologically motivated controller for a six-legged walking system. volume 4 of *Proc. of the 24th Annual Conference of the IEEE Industrial Electronics Society (IECON 98)*, pages 2168–2173. 1998.

[75] Thomas Kindermann. Behavior and adaptability of a six-legged walking system with highly distributed control. *Adaptive Behavior*, 9(1):16–41, 2002.

[76] Thomas Kindermann and Holk Cruse. Mmc - a new numerical approach to the kinematics of complex manipulators. *Mechanism and Machine Theory*, 37:375–394, 2002.

[77] C. A. Klein and S. Kittivatcharapong. Optimal force distribution for the legs of a walking machine with friction cone contraints. *IEEE Transactions*

on robotics and automation, 6(1):73–85, 1990.

[78] Andre Krause and Volker Dürr. Tactile efficiency of insect antennae with two hinge joints. *Biological Cybernetics*, 91(3):168–181, 2004.

[79] I. Krück. *Anatomie des Tarsus*. Staatsexamen, Universität Kaiserslautern, 1976.

[80] E. I. Kugushev and V. S. Jaroshefskij. Problems of selecting a gait for an integrated locomotion robot. In *Proc. Fourth Int. Conf. Artificial Intelligence*, pages 789–793, Tiblisi, Georgian SSR, USSR, 1975.

[81] Simone Kühn and Holk Cruse. Static mental representations in recurrent neural networks for the control of dynamic behavioural sequences. *Connection Science*, 17:343–360, 2005.

[82] K. Kutzbach. Mechanische leitungsverzweigung. *Maschienenbau, der Betrieb*, 8:710–716, 1929.

[83] P. Lanz and D. McFarland. Philosophical perspectives on representation, goals, and cognition. In Holk Cruse, Jeffrey Dean, and Helge Ritter, editors, *Prerational Intelligence: Adaptive Behavior and Intelligent Systems Without Symbols and Logic, Volume 1, Volume 2 Prerational Intelligence: Interdisciplinary Perspectives on the Behavior of Natural and Artificial Systems*, volume 3 of *Studies in Cognitive Systems*, pages 247–263. Kluwer Academic Publishers, Dordrecht, 2000.

[84] David McFarland and Thomas Boesser. *Intelligent behavior in animals and robots*. MIT Press, Cambridge, Mass., 1993.

[85] Robert B. McGhee and Geoffrey I. Iswandhi. Adaptive locomotion of a multilegged robot over rough terrain. *IEEE Transactions on Systems, Man and Cybernetics*, 9(4):176–182, 1979.

[86] Robert B. McGhee, V. C. Jaswa, and D. E. Orin. Interactive computer-control of a six-legged vehicle with optimization of stability, terrain adaptability and energy. *Proc. of 1976 IEEE conference on decision and control*, pages 382–391, 1976.

[87] T. Metzinger. *Being No One. The Self-Model Theory of Subjectivity*. MA: MIT Press, Cambridge, 2003.

[88] Thomas Metzinger. The subjectivity of subjective experience: A representationalist analysis of the first-person perspective. In Thomas Metzinger, editor, *Neural Correlates of Consciousness: Empirical and Conceptual Questions*. MA: MIT Press, Cambridge, 2000.

[89] M. L. Minsky and S. A. Papert. *Perceptrons*. MIT Press, Cambridge, MA, 1969.

[90] Marvin Minsky. *A Framework for Representing Knowledge*. Massachusetts Institute of Technology, Cambridge, MA, USA, 1974.

[91] M. A. Nahon and J. Angeles. Real-time force optimization in parallel kinematic chains under inequality constraints. *IEEE Transactions on robotics and automation*, 8(4):439–450, 1992.

[92] G. M. Nelson and R. D. Quinn. Posture control of a cockroach-like hexapod robot. pages 157–162, Leuven, Belgium, May 1998.

[93] A. Newell, J. C. Shaw, and H. A. Simon. Report on a general problem-solving program for a computer. pages 256–264, 1959.

[94] Prabir K. Pal and K. Jayarajan. Generation of free gait – a graph search approach. *IEEE Transactions on Robotic and Automation*, 7(3):299–305, 1991.

[95] K. G. Pearson and R. Franklin. Characersitics of leg movements and patterns of coordination in locusts walking on rough terrain. *Int'l. J. Robotics Res.*, 3:101–112, 1984.

[96] Y. Pelletier and C. D. McLeod. Obstacle perception by insect antennae during terrestrial locomotion. *Physiological Entomology*, 19??:360–362, 1994.

[97] Wilder Penfield and Edwin Boldrey. Somatic motor and sensory representation in the cerebral cortex of man as studied by electrical stimulation, by wilder penfield and edwin boldrey. london, j. bale, sons & curnow, 1937. *Brain*, LX:389–443, 1937.

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

[99] F. Pfeiffer, J. Eltze, and H. J. Weidemann. Six-legged technical walking considering biological principles. *Robotics and Autonomous Systems*, 14:223–232, 1995.

[100] J. M. Porta and E. Celaya. Efficient gait generation using reinforcement learning. In Karsten Berns and Ruediger Dillmann, editors, *Clawar 2001*, pages 411–418, London, 2001. Proceedings of the Fourth International Conference on Climbing and Walking Robots, Professional Engineering Publishing Limited.

[101] J. M. Porta and E. Celaya. Reactive free-gait generation to follow arbitrary trajectories with a hexapod robot. *Robotics and Autonomous Systems*, 47:187–201, 2004.

[102] Josep M. Porta and Enric Celaya. A control structure for the locomotion of a legged robot on difficult terrain. *IEEE Robotics and Automation Magazine, Special Issue on Walking Robots*, 5(2):43–51, 1998.

[103] W. Prinz. Why Donders has led us astray. In B. Hommel and W. Prinz, editors, *Theoretical issues in stimulus-response compatibility.*, Adavances in Psychology, book chapter 118, pages 247–267. Elsevier, max-planck-institut für psychologische forschung, münchen; und ludwig-maximilians-universität, münchen edition, 1997.

[104] William R. Provancher, Jonathan E. Clark, Bill Geisler, and Mark R. Cutkosky. Towards penetration-based clawed climbing. In *CLAWAR 2004 (in press)*. Proceedings of the CLAWAR 2004 Conference, 2004.

[105] V. S. Ramachandran, D. Rogers-Ramachandran, and S. Cobb. Touching the phantom limb. *Nature*, 377:489–490, 1995.

[106] G. Rizzolatti and M. A. Arbib. Language within our grasp. *Trends in Neuroscience*, 21:188–194, 1998.

[107] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Cornell Aeronautical Laboratory, Psychological Review*, 65(6):386–408, 1958.

[108] L. M. Roth and E. R. Willis. Tarsal structure and climbing ability of cockroaches. *J. exp. Zool.*, 119:483–517, 1952.

[109] S. Salmi and A. Halme. Implementing and testing a reasoning-based free gait algorithm in the six-legged walking machine "mecant". *Control Eng. Practice*, 4(4):487–492, 1996.

[110] E. Sauser and A. Billard. Three dimensional frames of reference transformations using gain modulated populations of neurons. In *In Proceedings of 12th European Symposium on Artificial Neural Networks, Bruges (Belgium)*, 2004.

[111] J. Schmitz, J. Dean, T. Kindermann, M. Schumm, and H. Cruse. A biologically inspired controller for hexapod walking: simple solutions by exploiting physical properties. *Biol. Bull.*, 200:195–200, 2001.

[112] Axel Schneider, Holk Cruse, and Josef Schmitz. Decentralized control of elastic limbs in closed kinematic chains. *The International Journal of Robotics Research (submitted)*.

[113] Axel Schneider, Holk Cruse, and Josef Schmitz. A biologically inspired active compliant joint using local positive velocity feedback (lpvf). *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 35(6):1120–1130, 2005.

[114] Axel Schneider, Josef Schmitz, and Holk Cruse. A bio-inspired joint controller for the decentral control of a closed kinematic chain consisting of elastic joints. In *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference (in press)*, Seville, Spain, 2005.

[115] P. Schneider and K. Reitberg. Beinkoordination und Lauf über Hindernisse von *Carabus violaceus* (*Coleoptera*). *Zool. Jb. Physiol.*, 92:351–364, 1988.

[116] N. Sharkey and T. Ziemke. *Life, mind and robots: the ins and outs of embodiment*, pages 313–332. Springer Verlag, Heidelberg, 2000.

[117] M. A. Smith and R. Shadmehr. Error correction and the basal ganglia. *Trends in Cognitive Sciences*, 4:367–369, 2000.

[118] Dirk Spenneberg. A hybrid locomotion control approach. In *CLAWAR 2005 (in press)*. Proceedings of the CLAWAR 2005 Conference, 2005.

[119] Dirk Spenneberg and Frank Kirchner. Scorpion: A biomimetic walking robot. VDI-Bericht 1679, VDI, 2002.

[120] U. Steinkuehler. *MMC-Modelle zur Loesung kinematischer Aufgabenstellungen eines redundanten Manipulators*. Thesis/dissertation, Doctoral

Dissertation,Univ. Bielefeld, Fakultaet fuer Mathematik, 1994.

[121] U. Steinkühler, C. Burkamp, and H. Cruse. Mmc - a holistic system for a nonsymbolic internal body representation. In H. Cruse, J. Dean, and H. Ritter, editors, *Prerational Intelligence*. Kluwer, 1998.

[122] U. Steinkühler and H. Cruse. A holistic model for an internal representation to control the movement of a manipulator with redundant degrees of freedom. *Biological Cybernetics*, 79:457–466, 1998.

[123] Jun Tani and M. Ito. Self-organization of behavioral primitives as multiple attractor dynamics: A robot experiment. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 33(4):481–488, 2003.

[124] David Wettergreen and Chuck Thorpe. Developing planning and reactive control for a hexapod robot. In *Proceedings of ICRA '96*, volume 3, pages 2718–2723, April 1996.

[125] Wigglesworth. How does a fly cling to the undersurface of a glass sheet? *J. exp. Biol.*, 129:373–376, 1987.

[126] D. M. Wilson. Insect walking. *Ann. Rev. Ent.*, 11:103–122, 1966.