

# BIO-INSPIRED EMERGENT CONTROL OF LOCOMOTION SYSTEMS

Mattia Frasca  
Paolo Arena  
Luigi Fortuna



Series Editor: Leon O. Chua

# BIO-INSPIRED EMERGENT CONTROL OF LOCOMOTION SYSTEMS

Mattia Frasca

Paolo Arena

Luigi Fortuna

Università degli Studi di Catania, Italy

# Contents

<i>Preface</i>	vii
1. Introduction	1
1.1 The Central Pattern Generator (CPG)	2
1.2 Locomotion control in hexapods	4
1.3 Main topics of the work	6
2. CNN-based Central Pattern Generators	9
2.1 Introduction	9
2.2 Brief overview on CNN architectures	12
2.3 The CPG neuron	14
2.3.1 The synapse model	18
2.4 The CNN-based CPG	21
2.4.1 RD-CNNs to design artificial locomotion patterns	22
2.4.2 Guidelines for CNN-based CPG design	24
2.5 Example: the caterpillar gait for hexapods	28
2.6 A spatio-temporal algorithm for controlling locomotion of a hexapod robot	32
2.7 Motor-neurons and inter-neurons	37
3. CNN-based CPGs with sensory feedback and VLSI implementation	43
3.1 Direction control	43
3.1.1 The CPG cell	44
3.1.2 CPG with sensory feedback for direction control	47
3.2 Feedback from ground contact sensors	50

3.2.1	Behavior of the CNN neuron driven by a periodic forcing signal . . . . .	51
3.2.2	CPG with ground contact feedback: results . . . . .	54
3.3	Reflex implementation . . . . .	57
3.3.1	The elevator reflex . . . . .	57
3.3.2	The searching reflex . . . . .	58
3.4	Speed control . . . . .	60
3.4.1	Speed of the gait . . . . .	61
3.4.2	Locomotion pattern . . . . .	63
3.5	The CNN-based CPG VLSI chip . . . . .	63
3.5.1	The hybrid approach . . . . .	63
3.5.2	The VLSI Circuit Design . . . . .	64
3.5.3	Experimental results . . . . .	66
4.	Decentralized locomotion control . . . . .	73
4.1	CNN-based decentralized control model . . . . .	73
4.1.1	The decentralized control paradigm . . . . .	75
4.1.2	The CNN leg controller . . . . .	77
4.1.3	The whole control system and results . . . . .	80
4.1.3.1	CNN Decentralized Control . . . . .	80
4.1.3.2	Choice of the parameters of the model . . . . .	83
4.1.3.3	Robustness of the CNN Decentralized Controller . . . . .	86
4.2	Integrate-and-fire neurons and decentralized control . . . . .	88
4.2.1	The leg controller . . . . .	90
4.2.1.1	Biological motivations . . . . .	90
4.2.1.2	The integrate-and-fire neuron . . . . .	90
4.2.1.3	Scheme of the leg controller . . . . .	91
4.2.1.4	The elevator reflex . . . . .	95
4.2.2	The whole control scheme . . . . .	97
4.3	CPG and decentralized control . . . . .	99
5.	A gallery of bio-inspired robots . . . . .	101
5.1	Lampbot: A lamprey robot controlled by RD-CNN . . . . .	101
5.2	MTA hexbot: a hexapod robot controlled by MTA-CNN . . . . .	106
5.3	MTA hexbot II: a remote-controlled hexapod robot . . . . .	110
5.4	MTA hexbot III: a robot driven by the CNN-based CPG VLSI chip . . . . .	112



6. High-level analog control: attitude control and Motor Maps	115
6.1 CNN-based attitude control	115
6.1.1 The CNN for gait control	117
6.1.2 The attitude control CNN	119
6.1.3 Experimental tests	123
6.2 Motor Maps and attitude control	125
6.2.1 Motor Maps	125
6.2.2 Motor Maps for Chaos Control	129
6.2.3 Motor Maps for attitude control	132
6.2.4 Motor Map-based attitude control in a simplified biped model	136
6.3 Learning with Motor Maps	142
7. High-level analog control: Turing patterns and autowaves	145
7.1 Reaction-Diffusion CNN	146
7.2 Navigation control based on Turing patterns	148
7.2.1 Turing patterns and CNNs	149
7.2.2 From CNN patterns to action patterns	151
7.2.3 Experimental Setup	153
7.2.4 To probe further	154
7.3 Navigation control based on autowaves	156
7.3.1 The CNN algorithm	157
7.3.2 Implementation on a roving robot and experimental results	160
8. Conclusions	165
Appendix A HexaDyn and CNNTLab: two tools for bio-inspired locomotion control	171
A.1 HexaDyn, a simulator for the hexapod robot	171
A.2 CNNTLab	173
Appendix B Design of the CNN circuit	177
Appendix C A chaos-based sensor for bio-inspired robots	181
C.1 Continuous CPPM	181
C.2 The CPPM Sonar	183

*References*

189

*Index*

197

# Preface

Living creatures show distinct abilities to interact adaptively with their environment. These characteristics find their roots in the self-organizing dynamics of neural circuits, which in nonlinear science represent the highest example of emergent system behavior. In this work bio-inspired locomotion control is achieved by means of arrays of locally coupled nonlinear systems (Cellular Nonlinear Networks). The main point of the approach is that nonlinear oscillators are an efficient way not only to model the principles underlying neural structures devoted to locomotion control in living creatures, but more importantly to implement them to build autonomous walking robots. In this context the concepts of self-organization and synchronization, universal paradigms of science and real world laws, play a fundamental role in the emergence of the complex behavior of the network of nonlinear circuits. The proposed method is efficient at all levels of the hierarchical control of locomotion: the low level of gait generation, reflex and sensory feedback implementation, and the high level of posture stabilization and behavioral modules. The strategy introduced is hardware-oriented, so the concepts of emergent behavior and self-organization can be appreciated in chip devices applied to walking machines. Experimental results obtained with a prototype of the VLSI chip implementing the Cellular Nonlinear Network generating the locomotion pattern and tests on a bio-inspired insect-like robot are presented.

This book constitutes a complete monograph on new locomotion control strategies of bio-inspired robots implemented through Cellular Nonlinear Networks (CNNs). The new approach for locomotion control, based on emergent properties of nonlinear dynamical systems, is exhaustively described and several case studies are presented. Topics related to the problem of locomotion control from low level to high level control are addressed.

The book contains both a detailed introduction on the fundamentals of Cellular Nonlinear Networks and bio-inspired control and research results with particular emphasis on applications. The proposed methodology is illustrated with the help of several examples dealing with bio-inspired hexapod robots. The present book is thus aimed both at newcomers to the field of bio-inspired locomotion control and graduate students in advanced courses on bio-inspired robotics as well as PhD students and researchers both from academic institutions and industrial companies. In particular, the main features of the approach beneficial to the reader are the following: analog computation is used to control locomotion systems; realization of well-tested circuit for control is discussed; each reader could implement its own circuits at low cost (few dollars for each analog unit).

We wish to express our gratitude to Prof. L. O. Chua who has been a constant source of inspiration and encouragement for this work. Part of this work has been partially supported by Office of Naval Research, project number N00014-03-1-0392, and by MIUR (“Ministero dell’Istruzione, dell’Università e della Ricerca”) under project PRIN “Innovative Bio-Inspired Strategies for the Control of Motion Systems”. Moreover, we thank STMicroelectronics (Catania site) for its fruitful support.

## Chapter 1

# Introduction

The problem of locomotion control in legged robots can be faced by taking inspiration from biology and applying to walking robots the principles underlying the neural control of locomotion in animals. This is the viewpoint of the so-called *biologically inspired robotics* (whose interests, of course, are not restricted to locomotion control) and is a major theme of this book.

Bio-inspired robotics [Webb and Consi (2001); Arkin (1998)] is, by definition, a broad field, including synergies from various disciplines: neuroscience, biology, ethology and robotics. Robotics itself is multi-disciplinary involving at least mechanics, electronics and computer science. Hence a large variety of bio-inspired approaches have been developed, whose degree of biological inspiration ranges from very accurate mechanical designs (the most representative example of which is the cockroach-like robot illustrated in [Quinn *et. al.* (1998)]) to neuromorphic systems [Horiuchi (1992)].

From the engineering viewpoint the main reason for the great interest in bio-inspired approaches is the fact that bio-robotics provides suitable solutions for the design of efficient walking robots since the nature of the problems an animal and a legged robot deal with is the same. These solutions are very often common principles shared by a large variety of animals and are even present in low phyla, so they appear as *simple* solutions to hard problems. Applications of these principles are possible since great advances have been made by biologists in understanding animal locomotion, and at the same time they are interesting topics of study for biologists since bio-robots are a good realistic way to verify a hypothesis regarding the biological model and a good source for new ideas.

For this reason bio-inspired robotics includes both robotic animal models (that can be useful for a better understanding of biological behavior), and more abstract robotic models, and thus benefits both robotics and

biology [Marder (2001); Webb (2001)].

This work focuses on bio-inspired locomotion control in a hexapod robot. This includes both a low-level control and a brief look at the possibilities of high-level control, for example attitude control and trajectory planning. Two different biological hypotheses are considered for the low-level locomotion control: the *Central Pattern Generator* and the *decentralized locomotion control*. Both of them lead to suitable control schemes with different advantages and drawbacks.

The control of legged locomotion is by no means a trivial problem and large computational resources are often required to solve this problem in real time. On the other hand the high degree of adaptability even *simple* animals possess when moving in complex, unstructured environments is surprising. It is more exciting to observe how a universal paradigm emerges from studies performed on very different animals: the slow movement of the two wings of the sea angel *Clione limacina*, a marine mollusc, and the perfect running of the cat; the undulatory swimming movements of one of the oldest vertebrates, the lamprey; the flight of the locust and the scratching of domestic mice are regulated by motor systems whose structures share surprising similarities [Orlovsky *et. al.* (1999)]. All these examples exhibit a hierarchical organization of the motor system; this paradigm, called the *Central Pattern Generator* (CPG), is a central topic of this work and therefore needs to be dealt with in greater detail.

## 1.1 The Central Pattern Generator (CPG)

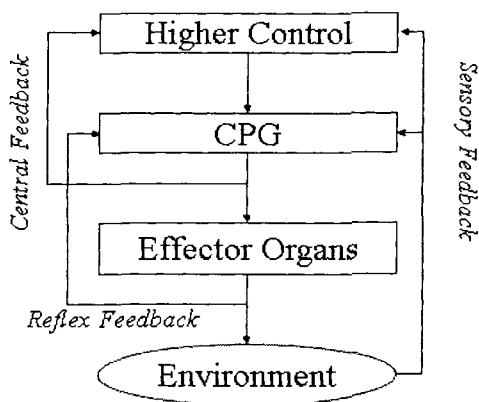
The history of progress in the knowledge and understanding of neural control of locomotion demonstrates the usefulness of the synergy between engineering and biology. The first pioneering studies, based on photographic techniques to capture the essential features of animal locomotion, have evolved to include high resolution frequency video-recordings of animal locomotion, recordings of the activity of individual muscles using an electromyograph (EMG) and even computer simulations of possible models [Shepherd (1997)]. These techniques provide quite an accurate sequence of animal movements and monitoring of the neurons involved in locomotion control. Although they do not provide a complete understanding of the whole system (often made up of a huge number of neurons), these studies reveal a common paradigm in many animal species, from invertebrates to vertebrates.

First of all, they reveal that the pattern of locomotor activity is due to a pattern of neural activity. Then different elements, hierarchically organized and common to the motor system of various animals, can be identified despite the differences in the kind of locomotion and body forms.

The key component of the motor system is the Central Pattern Generator (CPG), a neural circuit that can produce a rhythmic motor pattern with no need for sensory feedback or descending control. It is mainly located in the spinal cord of vertebrates or in relevant ganglia in invertebrates. The CPG includes all of the essential neural mechanisms for generating coordinated rhythmic motor-neuron and therefore muscle outputs. Signals from the CPG directly control the effector organs (cilia, legs), while the signals it receives from higher control neurons are necessary only to initiate the locomotion, but not to generate the correct pattern of movements: indeed the generation of rhythmic movements takes place at the neural level of the CPG. This has been demonstrated in a number of experiments, animals deprived of commands from high-level neural centres are still able to locomote provided there is external stimulation to the CPG. Likewise, the presence of feedback from environment is not strictly necessary to generate a rhythmic pattern: this has been observed even when the feedback is inhibited in what is called *fictive locomotion* [Cohen and Wallen (1980)], consisting of removing sensory feedback and descending control and eliciting the motor pattern. Obviously, the feedback signals modify the locomotion pattern and are fundamental to achieve good performance by the whole control system in real environments. Likewise, removing higher control signals often results in a loss of the stability for the animal. In other words, it appears that the posture control involves higher neuron centers. We will come back to this important issue when dealing with the problem of attitude control in the hexapod robot.

Figure 1.1 shows the key features of the motor system of many animals, emphasizing the presence of different types of feedback: central feedback directed to the higher control neurons, reflex feedback from the motor output to the CPG, and sensory feedback from the environment to either the CPG or the higher level control. In Fig. 1.1 the feedback paths on the left represent the flow of proprioceptive information, while those on the right represent the flow of exteroceptive information.

It should be pointed out that in many biological cases the functional subdivision shown in Fig. 1.1 does not correspond to different neural networks: the same networks may act at different levels of the structure (for example in the *Tritonia* [Shepherd (1997)]).



## The motor system

Fig. 1.1 The main functional features of the motor system are common to many animals, from the *Clyone* to the cat.

Besides eliciting the motor pattern, the higher control neurons are responsible for important decisions such as the choice of the right locomotion pattern (i.e. swimming if the animal is in water or walking if it is on the ground) or escape reactions.

The great importance of the CPG lies in the fact that central generators of rhythms have been demonstrated in all animals to account for rhythmic movements that are essential for survival: the ability to move (defined by Shepherd [Shepherd (1997)] as “the most important characteristic of animal life”) and, an even more significant example, breathing in vertebrates [Randall *et. al.* (2001)].

### 1.2 Locomotion control in hexapods

The walking of hexapod insects is probably the most controversial example of the role of the CPG in locomotion control. Stick insects, cockroaches and locusts have been widely studied by neurophysiologists. Recordings of walking by video cameras and other experiments reveal various types of behavior: reflexes and typical locomotion gaits can be reconstructed from experimental data and thoroughly investigated. These studies reveal



a fundamental role played by sensors in determining locomotion gait.

In absence of sensory feedback the behavior of these insects is different: while rhythmic movements of the single legs are still evident, but the coordination among legs is lost in the cockroach, in the stick insect even coordination among the joints of each single leg disappears [Orlovsky *et. al.* (1999)]. This behavior is explained by different models: CPG supporters hypothesize that there exist two different kinds of CPG: the first, accounting for the findings on the cockroach, consists of a common oscillator for different leg joints; the second, accounting for the findings on the stick insect, consists of an individual oscillator for each joint. Since the interactions between the CPGs are not sufficient for the normal coordination of legs in absence of feedback (the phase shifts between the legs are abnormal in such cases), it seems clear that sensory feedback plays an important role in these cases. In other hexapod walkers, the crustaceans, the situation seems less critical; the rhythm, artificially promoted, resembles the forward or backward walking of intact animals, but is usually slower [Chrachri and Clarac (1987)].

The neural network devoted to locomotion control in hexapod walkers remains unclear, since there are at least two other factors, not yet discussed, that play an important role in the locomotion control: posture control which involves high-level controls, and the high number of neurons (indeed it is more correct to talk about motor-neuron pools than motor-neurons).

Although the experiments on the stick insect can be explained by hypothesizing CPGs at the level of individual leg joints, the formulation of a CPG sounded quite artificial to some biologists and other theories have been developed to take into account the complex features of locomotion control in the stick insect. Cruse formulated a decentralized locomotion control model in which there is no endogenous oscillator: the stepping rhythm is induced by sensory feedback [Cruse (1990)].

The decentralized control paradigm is based on a distributed network of neural controllers, with no hierarchy, that are able to generate proper motions based on reflexes and local influences [Cruse *et. al.* (1998a)] (unlike the CPG it strictly needs sensory feedback for pattern generation, while the generation of the locomotion pattern in the CPG takes place even in absence of sensory feedback). The decentralized model is based on a controller network for each leg and on a set of local influences acting on two kinematic parameters: the posterior extreme position (PEP) and the anterior extreme position (AEP) of the leg contact point. The kinematics of a leg is divided into two phases: the *stance* phase (or *power stroke*) in which the leg is on

the ground, supports the weight of the body and moves in the opposite direction to the body motion; and the *swing* phase (or *return stroke*) in which the leg is lifted off the ground and moves in the direction of the insect motion. The AEP represents the point when the leg touches the ground, i.e. the transition from the return stroke to the power stroke, while the PEP represents the transition from the power stroke to the return stroke, i.e. the point when the leg is lifted in the air.

The leg controller alternately switches between the stance and swing phases on the basis of sensorial inputs (the ground contact signal, indicating when a leg is on the ground, and the PEP signal, indicating whether the PEP has been reached or not) and proprioceptive information (the joint angles). A set of local influences acting on the kinematic parameters (AEP, PEP) of each leg controller has been shown to be suitable to explain the emergence of a well-defined gait in stick insect locomotion [Cruse *et. al.* (1998a)].

### 1.3 Main topics of the work

This work presents a new strategy for bio-inspired locomotion control based on locally coupled nonlinear circuits, like the motor-neurons of the neural systems in animals. The aim is to discuss the control strategy focusing on both theory and implementation issues, starting from low-level control of locomotion (represented by the CPG or decentralized locomotion control) and extending to a brief discussion of possible bio-inspired strategies based on nonlinear circuits, which are suitable for the implementation of high-level behavior.

Both control schemes, the CPG and decentralized locomotion control, are investigated. From an engineering viewpoint the scheme based on the CPG paradigm is mainly feedforward control (it is able to generate the right signals in absence of feedback), while the decentralized control scheme totally relies on feedback. From the viewpoint of robotics implementing the two approaches is not a mere exercise. They rely on very different hypotheses and therefore offer very different advantages. In principle, it seems more robust to rely on the status of the sensor signals. However, in the case of hexapod walking for example it can be very hard to deal with a distributed networks of sensors: ground contact signals for example are strongly influenced by the force distribution of the hexapod on the legs in the stance phase, hence the ground sensor of a stance leg can give

erroneous information about the leg status. Moreover, biological findings strongly support the CPG scheme. Though the two approaches are by themselves self-contained and exhaustive, it may be worthwhile envisaging a framework that exploits the advantages of the two schemes: the CPG may be more suitable for high speeds, escaping behaviors and so on, while the decentralized scheme can be more efficient in slow walking, exploration and so on.

Features common to the two biological schemes are the concepts of motor-neurons, local influences and signals from high-level control centers. The main difference is the role played by sensor feedback: in decentralized locomotion control it is essential and without sensors no locomotion can be obtained. In the CPG feedback modulates the locomotion pattern, but it is not needed to generate it. These common features allow a unified approach to the implementation of these schemes. The concepts of local interaction, self-organization and nonlinear circuits are the main ingredients of the approach introduced in this book.

Cellular Neural/Nonlinear Networks (CNNs) play an important role in this research field since they offer the possibility of implementing nonlinear dynamics like those arising in natural phenomena, through parallel, fully analog but digitally programmable circuitry. They provide a framework for the implementation of the CPG for locomotion control. Even if the approach is general and can be applied to any locomotion system with particular benefits when the number of actuators grows, the focus of this work is mostly on a hexapod robot, since the need to have an experimental framework, in which to perform real tests of the methodology, forces us to choose a specific example.

The guidelines for the design of the CNN-based CPG are illustrated in Chap. 2. The role of feedback is discussed in Chap. 3. Since the work aims to give particular emphasis to the implementation issue, the design of a VLSI CNN-based CPG chip has been considered. Its features and the experimental results obtained with the first prototype of a CNN chip for locomotion control are also discussed in Chap. 3. The VLSI chip is implemented with a switched-capacitor technique: the choice of the implementation strategy adopted is fundamental to achieve a simple and efficient control strategy.

Chapter 4 introduces decentralized locomotion control: two different implementations, through CNNs and integrate-and-fire neurons, are discussed.

Chapter 5 shows several examples of bio-inspired robots. In particular,

a lamprey robot and several prototypes of hexapod robots are taken into account as applications of the analog approach to locomotion control.

The next two Chapters deal with high-level analog control. In particular, in Chap. 6 the important issue of attitude control for the hexapod robot is approached with two different strategies. The first one is based on distributed control through CNNs, while the second, based on Motor Maps, introduces a more general framework for the control of nonlinear systems in which self-organization and unsupervised learning play a fundamental role in realizing an adaptive controller inspired by the motor cortex of the human brain. Chapter 7 introduces a strategy for obstacle avoidance and navigation control based on autowaves and Turing patterns. A very important point of this strategy is that both the two approaches can be implemented in a CNN. The examples of robots implementing these control strategies are given in these two Chapters.

Chapter 8 compares our work with other bio-inspired approaches and draws the conclusions of the book.

One final remark concerns the tools exploited to investigate the topics dealt with. The implementation of real circuits on either discrete component electronic boards or dedicated chips is often a time-consuming process. On the other hand one of the key points of bio-inspired robotics is to perform experiments on real hardware. For these reasons two tools have been developed. Aiming to be a first step in the design of a legged robot and at the same time a tool for testing bio-inspired control, a software simulator of a hexapod robot with a graphic interface has been considered. Moreover, a framework in which the bio-inspired control system, implemented via software, is running on a PC driving the real robot, through the parallel port or a communication module, has also been implemented. These tools are presented in Appendix A.

Appendix B illustrates the design of the CNN used to implement the CPG. Appendix C illustrates a bio-inspired sensor for robot navigation in complex environments.

Movies and experiments of the bio-inspired robots presented in this book are available on-line [DIEES webpage].

## Chapter 2

# CNN-based Central Pattern Generators

The concept of a Central Pattern Generator (CPG), a network of neurons producing the locomotion pattern within a lattice of neural activity, is a suitable paradigm for biologically inspired control of artificial locomotion. In this Chapter a new approach to the design of CPGs, based on Cellular Neural Networks (CNNs), is presented. From a biological point of view this new approach includes an approximated chemical synapse implemented in a CNN structure. This allows us to obtain a general class of artificial CPGs in which the desired locomotion pattern and switching among patterns are achieved by means of a spatio-temporal algorithm implemented in the CNN structure.

### 2.1 Introduction

Walking animals employ several distinct periodic patterns of leg movements, called gaits. Stereotyped movements (locomotion patterns) are also observed in swimming, flying, crawling animals. The hypothesis underlying the CPG is that these patterns are due to a pattern of neural activities within the CPG network (the key feature of these networks is that the pattern of neural activity is mapped onto the pattern of locomotor activity). This concept is useful in robotics, since the control of a legged robot is a hard task, involving a high number of interconnected degrees of freedom (DOF). Its application to robotics gives rise to the following problem: given a desired behavior (i.e. a locomotion pattern), generating the feedforward signals driving the legs of the robot.

A widespread approach to modelling CPGs is to use dynamical systems [Cohen *et. al.* (1982); Collins and Stewart (1993a); Collins and Stewart (1993b); Golubtisky *et. al.* (1998)]: CPGs are viewed as networks of coupled

nonlinear systems. Some of these works exploit the symmetry of animal gait to build up CPGs and analyze the possible patterns of oscillation by means of considerations on the network symmetry and Hopf bifurcations in symmetrical systems [Golubtisky and Stewart (1985)].

This strategy considers a network of coupled nonlinear systems with given connections and allows the system parameters to be variable in order to account for distinct gaits. The so-called “group-theoretic approach” [Collins and Stewart (1994)] predicts all the possible solutions respecting the symmetry of the network, based on general considerations that are independent on the specific models, but it does not discuss the stability of the solution, which is strictly related to the specific model used. From an engineering viewpoint this is a fundamental feature that needs to be investigated, at least numerically, to guarantee the feasibility of the implementation.

There are common assumptions in the dynamical approach to CPGs: the nonlinear oscillators are often assumed to be identical, the stepping movements of each limb are controlled by a single oscillator, while inter-limb coordination is provided by the connections between the oscillators. Moreover, the results are often general and independent of the physiological interpretation of the oscillators: i.e. the oscillator may represent either a single motor-neuron or a population of neurons controlling the movement of a leg or an inter-neuron. Moreover, as discussed in Chap. 1, biologists and neurophysiologists often refer to CPGs for single leg control. Very often in the dynamical approach to CPGs the term CPG refers to the network controlling the whole motor system, i.e. it comprises both single leg controllers and connections between them.

In the present work the problem of locomotion control is also approached from the perspective of coupled nonlinear oscillators. However, our interest focuses primarily on the design of a network of coupled oscillators where distinct locomotion patterns emerge to provide a suitable control for bio-inspired robots, and in particular for six-legged robots.

Moreover, particular attention is devoted to implementation issues. Therefore, CNNs, known as universal tools for the study and implementation of nonlinear complex dynamics, have been chosen to model both the nonlinear oscillator and the whole population dynamics. However, in order to realize a robotic CPG we are forced to fix the interpretation level: the nonlinear oscillator consists of a second-order nonlinear circuit, made up of two elementary first-order CNN cells, and in the examples given in Chap. 5 it is used to directly control the stepping movement of a leg (in the case of

the hexapod robot) and the movement of a body segment (in the case of the lamprey robot). This oscillator is referred to as the CNN motor-neuron or CPG cell (since it is the basic cell to build up the CPG).

As the approach aims to be general, in the first part of the Chapter we will refer to a generic *locomotion structure* and to a generic *locomotion pattern*. Then we will focus on an example of legged robot.

In particular, the following definition of locomotion pattern is assumed:

**Definition 2.1** (Locomotion pattern.) A locomotion pattern is a pattern of periodic movements for the actuators of a given locomotion structure.

When the attention will focus on the example of hexapod robots, we will refer to gaits instead of locomotion patterns as the term is more appropriate for legged robots.

The hierarchical organization of the biological motor system inspired the design of the analog distributed control of several motion systems as illustrated in [Arena *et. al.* (1999); Arena *et. al.* (2000)]. The control of these robots is based on Reaction-Diffusion equations implemented by means of CNNs and is the starting point for the more general approach discussed in this Chapter. The approach to the design of CPGs for artificial locomotion control depicted in this Chapter is the result of a generalization of the synaptic connections between the neurons constituting the CPG.

It is not quite clear which physiological mechanisms are used to reorganize the neuron configuration in order to switch between locomotion patterns. Two mechanisms involved in this operation are focused on by neurobiologists. In some cases it is only necessary to change the parameters of the synapses, while in other cases a complex reorganization of the network is required. In this latter case some synaptic connections may well be inhibited and others excited towards neurons not yet involved in the former pattern generation. This latter case is implemented by using Reaction-Diffusion (RD) CNNs and is dealt with in Sec. 2.4.1. The more general approach to CPG implementation, based on the principle that the same group of neural cells can be reorganized by changing the synaptic connections, corresponds to a CNN implementation with space variant templates and is dealt with in Sec. 2.4.2.

## 2.2 Brief overview on CNN architectures

CNNs were introduced by L. O. Chua [Chua and Yang (1988a); Chua and Yang (1988b)] in 1988. His idea was to use an array of simple, identical, locally interconnected nonlinear dynamic circuits, called cells, to build large scale analog signal processing systems. The cell was defined as the nonlinear first-order circuit shown in Fig. 2.1(a),  $u_{ij}$ ,  $y_{ij}$  and  $x_{ij}$  being the input, the output and the state variable of the cell respectively. The output is related to the state by the nonlinear equation:

$$y_{ij} = 0.5(|x_{ij} + 1| - |x_{ij} - 1|) \quad (2.1)$$

A CNN is defined as a two-dimensional array of  $M \times N$  identical cells arranged in a rectangular grid, as depicted in Fig. 2.1(b). Each cell mutually interacts with its nearest neighbors by means of the voltage controlled current sources  $I_{xy}(i,j;k,l) = A(i,j;k,l)y_{kl}$  and  $I_{xu}(i,j;k,l) = B(i,j;k,l)u_{kl}$ . The constant coefficients  $A(i,j;k,l)$  and  $B(i,j;k,l)$  are known as the *feedback and input cloning templates*, respectively. If they are equal for each cell, they are called *space-invariant*. If  $B(i,j;k,l) = 0$  the CNN is said autonomous.

A CNN is described by the state equations of all cells:

$$C \cdot \frac{dx_{ij}}{dt} = -\frac{x_{ij}}{R_x} + \sum_{C(k,l) \in N_r(i,j)} A(i,j;k,l)y_{kl} + B(i,j;k,l)u_{kl} + I \quad (2.2)$$

with  $i = 1, 2, \dots, M$  and  $j = 1, 2, \dots, N$  where

$$N_r(i,j) = \{C(k,l) \mid \max(|k-i|, |l-j|) \leq r\}$$

with  $k = 1, 2, \dots, M$  and  $l = 1, 2, \dots, N$  is the *r-neighborhood*.

This model is known as the Chua-Yang model or linear CNN and refers to a single-layer CNN. This model can be extended to a multilayer CNN if instead of only one state variable, there may be several state variables in each cell of the multilayer CNN.

The Chua-Yang model has been generalized in many different ways. These generalizations allow the inclusion in the model (2.2) of nonlinear interactions, direct dependence on the state variables of the neighborhood cells, different grids, and lead to a more general definition for CNNs [Chua and Roska (1993)]:



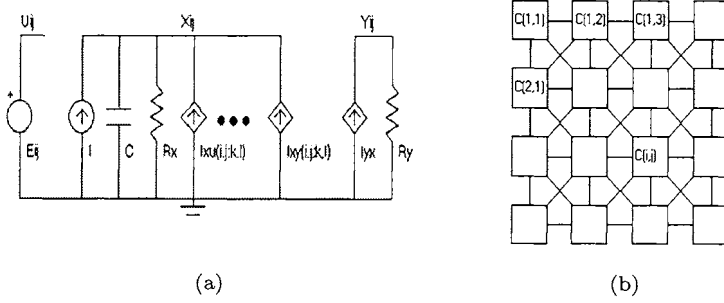


Fig. 2.1 (a) The CNN cell scheme. (b) A CNN bi-dimensional array.

**Definition 2.2** A CNN is an  $n$ -dimensional array of mainly identical dynamic systems, called cells, which satisfies two properties: (a) most interactions are local within a finite radius  $r$ , and (b) all state variables are continuous valued signals.

It follows that a more complete single-layer CNN model including some of the above mentioned generalizations is described by the following normalized state equations (setting  $C = 1$ ,  $R_x = 1$ ):

$$\begin{aligned} \frac{dx_{ij}}{dt} = & -x_{ij}(t) + \sum_{C(k,l) \in N_r(i,j)} \left\{ \hat{A}_{ij;kl}(y_{kl}(t), y_{ij}(t)) + \right. \\ & \left. + \hat{B}_{ij;kl}(u_{kl}(t), u_{ij}(t)) + \hat{C}_{ij;kl}(x_{kl}(t), x_{ij}(t)) \right\} + I_{ij} \end{aligned} \quad (2.3)$$

with

$$y_{ij} = f(x_{ij})$$

where  $\hat{A}_{ij;kl}(\cdot, \cdot)$ ,  $\hat{B}_{ij;kl}(\cdot, \cdot)$  and  $\hat{C}_{ij;kl}(\cdot, \cdot)$  are two-variable nonlinear functions (the nonlinear templates) and  $f(\cdot)$  is the output nonlinearity.

As the CNN for modelling and implementing artificial CPG is a two-layer CNN, we explicit the equations for an autonomous two-layer CNN:

$$\begin{aligned} \frac{dx_{1,ij}}{dt} = & -x_{1,ij} + \sum_{C(k,l) \in N_r(i,j)} \left\{ A_{11}^{ij}(kl)y_{1,kl} + A_{12}^{ij}(kl)y_{2,kl} \right\} + i_1 \\ \frac{dx_{2,ij}}{dt} = & -x_{2,ij} + \sum_{C(k,l) \in N_r(i,j)} \left\{ A_{21}^{ij}(kl)y_{1,kl} + A_{22}^{ij}(kl)y_{2,kl} \right\} + i_2 \end{aligned} \quad (2.4)$$

where  $y_{h,ij}(x_{h,ij})$  (with  $h = 1, 2$ ) is the saturation nonlinearity as in Eq. (2.1) and the feedback template is:

$$A^{ij} = \begin{pmatrix} A_{11}^{ij} & A_{12}^{ij} \\ A_{21}^{ij} & A_{22}^{ij} \end{pmatrix}.$$

Significant attention has been directed to studying the dynamic properties of the various CNN models. One of the most challenging issues is surely stability [Chua and Yang (1988a)]. In fact, the particular structure, high order and nonlinearity of these systems create serious problems. Almost all kinds of dynamic behavior, ranging from simple equilibria to chaos, have been reported among the different kinds of networks.

The CNN behavior is basically dictated by the templates. However, the choice of templates that are suitable to achieve a desired processing task is hard to accomplish in a direct way.

This leads to the so-called *learning and design problem* [Nossek (1994); Nossek *et. al.* (1993)]. The term *design* is used when the desired task can be translated into a set of local dynamic rules, while the term *learning* is used when the templates need to be obtained by learning techniques, so pairs of inputs and outputs must correspond. Some good results have been obtained with discrete-time CNNs in simple cases, but this is a really difficult problem for continuous-time models. Most of the templates currently available have been obtained by intuitive principles and refined by trial and error with the aid of simulators.

### 2.3 The CPG neuron

As basic unit to build up the CNN-based CPG a second-order CNN cell will be taken into consideration. This will be referred in the following as CNN neuron or CPG cell. More precisely, the following definition can be stated:

**Definition 2.3** (The CNN cell-neuron.) The second-order CNN cell represented by the following equations:

$$\begin{cases} \dot{x}_{1,ij} = -x_{1,ij} + (1 + \mu)y_{1,ij} - sy_{2,ij} + i_1 \\ \dot{x}_{2,ij} = -x_{2,ij} + (1 + \mu)y_{2,ij} + sy_{1,ij} + i_2 \end{cases} \quad (2.5)$$

where

$$y_{k,ij} = 0.5(|x_{k,ij} + 1| - |x_{k,ij} - 1|) \quad (2.6)$$

with  $i = 1, 2, \dots, M$ ,  $j = 1, \dots, N$  and  $k = 1, 2$  is the basic unit to build up the CNN-based CPG and is referred as the *CNN neuron* or the *CPG cell*.

The most important characteristics of Eq. (2.5) is that for a suitable choice of parameters it behaves as a nonlinear oscillator with slow-fast dynamics. In other words, Eq. (2.5) admits a stable limit cycle.

**Proposition 2.1** *The CNN neuron of Eq. (2.5) with parameters as in Table 2.1 has a stable limit cycle with slow-fast dynamics.*

The behavior of the CNN neuron (2.5) can be accurately analyzed by considering the phase plane divided into nine different affine subspaces in which either all or some of the cell states are saturated or not, and examining the equilibria of these regions. The corresponding equilibria are indicated with subscripts relating to the partial saturated region ( $p$ ) or saturated region ( $s$ ) and superscripts relating to the value of the saturation (+1, -1) or linear ( $L$ ) value of both variables. They are:

$$\begin{aligned} p_s^{++} &= (1 + \mu - s + i_1, 1 + \mu + s + i_2); \\ p_s^{+-} &= (1 + \mu + s + i_1, -1 - \mu + s + i_2); \\ p_s^{-+} &= (-1 - \mu + s + i_1, -1 - \mu - s + i_2); \\ p_s^{--} &= (-1 - \mu - s + i_1, 1 + \mu - s + i_2); \\ p_p^{L,+1} &= \left(\frac{s-i_1}{\mu}, 1 + \mu + \frac{s}{\mu} \cdot (s - i_1) + i_2\right); \\ p_p^{+1,L} &= \left(1 + \mu + \frac{s}{\mu} \cdot (s + i_2) + i_1, -\frac{s+i_2}{\mu}\right); \\ p_p^{L,-1} &= \left(-\frac{s+i_1}{\mu}, -1 - \mu - \frac{s}{\mu} \cdot (s + i_1) + i_2\right); \\ p_p^{-1,L} &= \left(-1 - \mu - \frac{s}{\mu} \cdot (s - i_2) + i_1, \frac{s-i_2}{\mu}\right); \\ p_p^{L,L} &= \left(-\frac{i_1}{\mu} + \frac{s}{\mu} \cdot \frac{s \cdot i_1 + \mu \cdot i_2}{\mu^2 + s^2}, \frac{s \cdot i_1 + \mu \cdot i_2}{\mu^2 + s^2}\right). \end{aligned} \quad (2.7)$$

The oscillating behavior of the CNN neuron may be proved by verifying the conditions for which the Poincaré-Bendixson Theorem holds [Strogatz (1994)].

This theorem is a powerful tool to establish the existence of periodic orbits in 2D flows. It states that if  $R$  is a closed region that does not contain fixed points for the vector field  $\dot{x} = f(x)$  and a trajectory  $C$  confined in  $R$  does exist, then  $R$  contains a closed orbit (and either  $C$  is itself

the closed orbit or spirals towards to it). The behavior of cell (2.5) is analyzed by dividing the plane into nine regions ( $D_0, D_1, \dots, D_8$ ) delimited by the saturation points. In each of these regions system (2.5) is linear. The region  $R$  can easily be identified by constructing a trapping region, as schematically illustrated in Fig. 2.2.  $H$  represents a closed disk in  $D_0$ , surrounding the equilibrium point of  $D_0$  and whose boundary constitutes the inner boundary of the region  $R$  (see Fig. 2.2). The equilibrium point of  $D_0$  is an unstable equilibrium point, thus the flux is towards  $R$  and outside  $H$ .

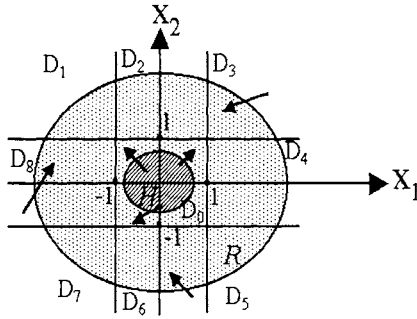


Fig. 2.2 The trapping region  $R$ .

Table 2.1 Parameter values of the CNN neuron (2.5).

$\mu$	$s$	$i_1$	$i_2$
0.5	1	-0.3	0.3

Therefore, to prove the existence of the limit cycle it is necessary to verify that all the stable points among the equilibria (2.7) are virtual, that is, they do not lie in the region to which they belong. In fact, if at least one of these equilibria is not virtual, the trajectory asymptotically converges to this point and so no oscillation may exist.

The importance of this consideration is related to the fact that the values of the bias can influence the *virtuality* of an equilibrium point. Moreover, the role of the bias is fundamental in order to achieve slow-fast dynamics. This is obtained by the interaction of the trajectory with a virtual equilibrium point. The closer a virtual point is to the boundary (between the

region to which it belongs and the region in which it lies) the stronger its slowing action on the dynamics of the trajectory near the virtual point will be.

With the parameters shown in Table 2.1, two virtual points influence the trajectory. Choosing  $i_1 \neq 0$  and  $i_2 \neq 0$  breaks the symmetry of the system, thus letting the trajectory close to two virtual points,  $p_s^{-+}$  and  $p_s^{-}$ . These equilibria are virtual if the following conditions hold:

$$i_1 < \mu + s \quad (2.8)$$

$$i_2 < s - \mu \quad (2.9)$$

$$i_1 > \mu - s \quad (2.10)$$

$$i_2 < \mu + s \quad (2.11)$$

Conditions (2.8) and (2.9) refer to  $p_s^{-+}$ , while conditions (2.10) and (2.11) refer to  $p_s^{-}$ . These conditions are found by requiring the point  $p_s^{-+}$ , which belongs to the saturation region  $\{x \in R^2 : x_1 < -1, x_2 > 1\}$ , to lie in the region  $\{x \in R^2 : x_1 < -1, |x_2| < 1\}$ . Conditions (2.10) and (2.11) are found similarly.

When the trajectory passes near one of these points, slow dynamics is evident. Moreover, this slow dynamics can be modulated by the values of the biases. Acting on  $i_1$ , and respecting conditions (2.8)-(2.11), implies moving both the two equilibria; but while the distance between  $p_s^{-+}$  and the boundary  $x_2 = 1$  is not varied,  $p_s^{-}$  is closer to the boundary  $x_1 = -1$ . This increases the duration of the slow dynamics. This effect is common to systems near bifurcation points as well clarified in [Strogatz (1994)]: the role of these virtual points or *ghosts* is to slow the dynamics of the system when trajectories pass in their neighborhood.

In Fig. 2.3 an example of what happens by keeping  $i_2$  constant and varying the parameter  $i_1$  is shown. In Fig. 2.3(c) the phase plane  $x_1 - x_2$  is represented, emphasizing that by increasing  $i_1$  both  $p_s^{-+}$  and  $p_s^{-}$  move.

These considerations led to the idea of using suitable synaptic laws to modulate the bias values so as to control the dynamics of a population of neurons, each one represented by the dynamical model (2.5).

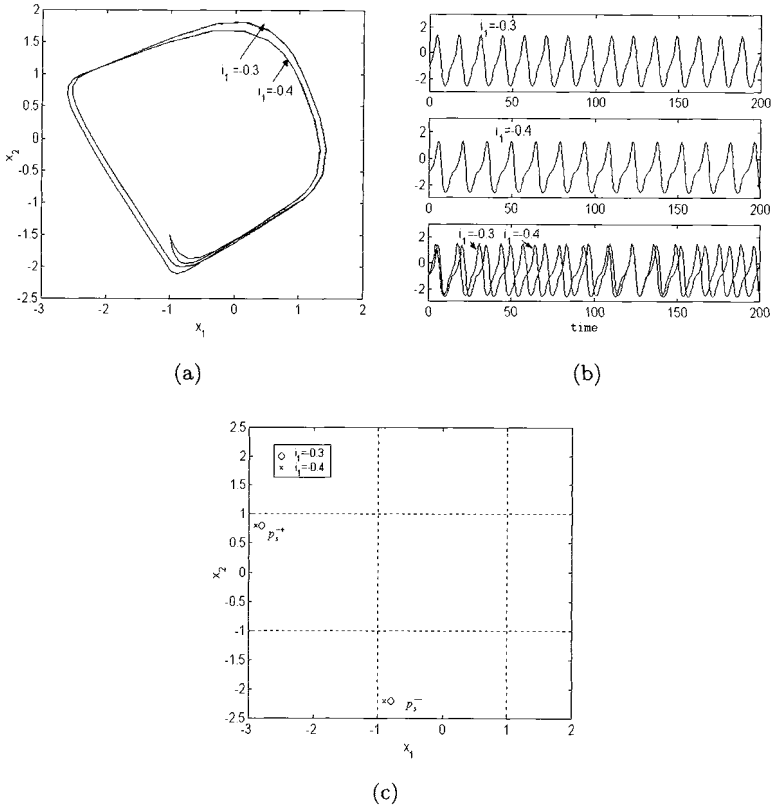


Fig. 2.3 Behavior of the CNN neuron with different values for the bias  $i_1$  (the other parameters are fixed to the values  $\mu = 0.5$ ;  $s = 1$ ;  $i_2 = 0.3$ ): (a) limit cycle in the phase plane  $x_1 - x_2$ ; (b)  $x_1$  versus time; (c) equilibria moving with increasing values of  $i_2$ .

### 2.3.1 The synapse model

A fundamental element in the CPG design is the way in which CNN neurons are connected. Biologists identify two kinds of synapses: electrical and chemical. In this Section we introduce the idea of an approximated CNN chemical synapse. This is derived from the properties of the basic cells and therefore follows the discussion on the CNN neuron. We will show in following Sections that this way of connecting CNN neurons is more general than the RD-CNN approach that corresponds to electrical synapses.

Various mathematical models of the chemical synapse have been proposed: they are either static (see for example [Rabinovich *et al.* (1997)]) or dynamic models (see for example [Huerta *et al.* (2001)]). We will focus

our attention on one of these models. In order to use this kind of synapse in a CNN structure some approximations of the model are required to make it consistent with the CNN paradigm.

The model assumed for the chemical synapse is described in [Rabinovich *et. al.* (1997)], where the so-called Hindmarsh-Rose neuron model [Hindmarsh and Rose (1984)] is considered. This model is represented by a set of three ordinary differential equations, while the CNN neuron (2.5) is a second-order model that does not take into account the recovery time of biological neurons. Obviously, both models have a variable mimicking the cross membrane potential. The mathematical model of the chemical synapse connecting a neuron  $b$  with a neuron  $a$  can be expressed by the following additive term in the differential equation of the cross membrane potential (the variable  $x_1$  in Eq. (2.5)) of neuron  $a$ :

$$\dot{x}_a = g(x_a, y_a, z_a) + \varepsilon \cdot (x_a(t) + V_c) \theta(x_b(t - \tau_c) - X) \quad (2.12)$$

where  $x_a$  and  $x_b$  represent the cross membrane potentials of the two neurons,  $\varepsilon$  is the strength of the coupling,  $V_c$  the reverse potential,  $\tau_c$  the synaptic delay,  $X$  the threshold,  $\theta(\cdot)$  indicates the Heaviside function,  $y_a$  and  $z_a$  are the two other state variables and  $g(x_a, y_a, z_a)$  is the reactive part of the dynamics (omitted for the sake of simplicity).

This synapse is called inhibitory if  $\varepsilon < 0$ , excitatory if  $\varepsilon > 0$ . In the case of inhibition, when neuron  $b$  bursts, it acts on neuron  $a$  with a negative current input, thus hyperpolarizing neuron  $a$  and inhibiting its burst. In the other case (excitatory synapse), when neuron  $b$  bursts, a positive current input due to the activity of neuron  $b$ , depolarizes neuron  $a$ , exciting the burst.

Synapse (2.12) can be simplified to obtain a *CNN chemical synapse*, considering the properties of the reactive part of the CNN neuron in Eq. (2.5). Therefore, the result will depend in some manner on this specific neuron model.

Let us assume that the bias  $i_1$  is not fixed, but is a function of a variable related to another cell (another neuron), and let us assume that this variable is either the variable  $x_1$  or  $x_2$  of the second cell. Let us consider a nonlinear law, such that when the variable is greater than the threshold value  $+1$  an excitatory action is performed by adding a positive term  $\varepsilon$  to the bias and so by increasing the distance between the virtual equilibrium point and its boundary and encouraging the onset of a spike (i.e. reducing the slow part of the trajectory); otherwise, when the synaptic input takes a value

lower than the threshold  $-1$ , the constant term  $\varepsilon$  is subtracted and so the trajectory is slowed down. Let us consider a linear law for the other values of the synaptic input between the two threshold values. This leads the same nonlinearity characterizing the output of the CNN cell; the synaptic influence can depend linearly on the cell output. Thus, the chemical synapse (2.12) can be approximated, in a CNN structure based on cell (2.5), with the following linear additive term depending on the output of neuron  $b$  in the state equation relative to the variable  $x_1$  of neuron  $a$ :

$$\dot{x}_1 = g_1(x_1, x_2) + \varepsilon y_{i,b} \quad (2.13)$$

where again for the sake of brevity the reactive part of the dynamics of the CNN neuron has been indicated with  $g_1(x_1, x_2)$ . We consider Eq. (2.13) as the definition of the simplified CNN chemical synapse.

**Definition 2.4** (Simplified CNN synapse.) Equation (2.13) constitutes a simplified CNN synapse and is called inhibitory if  $\varepsilon < 0$  or excitatory if  $\varepsilon > 0$ .

Either the output of the first layer or that of the second one can be considered; therefore  $i = \{1, 2\}$  in (2.5). When the CNN chemical synapse is realized by using  $y_2$  instead of  $y_1$ , in the case of system (2.5) it is possible to obtain a *simplified delayed chemical synapse*, because in the cell dynamics,  $y_2$  has a phase lag with respect to  $y_1$ . It must be pointed out that the addition of a chemical synapse, in the case of both a delayed and a non-delayed synapse, corresponds to a change in the coefficients of the template  $A$ . Therefore, synaptic weights are set up through the feedback template coefficients.

Let us show an example of a network based on this synapse: this is namely the first and simplest example of CPG presented in this Chapter. Let us consider the flexor-extensor CPG. This is one of the most popular examples because of its simplicity and its importance in biological control of locomotion [Orlovsky *et. al.* (1999)] and is characterized by two neurons firing in out-of-phase synchronization. Such a CPG can be obtained by two mutually inhibited neurons.



Let us consider the system obtained by coupling two CNN neurons with two *inhibitory* CNN chemical synapses as follows:

$$\begin{aligned}
 \dot{x}_{1,a} &= -x_{1,a} + (1 + \mu)y_{1,a} - s \cdot y_{2,a} + i_1 + \varepsilon \cdot y_{1,b} \\
 \dot{x}_{2,a} &= -x_{2,a} + (1 + \mu)y_{2,a} + s \cdot y_{1,a} + i_2 \\
 \dot{x}_{1,b} &= -x_{1,b} + (1 + \mu)y_{1,b} - s \cdot y_{2,b} + i_1 + \varepsilon \cdot y_{1,a} \\
 \dot{x}_{2,b} &= -x_{2,b} + (1 + \mu)y_{2,b} + s \cdot y_{1,b} + i_2
 \end{aligned} \tag{2.14}$$

Figure 2.4 shows a schematic representation of system (2.14) and the result of the simulation with a value of  $\varepsilon = -0.6$  (the other parameters were chosen according to Table 2.1).

The result agrees with the expected anti-phase synchronous behavior that characterizes two mutually inhibited coupled neurons.

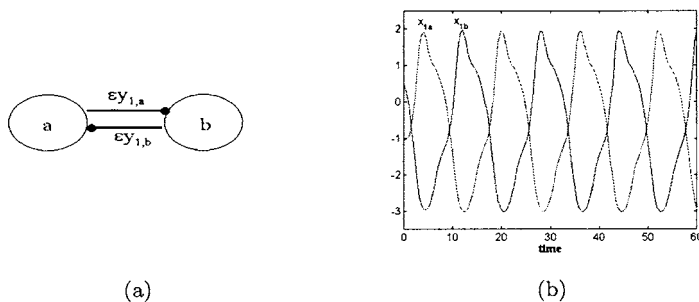


Fig. 2.4 (a) Flexor-extensor model. (b) Simulation of the flexor-extensor ( $x_1$  versus time).

## 2.4 The CNN-based CPG

In this Section it is shown how connections among CNN neurons lead to complex systems able to constitute the CPG for artificial locomotion control. Firstly we discuss the approach based on RD-CNN that is the starting point for the more general approach based on the chemical synapse introduced above, then we show some guidelines for the CNN-based CPG design.

### 2.4.1 RD-CNNs to design artificial locomotion patterns

Reaction-Diffusion (RD) phenomena occur in many systems (biological, chemical, physical systems) [Murray (1993)], but they obey the same laws that can be represented under the following general RD equations:

$$\frac{\partial \mathbf{u}}{\partial t} = F(\mathbf{u}) + D \cdot \nabla^2 \mathbf{u} \quad (2.15)$$

where  $\mathbf{u}$  is a vector of at least two elements. For instance  $\mathbf{u}$  can represent the dynamics of an activator/inhibitor system diffusing in a nonlinear medium and showing pattern formation. Another example is the propagation of *autowaves* in a nonlinear medium. This term was first coined by R. V. Khorhlov to indicate autonomous waves [Krinsky (1984)]. They represent a particular class of nonlinear waves which propagate without forcing in active nonlinear mediums. Autowaves have some typical characteristics, basically different from those of classical waves in conservative systems: their shape remains constant during propagation and reflection and interference do not take place, while diffraction is a common property between classical and autowaves. These autowaves are the basis of the RD-CNN approach.

CNNs are able to show complex phenomena arising in space-distributed fields such as Reaction-Diffusion phenomena in 2D arrays [Arena *et. al.* (1997)]. Examples are autowaves travelling in nonlinear media or Turing patterns generated in CNNs. We will come back to this important topic in Chap. 7. In this Section we introduce a CNN for autowave propagation in a 1D array.

CNNs can map Eq. (2.15) if the diffusion term represented by the Laplacian  $\nabla^2 \mathbf{u}$  is discretized. In a monodimensional field the following discretized Laplacian can be assumed:

$$\nabla^2 \mathbf{u} = u_{i+1} - 2u_i + u_{i-1}$$

As regards the reactive part of Eq. (2.15), namely  $F(\mathbf{u})$ , we assume that it implements the dynamics of the CNN neuron (2.5). This allows to establish an analogy with the biological case: the reactive part of (2.15)  $F(\mathbf{u})$  models the action potential generated by a neural cell.

In this analogy the diffusive term is the equivalent of the synapses between neurons. Thus, the entire CNN can realize a net of diffusive coupled neurons in which the nerve pulse propagates. This kind of coupling represents the equivalent of electric synapses, the next Section will deal with the use of CNN chemical synapses.

Let us thus consider a ring-like structure as follows:

$$\begin{cases} \dot{x}_{1,i} = -x_{1,i} + (1 + \mu)y_{1,i} - sy_{2,i} + i_1 + D_1(y_{1,i+1} - 2y_{1,i} + y_{1,i-1}) \\ \dot{x}_{2,i} = -x_{2,i} + (1 + \mu)y_{2,i} + sy_{1,i} + i_2 + D_2(y_{2,i+1} - 2y_{2,i} + y_{2,i-1}) \end{cases} \quad (2.16)$$

where the discretized Laplacian acts on the output of the CNN,  $i = 1..N$  ( $N$  is the number of CNN neurons in the ring) and periodic boundary conditions are assumed. The key point is that the CNN neuron shows a steady-state dynamics consisting of a stable slowfast limit cycle, while the corresponding RD-CNN generates autowaves. At this point, if the output variables of the CNN neurons constitute the signals driving the actuators of the locomotion system, the CNN plays the same role as the CPG in biological control of locomotion. In fact the wave propagating in the CNN resembles the wave that characterizes several locomotion patterns in nature.

The following definition of RD-CNN for locomotion control can be therefore given.

**Definition 2.5** (RD-CNN for locomotion control.) A RD-CNN for locomotion control is the CNN approximation (2.16) of the RD equation where some of the CNN state variables are used to drive the actuators of the locomotion structure.

The reactive part of the RD-CNN corresponds to the dynamics of the CNN neuron (2.5), while the diffusive part indicates the way in which neurons are connected. They are connected through the outputs of nearest neighbors.

Various locomotion patterns can be implemented using such a structure. The way to change the locomotion pattern is to use a different number of cells constituting the RD-CNN and to rearrange the connections between actuators and motor-neurons (CNN cells). This corresponds in biological terms to a great topological reorganization of the neural network and allows to obtain effector organs synchronized in a way that depend on the number of neurons in the ring. The synapses are reorganized either involving new neurons or decreasing the number of neurons in the network. It is worth remarking that the structure of the connections remains unchanged (thus the feedback template of the CNN is kept constant). When the connections are reorganized to build up the new CPG, the connections between neurons and actuators are also rearranged. Thus a given actuator does not always correspond to a given motor-neuron, but changes according to the locomotion pattern.

The approach depicted in the next Section allows us to have a network in which the connections between actuators and motor-neurons do not depend on the locomotion pattern, as in the biological case. To this end the CNN chemical synapse has been introduced. Diffusive connections play the same role as electrical synapses. The introduction of chemical synapses in our CPG model allows to obtain a network with a given number of cells, showing a much greater variety of behaviors than those considered in the RD-CNN.

#### 2.4.2 Guidelines for CNN-based CPG design

Analogously to the RD-CNN for locomotion control the definition of CNN-based CPG extending the connection paradigm to include simplified CNN synapses can be introduced as follows:

**Definition 2.6** (CNN-based CPG.) A CNN-based CPG is a CNN made of  $n$  CNN neurons (2.5) connected by simplified CNN synapses. Some of the CNN state variables are used to drive the actuators of the locomotion structure.

In this Section some guidelines for the realization of CNN-based CPG are sketched. These heuristic considerations can help to build up networks of neurons that behave in a given way. A network of neurons is made up of coupled nonlinear oscillators. Even with a reduced number of oscillators, the analysis of such a network can be very difficult in the more general case of coupling. For example in [Heagy *et. al.* (1994)] several examples of systems made up of coupled oscillators were analyzed, but only approximate results were given for the case of single coupling. It is also true that analytical results often arise only in the case of global coupling or an infinite number of elementary units. This is not our case, in which a few cells and flexible couplings among them are needed.

To explore the behavior of the network and thus to design a CPG, the problem is split up into a simpler one. If a simple ring-like structure can be assumed to be the core of such networks, then this sub-network can be coupled with other neurons or sub-networks of neurons to achieve the desired CPG.

The CPGs that can be implemented in our approach have the common feature that a one-to-one correspondence between motor-neurons and the moving machine legs is assumed and the connections neighborhood is  $r = 1$ , therefore this will be the neighborhood of the CNN.

The design of such a CPG for a specific locomotion pattern must take some considerations into account. In general, it has to be possible to focus on the distinct phases forming a pattern: at each phase one-leg motion, synchronous motion by two or more than two legs, or no leg motion are allowed. This is fundamental to build up the ring-like structure that is the core of the CPG.

To clarify this consideration, let us introduce the definition of ring of  $N$  neurons.

**Definition 2.7** (Ring of  $N$  neurons.) A *ring of  $N$  neurons* is an array of  $N$  neurons with periodic boundary conditions, each neuron being connected to its neighbor with an excitatory (or inhibitory) synapse in a well-defined direction (clockwise or counter-clockwise).

The importance of these networks is related to the simplicity of their behavior. Simulations carried out indicate that in these networks the firing sequence respects the order of the neurons in the ring and it has the same direction as the synapses if they are excitatory, the opposite direction otherwise. This result agrees with [Wolpert *et. al.* (2000)], in which a ring of an odd number of cells is examined. Moreover, the simulations carried out allow this conjecture to be extended for the first time to rings of even neurons in which delayed synapses are used to connect the CNN neurons: for a given range of coupling ( $\varepsilon$  in the CNN chemical synapse), a travelling wave arises and each neuron fires in a different phase. The result of the self-organization of the network is thus that the neurons oscillate at the same frequency despite parameter variations (a very important point for the hardware implementation) and that between adjacent neurons the phase lag is constant ( $\Delta\phi = \frac{2\pi}{N}$ ).

It should be remarked that while the oscillator model assumed in [Wolpert *et. al.* (2000)] is different, the results obtained are similar to the case treated here, thus confirming the hypothesis [Golubtisky *et. al.* (1998)] that the behavior of the whole system is to a certain extent quite independent of the single oscillator dynamics.

Let us now move from the behavior of simple ring structures to more complex networks. Let us consider a given locomotion pattern (i.e. a given sequence of coordinated leg movements), and let us indicate with  $N$  the number of phases constituting the pattern (for example in the anti-phase synchronization pattern  $N = 2$ ) and with  $n$  the number of movement actuators (for example, legs). We will design a network of  $n$  neurons. These  $n$  neurons will provide the signals to drive the  $n$  movement actuators.

The following steps are required to design a CPG.

- *Create a ring of  $N$  neurons*

The aim of this step is to obtain a network with  $N$  phases that will constitute the core of the CPG. To do this, a ring of a number of neurons equal to the number of the CPG phases, namely  $N$ , is built up. Furthermore, this consideration implies that the smallest ring in the CPG must be the one formed by  $N$  neurons. This condition is necessary due to the fact that the smallest ring establishes the phases of the pattern, but it does not exclude the possibility of having two rings of  $N$  neurons. Of course connections among the  $N$  neurons have to assume that neurons only fire one at a time.

- *Add  $n - N$  neurons and synchronize them with the other ones*

In the second step the  $n - N$  neurons are added and connections with the other neurons are established. These new neurons need to be synchronized with the other ones. Two methods of synchronization are defined: *synchronization via coupling* and *synchronization via duplication*. In the former case, since the coupling establishes the phase in which each neuron fires, the synchronization is achieved by considering two different rings formed by  $k$  neurons and sharing a neuron, as in Fig. 2.5(a) with  $k = 3$ . In this case, considering the shared neuron (A),  $k - 1$  other neurons are “enrolled” in the network; the connections are exactly the same as those among corresponding neurons (B and B’; C and C’ in Fig. 2.5(a)). In the second case synchronization is achieved by exactly duplicating the connections of the neuron that has to be “cloned”, as shown in Fig. 2.5(b). In the latter case new neurons are added (B’ in Fig. 2.5(b)), while in the first case it was possible to synchronize two subnetworks of CNN neurons, that, in principle, can deal with different tasks. However, the results that can be obtained by using these two different approaches are equivalent.

It is worth remarking that the  $n - N$  neurons are added in order to obtain a network of  $n$  neurons, in which a one-to-one correspondence between motor-neurons and movement actuators is established. This mimics the biological case and offers the possibility of having a flexible network able to implement various locomotion patterns, provided there is a suitable change in the synaptic connections.

- *Choose the synaptic weights in the whole network*

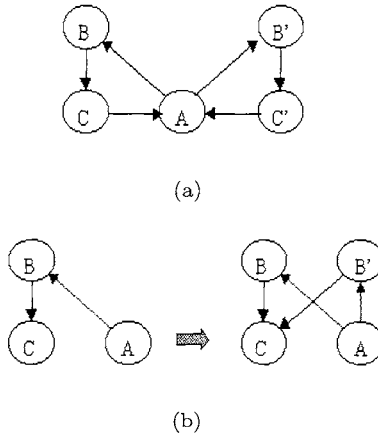


Fig. 2.5 Neuron synchronization via coupling (a) or via duplicating (b).

Finally, the synaptic weights have to be chosen. The choice of the weights is made bearing in mind that the oscillation period depends on them. In our case they were selected by performing numerical simulations of the ring-like core of the CPG and choosing the same weight value for all the synapses in the network. This synaptic weight  $\varepsilon$  is chosen in the first step in order to guarantee the stability of the synchronization pattern in which each neuron fires in a different phase. Then, once the other neurons are connected in the second step of the design, the synaptic weights of these neurons are arranged so that if a neuron has  $k$  synaptic inputs and  $\varepsilon$  is the synaptic weight in the case of a single neuron input, then the weights of these inputs are  $\varepsilon/k$ .

Once established the network structure able to achieve a given locomotion pattern has been established, the set of corresponding templates is written. Template A takes into account the connections between neurons. In general this template is space-dependent, related to the position of the cell to which it applies. Moreover, a different set of templates corresponds to each pattern. Thus, changing the locomotion pattern only requires reloading the set of corresponding templates. Since the number of neurons required to implement such a strategy is often small, it does not seem particularly disadvantageous to have space-variant templates.

Moreover, the whole methodology assumes the role of a spatio-temporal algorithm for locomotion generation, in which a given template set establishes a given locomotion type.

It should be remarked that the considerations presented in this Section give some heuristic guidelines for the design of a CPG and thus constitute one way to design new templates. Moreover, this approach, here called the *Multi-Template Approach (MTA)* [Arena *et. al.* (2002c)], avoids the need to rearrange the connections between actuators and motor-neurons as needed in the RD-CNN, thus simplifying the hardware circuitry. The great flexibility of this method is fundamental, since the new set of templates, corresponding to an arbitrary selected locomotion pattern, can be simply designed by following these guidelines.

## 2.5 Example: the caterpillar gait for hexapods

In this Section the application of the guidelines outlined in the previous Section is illustrated with the help of an example. In the case of animal walking the locomotion pattern is called gait. Even if more general definitions can be given, in the following only periodic gaits will be taken into account. The characteristics of a locomotion gait can be defined through the concepts of *cycle time*, *duty factor* and *phase lag* [Waldron and Song (1989)].

**Definition 2.8** The *cycle time* is the time required for a leg to complete a locomotion cycle.

**Definition 2.9** The *duty factor*  $df_i$  is the time fraction of a cycle time in which the leg  $i$  is in the power stroke phase.

**Definition 2.10** The *phase lag*  $\varphi_i$  is the fraction of a cycle period by which the beginning of the return stroke of leg  $i$  lags behind the beginning of the return stroke of the left front leg, chosen as a reference.

Alternatively, instead of considering the phase lags between each pair of legs, some authors [Pfeiffer *et. al.* (1995)] make use of phase lags between ipsilateral and contralateral legs.

As an example a hexapod gait that following [Golubtisky *et. al.* (1998)] we call “caterpillar gait” is taken into account to illustrate the steps in the design of the corresponding CPG. In this pattern of locomotion the right and left legs move in synchrony: first legs R1 and L1 complete their swing



movement, then legs R2 and L2 move, finally legs R3 and L3 move. Thus legs move in synchrony in a wave passing from front to back; this sequence is inverted with respect to that actually adopted by caterpillars.

The *step pattern* of the caterpillar gait is shown in Fig. 2.6, where the white areas indicate the *stance* phase, i.e. the period when the legs are on the ground and help to move the body, while black areas indicate the *swing* phase, i.e. the period when the legs are lifted. In terms of duty cycle and leg phases the caterpillar gait is characterized by a duty factor equal for all the legs ( $df_i = \frac{2}{3}$ ) and by the following phase lags:

$$\varphi_{L2} = \frac{1}{3}; \varphi_{L3} = \frac{2}{3}; \varphi_{R1} = 0; \varphi_{R2} = \frac{1}{3}; \varphi_{R3} = \frac{2}{3} \quad (2.17)$$

where the legs (and consequently the corresponding neurons) are numbered from front to rear and labelled as left (L) or right (R).

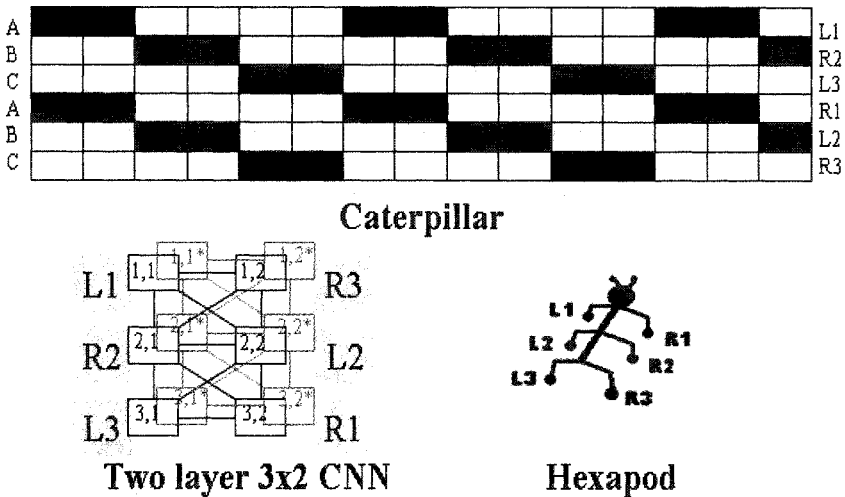


Fig. 2.6 Step pattern of the caterpillar gait and correspondence between legs and CNN neurons. In the step pattern black and white areas represent the swing and stance phases, respectively. Legs (and the corresponding neurons) are numbered from front to rear and labelled as left (L) or right (R).

Figure 2.6 also illustrates the correspondence between legs and CNN neurons. At this point we have also to specify how the variables of the CNN neuron are used to drive a leg of the hexapod robot. In other words, in

view of an hardware implementation the level of interpretation of the state variables of neuron (2.5) needs to be fixed. Generally speaking, depending on the application, one or both the outputs of the neuron to control one or more DOF of a given locomotion structure can be used. However, when used to control a leg with two DOF as in the hexapod robot (whose design is discussed in detail in Chap. 5), the variables  $y_1$  and  $y_2$  are used to control the elevation/flexion ( $\beta$  - joint) and the retraction/protraction ( $\alpha$  - joint) of the leg, respectively (see Fig. 2.7; the  $\gamma$  - joint is kept at a constant value). In detail the relationship between the motor-neuron outputs and the joint variables can be described by the following equation:

$$\begin{cases} \alpha_{ij} = a_{2,ij}y_{2,ij} + b_{2,ij} \\ \beta_{ij} = a_{1,ij}y_{1,ij} + b_{1,ij} \\ \gamma_{ij} = b_{3,ij} \end{cases} \quad (2.18)$$

where the indexes  $ij$  account for the robot legs (R1,R2,...,L3),  $\alpha_{ij}$ ,  $\beta_{ij}$  and  $\gamma_{ij}$  are used to control the joint positions and  $a_{1,ij}..b_{3,ij}$  are constant values that set the joint excursion.

With these assumptions the limit cycle of the CNN neuron (2.5) represents the periodic movement of the leg in the forward direction, and in particular fast dynamics (the action potential) by the motor-neuron corresponds to the swing phase, while slow dynamics represents the stance phase.

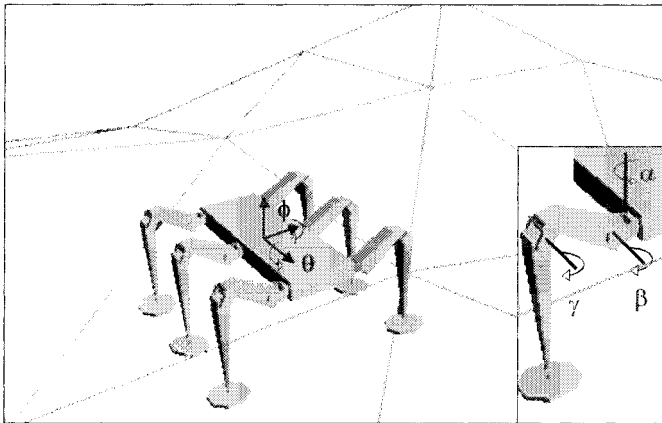


Fig. 2.7 Structure of the hexapod robot.  $\alpha$ ,  $\beta$  and  $\gamma$  indicate the 3 DOF of each leg.  $\theta$  and  $\phi$  indicate roll and pitch angles, respectively.

Figure 2.8 illustrates in a graphical way the various steps in the design of the caterpillar CPG. The hexapod has six legs, thus it needs  $n = 6$  motor-neurons. The pattern has  $N = 3$  phases: the first is characterized by the movement of the first pair of legs, then the middle legs move, and finally the hind legs move.

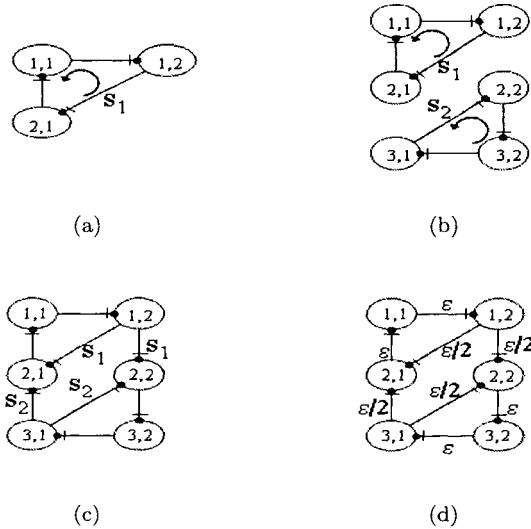


Fig. 2.8 The design of the CPG for the caterpillar gait.

First, a ring of  $N = 3$  neurons is created (Fig. 2.8(a)) and a synaptic value equal to  $\epsilon$  is chosen so that each neuron fires one at a time: according to the inhibitory feature of synapses, the firing sequence follows the arrow outlined in Fig. 2.8(a). Assuming the use of synchronization via duplication, the second step is to consider another ring of  $n - N = 3$  neurons, firing in the same sequence as the first ring (Fig. 2.8(b)). Subsequently, if neurons (2,2) and (2,1) have to fire synchronously, they must possess the same synaptic inputs. Therefore, neuron (2,2) has to be connected to neuron (1,2) with synapse  $s_1$ . Similarly, neuron (2,1) must have the synaptic input  $s_2$  like (2,2) from neuron (3,1), in order to fire synchronously with (2,2) (Fig. 2.8(c)).

If  $\epsilon$  indicates the synaptic weight chosen in the first design phase, the final step is to rearrange the synaptic weights of the whole network taking into consideration the new connections after the phase in which the  $n - N$

neurons are added: since each neuron must have a total synaptic input equal to  $\epsilon$ , the final synaptic values are chosen as in Fig. 2.8(d).

The complete firing sequence is shown in Fig. 2.9(a), which, adopting the cell-leg connections as depicted in Fig. 2.6, exactly matches the caterpillar gait.

Note that the displacement of the neurons does not reflect the leg position. The great advantage is that, by adopting this configuration, distinct hexapod gaits in a CNN structure with neighborhood  $r = 1$  can be implemented in the same configuration, only changing the synaptic connections (i.e. the templates).

The circuit realizing the core of the CPG (the CNN neuron) is described in Appendix B. Connections among the six motor-neurons are rearranged in order to implement the feedback templates corresponding to the caterpillar gait as follows  $\epsilon_c = -0.6$ :

$$\begin{aligned}
 A_{11}^{i,j} = A_{22}^{i,j} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 + \mu & 0 \\ 0 & 0 & 0 \end{pmatrix}; A_{21}^{i,j} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{pmatrix}; A_{12}^{1,1} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -s & 0 \\ 0 & \epsilon_c & 0 \end{pmatrix}; \\
 A_{12}^{1,2} &= \begin{pmatrix} 0 & 0 & 0 \\ \epsilon_c & -s & 0 \\ 0 & 0 & 0 \end{pmatrix}; A_{12}^{2,1} = \begin{pmatrix} 0 & 0 & \epsilon_c/2 \\ 0 & -s & 0 \\ 0 & \epsilon_c/2 & 0 \end{pmatrix}; A_{12}^{2,2} = \begin{pmatrix} 0 & \epsilon_c/2 & 0 \\ 0 & -s & 0 \\ \epsilon_c & 0 & 0 \end{pmatrix}; \\
 A_{12}^{3,1} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & -s & \epsilon_c \\ 0 & 0 & 0 \end{pmatrix}; A_{12}^{3,2} = \begin{pmatrix} 0 & \epsilon_c & 0 \\ 0 & -s & 0 \\ 0 & 0 & 0 \end{pmatrix}.
 \end{aligned}$$

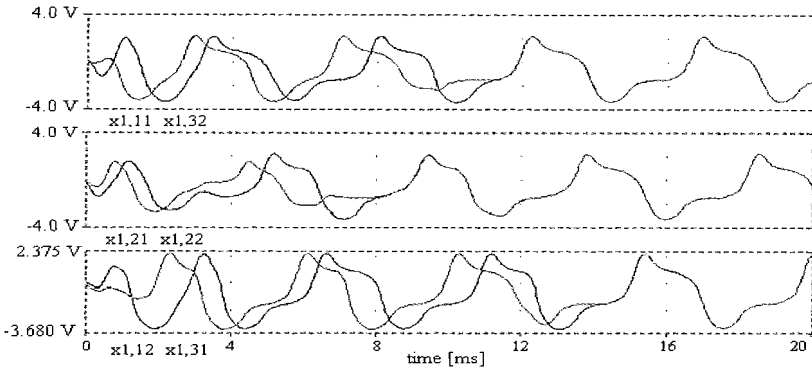
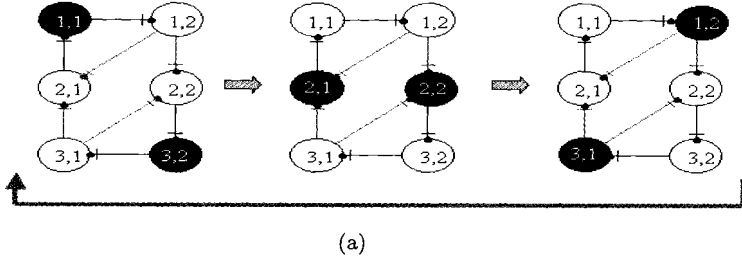
As the system is autonomous, the template  $B$  is zero.

In Fig. 2.9(b) the trend of the six neuron outputs ( $x_1$  of the CPG circuit simulated by SPICE) is shown. The outputs are synchronized as required in the caterpillar gait to generate the signals driving the legs.

## 2.6 A spatio-temporal algorithm for controlling locomotion of a hexapod robot

We focus now on a specific example, the design of a CPG for a hexapod robot that accounts for the several different gaits that hexapods adopt in different environmental conditions. We discuss an algorithm accounting for the generation of distinct gaits and show simulation results.

Hexapod walkers typically adopt several gaits characterized by differ-



(b)

Fig. 2.9 Firing sequence (a) and SPICE simulation ( $x_1$  of each motor-neuron versus time) (b) for the caterpillar CPG.

ent speeds: at low speeds the so-called *slow gait* or *wave gait* is adopted, *medium gait* emerges at intermediate speeds, and *alternating tripod* or *fast gait* is adopted at high speeds [Collins and Stewart (1993b); Pearson (1976); Wilson (1966)].

Using the parameters previously defined of duty cycle and phase lags, it is possible to characterize fast gait as a periodic gait with the same duty factor for all the legs ( $df_i = 0.5$ ) and the following phase lags:

$$\varphi_{L2} = \frac{1}{2}; \varphi_{L3} = 0; \varphi_{R1} = \frac{1}{2}; \varphi_{R2} = 0; \varphi_{R3} = \frac{1}{2} \quad (2.19)$$

For medium gait the duty factor is  $df_i = \frac{5}{8}$  and the phase lags are as follows:

$$\varphi_{L2} = \frac{3}{4}; \varphi_{L3} = \frac{2}{4}; \varphi_{R1} = \frac{2}{4}; \varphi_{R2} = \frac{1}{4}; \varphi_{R3} = 0. \quad (2.20)$$

For slow gait the duty factor is  $df_i = \frac{9}{12}$  and the phase lags are as follows:

$$\varphi_{L2} = \frac{4}{6}; \varphi_{L3} = \frac{2}{6}; \varphi_{R1} = \frac{3}{6}; \varphi_{R2} = \frac{1}{6}; \varphi_{R3} = \frac{5}{6}. \quad (2.21)$$

These three gaits are obtained by choosing the connections among the six neurons as in Fig. 2.10. Since the symbols adopted are not standard, it should be pointed out that each circle represents a CPG cell as in Eq. (2.5) controlling a hexapod leg. Moreover, the connections indicate the synapses realizing the CPG scheme: inhibitory synapses (i.e. with a negative sign) terminate with a dot, while excitatory ones terminate with an arrow.

The possibility to implement the three gaits with templates gives us the opportunity to build a spatio-temporal algorithm based on templates and running on a CNN structure. This algorithm is able to select the most suitable gait pattern based, for example, on sensor information. Therefore the whole robot locomotion generation and control is driven by a complex spatio-temporal dynamics, where each template subroutine is devoted to controlling the robot in particular environmental conditions.

The hexapod robot discussed in [Arena *et. al.* (1999)] is controlled by the RD-CNN CPG, where the slow, medium and fast gait differ in the number of the cells involved in the ring. If simplified chemical synapses are used instead of diffusive synapses and so if the approach depicted in Sec. 2.4.2 is adopted for the CPG realization, three different template sets will be obtained. Indeed, the templates depend on the cell (they are space-variant) and on the pattern. On the other hand, the CNN able to generate and control the whole locomotion dynamics is often made up of a low number of neurons. Therefore the space-variant template approach does not constitute a drawback. Moreover, the advantage of this approach is that the network structure, as well as all the cell-leg connections are fixed. A particular gait to be implemented depends only on the template values. Therefore the locomotion problem is solved through a fixed hardware structure adaptively controlled by a spatio-temporal algorithm.

Simulation results are presented in Fig. 2.10. The scheme of the three gaits (slow, medium and fast gait), the corresponding CNN-based CPG and waveforms of the state variables of the CNN neurons are shown. The corresponding templates that can be easily derived from the CPG schemes of Fig. 2.10 are summarized in Table 2.2. For example, in Fig. 2.10(a) neuron (1,1) in the MTA diagram exactly follows the fast gait templates

of Table 2.2. It is, in fact, driven by the template coefficients. The absence of connections in the MTA diagram indicates zero coefficients in the corresponding template.

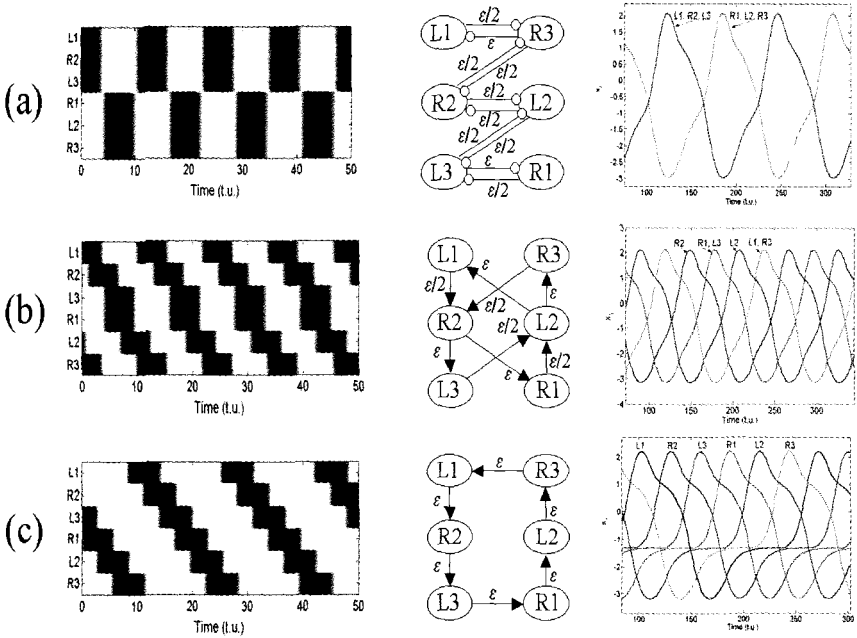


Fig. 2.10 Gait schemes, CNN-based CPG and waveforms of (a) CPG for fast gait; (b) CPG for medium gait; (c) CPG for slow gait. The step patterns are shown on the left and their correspondence to the MTA CPG is shown in the middle. For example, in Fig. 2.10(a) neuron 1,1 in the MTA diagram exactly follows the fast gait templates of Table 2.2. It is, in fact, driven by the template coefficients. The absence of connections in the MTA diagram indicates zero coefficients in the corresponding template. Excitation and inhibition, corresponding to positive and negative template coefficients, are shown in the MTA diagram as arrows and circles respectively. Delayed synapses are assumed for medium and slow gait.

Even if locomotion gaits are often regarded as distinct, the locomotion control for instance in insects [Graham (1985)] is much more complex: the transition between two gaits is continuous and leads to a class of intermediate gaits. We briefly discuss how to extend the spatio-temporal algorithm for the control of hexapod locomotion to account for continuous transition

between gaits.

Behavioral approaches based on CPG (for instance [Klaassen *et. al.* (2002)] and [Ayers *et. al.* (1998)]) often consider a set of basic motion patterns dealing with different forms of behavior, for example to account for forward locomotion and left and right turning patterns. These basic patterns can be simultaneously stimulated and combined to form the desired pattern. These considerations inspired the idea of fading the distinct basic gaits (i.e. slow, medium and fast) to obtain a gait able to show continuous transitions between them. The idea consists of activating more than one set of templates at the same time, i.e. as the gait template we consider a weighted sum of the basic templates for fast, medium and slow gait in Table 2.2. However, an important difference from the behavioral approach emerges. In [Kirchner (1997)] distinct behavioral modules are simultaneously activated, while in our case we define a new set of connections. In other words, we do not sum the effects of two or more distinct units, but use a dynamical CPG network to generate the locomotion gait.

More precisely, let us indicate with  $A^{ij,fg}$  the set of templates associated with the fast gait (Table 2.2) as follows:

$$A^{ij,fg} = \begin{pmatrix} A_{11}^{ij,fg} & A_{12}^{ij,fg} \\ A_{21}^{ij,fg} & A_{22}^{ij,fg} \end{pmatrix}$$

and analogously with  $A^{ij,sg}$  and  $A^{ij,mg}$  the templates for slow and medium gait, respectively. The following definition can be given:

**Definition 2.11** The *continuous generalized gait* (CG gait) is defined through its feedback templates as follows:

$$A^{ij,cgg} = \alpha A^{ij,fg} + \beta A^{ij,mg} + \gamma A^{ij,sg} \quad (2.22)$$

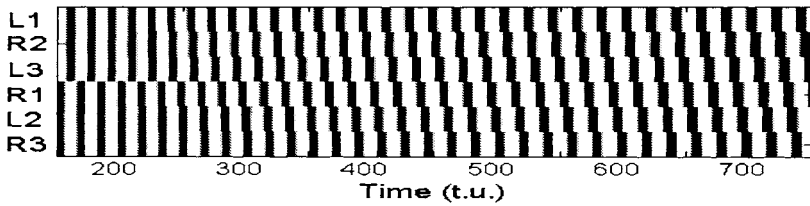
where  $\alpha$ ,  $\beta$  and  $\gamma$  are control parameters having values between 0 and 1 and whose sum is 1, i.e. they should respect the following constraints:

$$0 \leq \alpha, \beta, \gamma \leq 1 \quad (2.23)$$

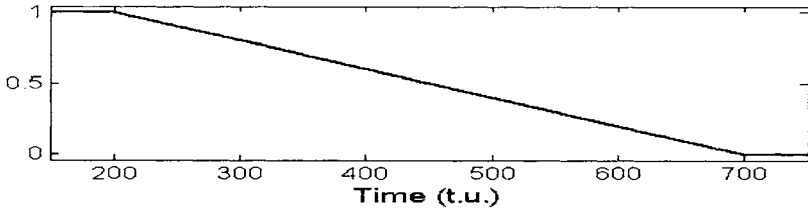
$$\alpha + \beta + \gamma = 1 \quad (2.24)$$



When  $\alpha = 1$  and  $\beta = \gamma = 0$  the CG gait clearly matches fast gait, while when  $\alpha = \gamma = 0$  and  $\beta = 1$  the CG gait corresponds to medium gait and when  $\alpha = \beta = 0$  and  $\gamma = 1$  we have slow gait. In Fig. 2.11 an example is given, the control parameter  $\alpha$  is changed from 1 to zero, correspondingly  $\gamma$  is varied according to  $\gamma = 1 - \alpha$  and  $\beta = 0$ . In this way the gait changes from fast to slow gait with continuity. The simulation results show a smooth transition between fast and slow gait.



(a)



(b)

Fig. 2.11 Behavior of the CG gait. (a) Transition from fast gait to slow gait. (b) Values of the control parameter  $\alpha$  ( $\beta = 0$  and  $\gamma = 1 - \alpha$ ).

It is worth remarking that the approach presented can be extended to structures with a larger number of legs (or other bio-inspired actuators [Arena *et. al.* (2002d)]) by following the guidelines sketched in Sec. 2.4.2. As an example of application of the control strategy in Chap. 5 a hexapod robot controlled by a CNN-based CPG is illustrated.

## 2.7 Motor-neurons and inter-neurons

The neurons belonging to the CPG network have been referred to in this Chapter as motor-neurons. The reason for this is that they are directly used to drive the legs of the hexapod robot. Many biological models also include

inter-neurons: these neurons are involved in the generation of the rhythm of the locomotion and are connected to the motor-neurons that directly synapse the muscles. The approach presented here is easily generalized to include this scheme: it can be assumed that the neurons of the CPG simply give the rhythm for the locomotion system, while other motor-neurons are included in a hierarchical structure, receiving signals from the CPG and driving the actuators. The approach depicted is therefore suitable for the control of a more general locomotion system, with a given number of  $n$  actuators, driven by  $n$  motor-neurons of a CPG with  $N$  inter-neurons. Moreover, the assumption that the neurons of the network are identical can be removed.

Therefore, two generalizations leading to a more flexible CPG design can be introduced:

- including inter-neurons;
- considering neurons with different parameters.

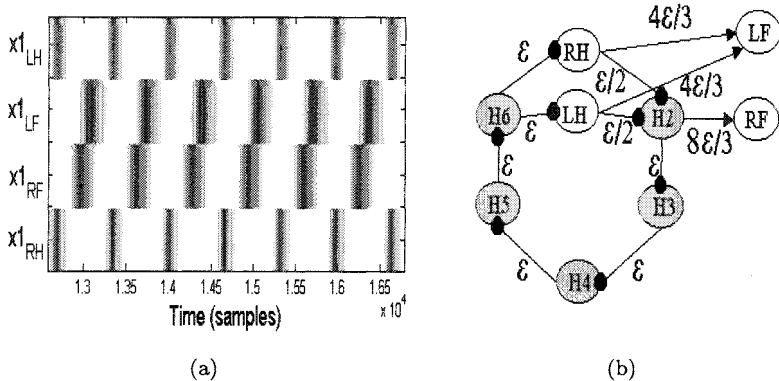


Fig. 2.12 Half-bound of the hare: (a) the locomotion pattern, obtained by plotting on a gray scale the trends of  $x_{1,i}\theta(x_{1,i})$  ( $\theta(x)$  is the Heaviside function); (b) the CPG network: H2..H6 are inter-neurons as in Eq. (2.5), RH and LH are motor-neurons as in Eq. (2.5), while LF and RF are motor-neurons as in Eq. (2.25).

An example of such a CPG is illustrated below. The half-bound of the hare [Collins and Stewart (1993a)] is a locomotion pattern in which the back legs move synchronously and the front legs are half a period out of phase with the back pair, and slightly out of phase with one another. This pattern is adopted by some animals at galloping speeds (for instance,

squirrels). To realize this pattern a network with a ring of six neurons has been considered. The network includes five neurons and four motor-neurons, as shown in Fig. 2.12(b): the inter-neurons are labelled  $H2..H6$ , while the motor-neurons are  $LF$  (left front leg),  $RH$  (right hind leg),  $LH$  and  $RF$ . The motor-neurons associated with the front legs are slightly different. They can be expressed by the following equations:

$$\begin{cases} \dot{x}_{1,ij} = \frac{1}{2}(-x_{1,ij} + (1 + \mu)y_{1,ij} - sy_{2,ij}) \\ \dot{x}_{2,ij} = \frac{1}{2}(-x_{2,ij} + (1 + \mu)y_{2,ij} + sy_{1,ij}) \end{cases} \quad (2.25)$$

The factor  $\frac{1}{2}$ , multiplying the right hand side of both Eq. (2.25), scales the dynamics of the system making it slower than system (2.5). The locomotion pattern shown in Fig. 2.12(a) is very similar to the half-bound of the hare reported in [Collins and Stewart (1993a)]. The waveforms of variable  $x_{1,i}$  for the four motor-neurons of the CPG are shown in Fig. 2.13.

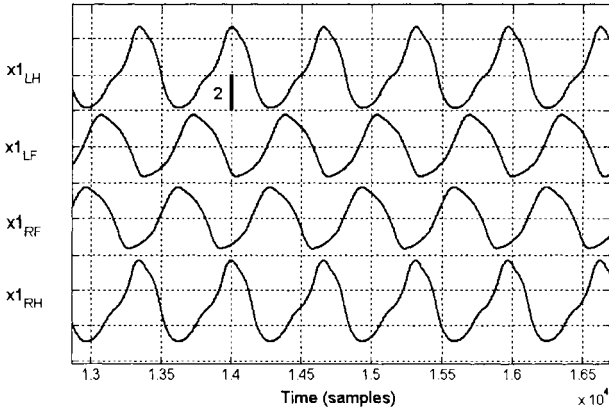


Fig. 2.13 Waveforms of variable  $x_{1,i}$  for the 4 motor-neurons of the CPG generating the half-bound of the hare.

Finally, it is worth remarking that the MTA-CNN for real-time locomotion control in bio-inspired robots allows us to overcome a limit of the RD-CNN implementation for CPGs: the absence of chemical synaptic input which prevented us from obtaining different locomotion patterns without varying the CNN structure. The simplified CNN chemical synapse introduced is the core of a more general class of CNN-based CPGs mimicking the behavior of the neural systems involved in locomotion of a very large

variety of beings. The most evident consequence of this generalization is the necessity to have templates that depend on the position of the cell instead of space-invariant templates. On the other hand, the low number of neurons of the CNN controlling the locomotion makes this drawback almost negligible. The way in which the kind of locomotion changes is now completely different from the previous one: it is not necessary to vary the topology of the neurons involved as well as the cell-actuator connections (as happens in RD-CNN), but it simply implies a new set of templates establishing synaptic connections. Indeed, the generality of the approach is evident, since it does not exclude a reorganization of the network to include other neurons not yet involved in the generation of the locomotion pattern. The problem of building up a new CPG is thus reconnected to a choice of new templates. This choice can be made by following the guidelines remarked in Sec. 2.4.2 and the considerations discussed above.

Table 2.2 MTA templates for different gaits.

Locomotion Pattern	Feedback templates		
FAST GAIT $\epsilon_f = -0.6$	$A_{11}^{1,1} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 + \mu & \epsilon_f \\ 0 & 0 & 0 \end{pmatrix}$ $A_{11}^{2,2} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{\epsilon_f}{2} & 1 + \mu & 0 \\ \frac{\epsilon_f}{2} & 0 & 0 \end{pmatrix}$ $A_{12}^{i,j} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -s & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$A_{11}^{1,2} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{\epsilon_f}{2} & 1 + \mu & 0 \\ \frac{\epsilon_f}{2} & 0 & 0 \end{pmatrix}$ $A_{11}^{3,1} = \begin{pmatrix} 0 & 0 & \frac{\epsilon_f}{2} \\ 0 & 1 + \mu & \frac{\epsilon_f}{2} \\ 0 & 0 & 0 \end{pmatrix}$ $A_{21}^{i,j} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$A_{11}^{2,1} = \begin{pmatrix} 0 & 0 & \frac{\epsilon_f}{2} \\ 0 & 1 + \mu & \frac{\epsilon_f}{2} \\ 0 & 0 & 0 \end{pmatrix}$ $A_{11}^{3,2} = \begin{pmatrix} 0 & 0 & 0 \\ \epsilon_f & 1 + \mu & 0 \\ 0 & 0 & 0 \end{pmatrix}$ $A_{22}^{i,j} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 + \mu & 0 \\ 0 & 0 & 0 \end{pmatrix}$
MEDIUM GAIT $\epsilon_m = 0.66$	$A_{11}^{i,j} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 + \mu & 0 \\ 0 & 0 & 0 \end{pmatrix}$ $A_{12}^{2,1} = \begin{pmatrix} 0 & \frac{\epsilon_m}{2} & \frac{\epsilon_m}{2} \\ 0 & -s & 0 \\ 0 & 0 & 0 \end{pmatrix}$ $A_{12}^{3,2} = \begin{pmatrix} \epsilon_m & 0 & 0 \\ 0 & -s & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$A_{12}^{1,1} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -s & 0 \\ 0 & 0 & \epsilon_m \end{pmatrix}$ $A_{12}^{2,2} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -s & 0 \\ \frac{\epsilon_m}{2} & \frac{\epsilon_m}{2} & 0 \end{pmatrix}$ $A_{21}^{i,j} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$A_{12}^{1,2} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -s & 0 \\ 0 & \epsilon_m & 0 \end{pmatrix}$ $A_{12}^{3,1} = \begin{pmatrix} 0 & \epsilon_m & 0 \\ 0 & -s & 0 \\ 0 & 0 & 0 \end{pmatrix}$ $A_{22}^{i,j} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 + \mu & 0 \\ 0 & 0 & 0 \end{pmatrix}$
SLOW GAIT $\epsilon_s = 0.72$	$A_{11}^{i,j} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 + \mu & 0 \\ 0 & 0 & 0 \end{pmatrix}$ $A_{12}^{2,1} = \begin{pmatrix} 0 & \epsilon_s & 0 \\ 0 & -s & 0 \\ 0 & 0 & 0 \end{pmatrix}$ $A_{12}^{3,2} = \begin{pmatrix} 0 & 0 & 0 \\ \epsilon_s & -s & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$A_{12}^{1,1} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -s & \epsilon_s \\ 0 & 0 & 0 \end{pmatrix}$ $A_{12}^{2,2} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -s & 0 \\ 0 & \epsilon_s & 0 \end{pmatrix}$ $A_{21}^{i,j} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$A_{12}^{1,2} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -s & 0 \\ 0 & \epsilon_s & 0 \end{pmatrix}$ $A_{12}^{3,1} = \begin{pmatrix} 0 & \epsilon_s & 0 \\ 0 & -s & 0 \\ 0 & 0 & 0 \end{pmatrix}$ $A_{22}^{i,j} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 + \mu & 0 \\ 0 & 0 & 0 \end{pmatrix}$

This page intentionally left blank

## Chapter 3

# CNN-based CPGs with sensory feedback and VLSI implementation

The role of feedback in the CPG for walking is essential to deal with complex environments. In this Chapter the topic of including feedback from sensors in the CNN-based CPG is addressed: an approach based on a local bifurcation of the CNN cells constituting the sub-units of the CPG network is introduced, allowing the direction of the robot to be controlled. The dynamics near the bifurcation is furthermore exploited to include feedback from ground contact signals in the CPG. Moreover, the speed of the pattern can be controlled in a very simple way by adopting a switched-capacitor technique to implement the CNN-based CPG. This leads to the so-called hybrid approach, in which the core of the control is analog and a digital control provides the signals modulating the outputs of the analog core. At the end of this Section experimental results obtained with the first prototype of the CNN-based CPG chip are discussed.

### 3.1 Direction control

The approach described here aims to include feedback in the CNN-based CPG and relies on a local bifurcation of the CNN neurons constituting the sub-units of the CPG network. Suitable control can be achieved by changing the value of the bias of the CNN neurons, according to a strategy inspired by the idea of Braitenberg creatures [Arkin (1998); Braitenberg (1984)].

Braitenberg showed how surprisingly complex behavior (that he compared with cowardice, aggression, and other forms of human behavior) can be obtained by using relatively simple direct couplings between the sensors and motors of vehicles. These couplings can be either inhibitory or excitatory depending on the “personality” of the creature. An example is given

where this approach is successfully applied to build up a purely reactive hexapod able to avoid obstacles.

### 3.1.1 The CPG cell

In this Section it is shown that, when the bias is varied, the dynamics of each CNN neuron of the CPG changes from a limit cycle to a stable fixed point.

This property is used, as shown in Sec. 3.1.2, to include feedback signals and to allow the robot to change the direction of its motion.

The equations of the CPG cell are the following:

$$\begin{cases} \dot{x}_1 = -x_1 + (1 + \mu)y_1 - sy_2 + i_1 \\ \dot{x}_2 = -x_2 + sy_1 + (1 + \mu)y_2 + i_2 \end{cases} \quad (3.1)$$

where  $y_i = \tanh(2x_i)$  with  $i = \{1, 2\}$ . System (3.1) is analogous to system (2.5). Below we will refer to Eq. (3.1) because of its smoothness. Moreover, the circuit will implement the smooth nonlinearity. The results are qualitatively the same and the same considerations hold both for neuron (2.5) and neuron (3.1).

Like neuron (2.5) for the choice of parameters reported in Table 3.1, system (3.1) admits a periodic solution with slow-fast dynamics. Moreover, as shown below, this limit cycle disappears when the bias  $i_2$  is varied.

Table 3.1 Parameters of system (3.1).

$\mu$	$s$	$i_1$	$i_2$	$i_c$
0.5	1	-0.3	0.3	0.3484

**Proposition 3.1** *The global behavior of system (3.1) with the parameters in Table 3.1 changes from a periodic solution to a stable fixed point, when  $i_2 \geq i_c$ .*

**Proof.** The behavior of system (3.1) is analyzed by means of the Poincaré-Bendixson theorem [Strogatz (1994)]. The dynamical system (3.1) has an instable point (the origin when the biases are zeros; a point close to the origin otherwise) that is surrounded by a stable limit cycle for  $i_2 < i_c$ . Indeed no other fixed point exists in the phase plane  $x_1 - x_2$  and since the flux is bounded, the hypotheses of the theorem are verified, thus proving the existence of the stable limit cycle.



When the bias  $i_2$  is changed, other equilibria can arise. In this case the hypotheses of the Poincaré-Bendixson theorem may no longer be valid. For this reason we will investigate the equilibria of system (3.1) when the bias  $i_2$  changes.

With  $i_2 = i_c$  there is a bifurcation of the equilibrium points. Two other fixed points arise for  $i_2 > i_c$ , one stable and one unstable. This appears in Fig. 3.1 showing the nullclines (the curves where either  $\dot{x}_1 = 0$  or  $\dot{x}_2 = 0$ ) of the system. While in Fig. 3.1(a) (where  $i_2 < i_c$ ) the nullclines intersect in only one equilibrium point (unstable), for  $i_2 > i_c$ , as shown in Fig. 3.1(c), three equilibrium points exist. Figure 3.1(b) shows the case where  $i_2 \simeq i_c$ .

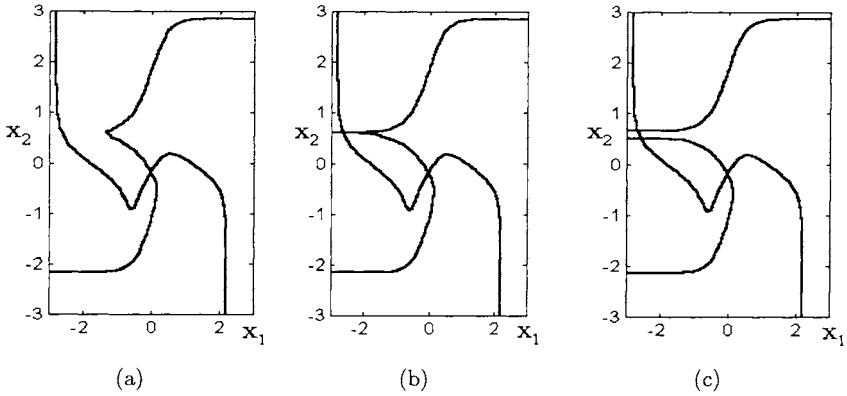


Fig. 3.1 Plane  $x_1 - x_2$  showing the nullclines for system (3.1) with different values for the parameter  $i_2$ . (a)  $i_2 = 0.34$ ; (b)  $i_2 = 0.35$ ; (c)  $i_2 = 0.36$ .

The value of  $i_c$  can be calculated as follows. We examine the behavior of the cell near the critical point. As can be noticed in Fig. 3.1, the bifurcation occurs in a region in which the variable  $x_1$  is less than  $-1$ . Thus we can assume that the output  $y_1$  is saturated at the value  $-1$ . In this approximation system (3.1) can be rewritten as follows:

$$\begin{cases} \frac{dx_1}{dt} = -x_1 - (1 + \mu) - s \cdot y_2 + i_1 \\ \frac{dx_2}{dt} = -x_2 + (1 + \mu) \cdot y_2 - s + i_2 \end{cases} \quad (3.2)$$

The Jacobian of the system (3.2) is:

$$J_x = \begin{pmatrix} -1 & \frac{-2s}{\cosh^2(2x_2)} \\ 0 & \frac{2(1+\mu)}{\cosh^2(2x_2)} - 1 \end{pmatrix}$$

The first eigenvalue of  $J_x$  is  $\lambda_1 = -1$ . The bifurcation occurs when  $\lambda_2$  is zero. The values of the equilibrium point and the critical parameter can be found by verifying this condition.

First, it is verified that  $\lambda_2 = 0$ . This holds for a given equilibrium point that can be computed as follows:

$$\frac{2(1+\mu)}{\cosh^2(2\bar{x}_2)} - 1 = 0$$

With the parameters in Table 3.1  $\bar{x}_2$  is derived:

$$\bar{x}_2 = \frac{1}{2} \cosh^{-1}(\sqrt{2(1+\mu)}) = 0.5731$$

Then, it should be verified that the point  $\bar{x}_2$  is an equilibrium for system (3.2). This is true if the following relation (from which we find the value  $i_c$ ) holds:

$$\dot{x}_2 = 0 \Rightarrow -\bar{x}_2 + (1+\mu) \cdot \tanh(2\bar{x}_2) - s + i_c = 0 \Rightarrow i_c = 0.3484$$

To further prove our assertion, we can observe that, since the eigenvalue associated with the first equation in system (3.2) is negative and the second equation does not depend on  $x_1$ , we can apply the center manifold theorem [Kuznetsov (1998)] and investigate the behavior of the system by studying the behavior of the second equation. Let us rewrite it as follows:

$$\dot{x}_2 = f(x_2, i_2)$$

The genericity conditions of the fold bifurcation [Kuznetsov (1998)] for the second equation of system (3.2) are:

$$\left. \frac{\partial^2 f}{\partial x_2^2} \right|_{\bar{x}_2, i_c} = -\frac{8(1+\mu) \tanh(2\bar{x}_2)}{\cosh^2(2\bar{x}_2)} = -6.5321 \neq 0 \quad (3.3)$$

$$\left. \frac{\partial f}{\partial i_2} \right|_{\bar{x}_2, i_c} = 1 \neq 0 \quad (3.4)$$

The non-zero values of relations (3.3) and (3.4) prove that there is a local bifurcation of the equilibria of system (3.1). This consideration and the Poincaré-Bendixson theorem allow us to draw conclusions on the global behavior of the system. When  $i_2 > i_c$  the limit cycle disappears and the trajectories go towards the stable equilibrium.  $\square$

**Remark 3.1** *Near the bifurcation there is a bottleneck due to the “ghost” (or virtual) equilibrium point (see [Strogatz (1994)] and the discussion in Chap. 2). This leads to a slow-fast ratio in the system that depends on the value of the bias  $i_2$ . Figure 3.2 shows how the period  $T$  of oscillation depends on the parameter  $i_2$ : the function  $T(i_2) \approx \frac{10}{(i_c - i_2)^{1/3}}$  (continuous line) is also shown for comparison; it has a similar form to that reported in [Strogatz (1994)].*

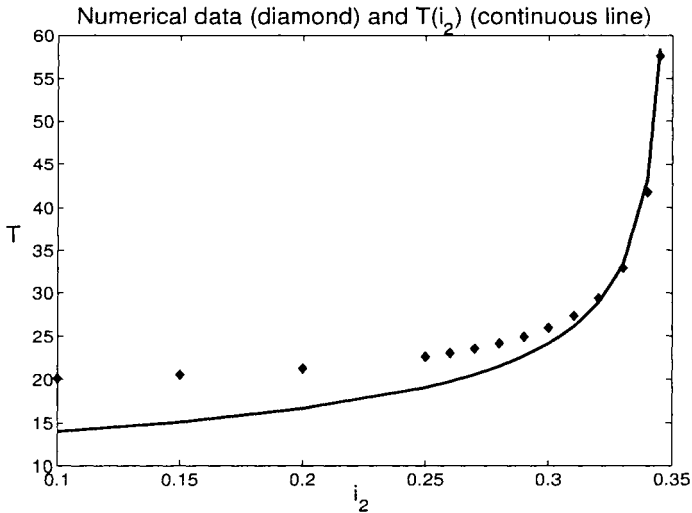


Fig. 3.2 Period of oscillation of system (3.1) versus  $i_2$ . As  $i_2$  approaches the critical value  $i_c$ , the period  $T$  increases until the periodic solution disappears. The function  $T(i_2) \approx \frac{10}{(i_c - i_2)^{1/3}}$  (continuous line) is also shown for comparison.

### 3.1.2 CPG with sensory feedback for direction control

In this Section the design of a CNN-based CPG with sensory feedback is discussed with an example. A CPG for a hexapod robot including feedback

signals to avoid obstacles is designed.

The robot head has been endowed with two sensors, as shown in Fig. 3.3(a) with the CPG scheme of the robot. Each circle represents a CPG cell, as in equation (3.1) with an appropriately scaled time unit, controlling a hexapod leg. The connections in dashed lines indicate the synapses realizing the CPG scheme, while the connections in continuous lines close the feedback loop.

The sensors detect the distance from the obstacles and are directly coupled with the motor controllers (in this case the CPG cells). This approach was inspired by the well-known Braitenberg vehicles [Braitenberg (1984)], where the sensors are directly coupled to the motors. In our case the coupling is realized with connections between sensors and CNN motor-neurons that are also connected with each other. The suitability of the approach is guaranteed by the fact that the sensor feedback modifies the locomotion pattern without dramatic changes in the behavior of the cells not directly coupled to the sensors.

To avoid obstacles, the configuration shown in Fig. 3.3 was assumed: the left sensor is connected with an inhibitory coupling to the CPG cell controlling the middle right leg (R2), while the right sensor is connected to the middle left leg (L2). The sensor state establishes the value of the bias of the CNN neuron: thus, when the sensor detects an obstacle (i.e. when the distance from an object is less than a given threshold), it inhibits oscillation of the CPG cell by setting  $i_2 > i_c$ . As a result the corresponding leg is blocked and the robot turns in one direction.

The example discussed here deals with phobic behavior. To implement other kinds of behaviors, for example to include some attracting source in the obstacle-avoiding control system, it is sufficient to reverse the wires connecting the sensors and motors.

The behavior of the system has been tested by using a hexapod model in a VisualNastran environment. A scenario with two sidewall obstacles was used to test the performance of the system. The hexapod walks in the passage between the two sidewalls with the trajectory shown in Fig. 3.4.

Figure 3.5 shows the signals from the CPG. The state  $x_1$  of each of the CPG cells is given as well as the sensor signals  $Out_{RX}$  and  $Out_{LX}$  (for the right and left middle leg, respectively). These signals indicate when the leg should be stopped and depend on the output of the opposite side sensor (as in Fig. 3.3); a digital circuit prevents both signals from taking high values at the same time.

As can be noticed, when the middle right leg (R2) is stopped, large

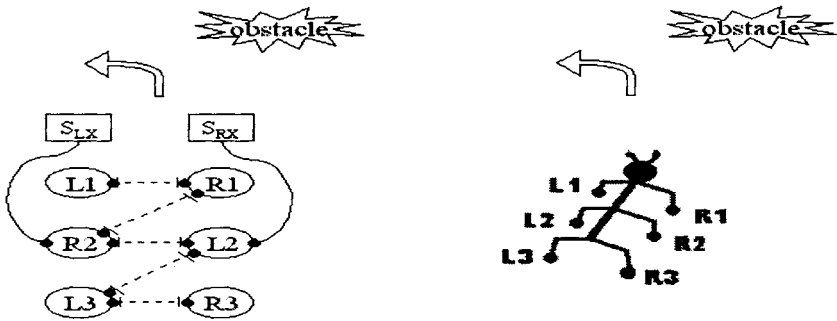


Fig. 3.3 Scheme of the CNN-based CPG with sensory feedback and leg labels of the hexapod. Left and right distance sensors are labelled  $S_{LX}$  and  $S_{RX}$ , respectively.

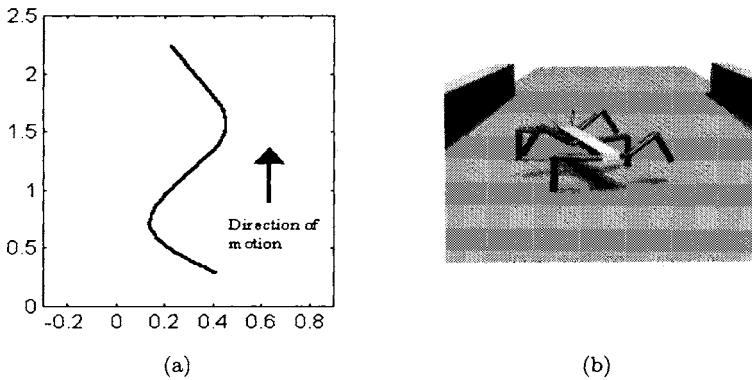


Fig. 3.4 (a) Trajectory of the robot. (b) A frame of the simulation.

oscillations in its state  $x_{1,R2}$  disappear,  $x_{1,R2} < -1$  and the corresponding output  $y_{1,R2}$  is saturated at its low value  $-1$ .  $y_{1,R2}$  being the control signal for the leg motor regulating its height, the leg is kept on the ground until  $Out_{RX}$  takes the low value. As can be observed, the oscillations of cells  $R1$  and  $L2$  are only slightly affected by the dramatic change in the behavior of cell  $R2$ . Analogously, the effect of these two cells on  $R2$  is a small modulation of the stable equilibrium point.

The Braitenberg idea of direct couplings between sensors and motors led to the design of a CPG able to avoid obstacles as well as performing other tasks. Indeed, a purely reactive hexapod CPG able to follow a source can be

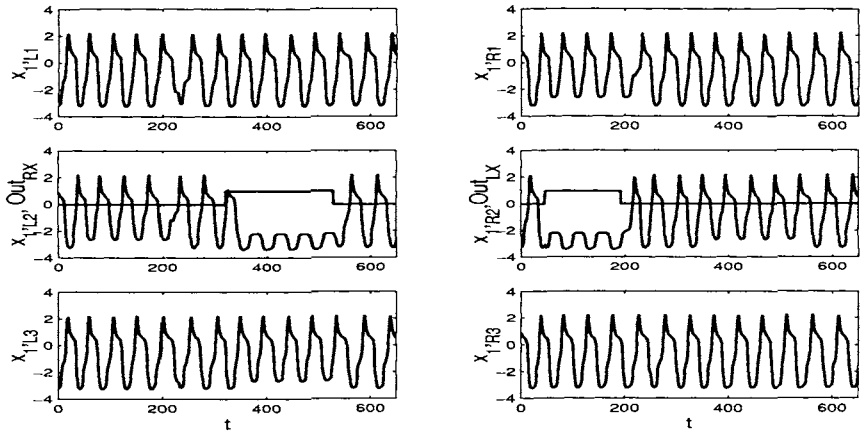


Fig. 3.5 State  $x_1$  for each of the CPG cells and sensor signals  $Out_{RX}$  and  $Out_{LX}$ .

designed in a similar way. This proves that simple principles can implement sensory feedback in the CPG-based hierarchical locomotion control system, providing the bio-inspired robot with further capabilities.

### 3.2 Feedback from ground contact sensors

In this Section another example of sensory feedback is presented [Arena *et. al.* (2003a)]. This approach exploits the synchronization of dynamical systems to allow modulation of the locomotion pattern by means of feedback from sensors. The approach does not depend on the sensors used. However it will be assumed that a sensor for each leg provides the ground contact information, i.e. it indicates if a leg is touching the ground or not.

The inspiration comes from the behavior observed in the CPG of crustaceans. As discussed in Chap. 1, sensory feedback is fundamental for walking. However, the rhythmic activity observed in insects in absence of sensory feedback varies from species to species: in the stick insect for instance the mutual influences between the leg controllers are not sufficient for their normal coordination. In other examples sensory feedback gives rise to many reflexes that are fundamental for good locomotion on rough terrain, but its removal does not imply loss of the locomotion pattern. In any case, sensory feedback is of great importance for the generation of stepping movements: its importance is witnessed by the multitude of different sense organs involved.

In crustaceans control of walking movements takes place in the thoracic ganglia. Generation of the rhythmic pattern can be promoted in isolated chains of thoracic ganglia by the application of pilocarpine [Orlovsky *et. al.* (1999)]. In this case the rhythm elicited is usually slower than normal. Starting from this observation we designed a CPG able to show similar behavior. Our interest of course is not merely to reproduce the CPG behavior of crustaceans, but is strongly motivated by the capabilities of such a system: indeed it adapts its rhythm to the sensory outputs.

From the viewpoint of the implementation of CPGs by using dynamical nonlinear systems, such behavior is possible if the oscillations of the system are able to synchronize themselves to an external signal provided by the sensor outputs. Therefore, the approach pursued for the design of the CNN-based CPG system was to investigate the synchronization of the second-order CNN motor-neurons and exploit them to include feedback from ground contact sensors.

### 3.2.1 *Behavior of the CNN neuron driven by a periodic forcing signal*

The concept of synchronization is an important topic for the study of nonlinear systems. Self-sustained oscillators can be entrained by a weak external periodic forcing signal, provided that the forcing frequency is close to that of self-sustained oscillations. For a suitable range of the parameters of the forcing signal (frequency and amplitude), called the *synchronization region* (or *Arnold tongue*), the oscillations of the entrained nonlinear system have the same frequency as the forcing signal. This behavior is independent of the characteristics of the forcing signal and common to many nonlinear oscillators [Pikovsky *et. al.* (2001)]. If the frequency of the forcing signal is not close to that of the nonlinear oscillators, synchronization may still occur, but in a more complicated form: in this case the general relation, in the synchronization region, between the frequency of the external periodic forcing signal  $\omega_e$  and that of the entrained nonlinear oscillator  $\omega$ , can be expressed as  $n\omega_e = m\omega$ . This synchronization regime is called  $n : m$  synchronization.

The feedback from the ground contact signals acts as an external signal entraining the self-sustained oscillations. Therefore, the synchronization properties of the CNN nonlinear oscillator (3.1) should be investigated to address the issue of feedback from ground contact signals. To this end, Eqs.

(3.1) have to be rewritten as follows:

$$\begin{cases} \dot{x}_1 = -x_1 + (1 + \mu)y_1 - sy_2 + i_1 \\ \dot{x}_2 = -x_2 + sy_1 + (1 + \mu)y_2 + i_2 - \zeta \end{cases} \quad (3.5)$$

where  $y_i = \tanh(2x_i)$  with  $i = \{1, 2\}$  and  $\zeta$  is the external periodic signal. This is a periodic pulse-like signal with period  $T_e$ , amplitude  $A_e$  and duty cycle  $\delta_e$ . The duty cycle was kept constant at a value of  $\delta_e = 0.2$ , while the other parameters of the periodic forcing signal were changed. The parameters of the CNN neuron (3.5) are those in Table 3.1 except for the bias which was set at  $i_2 = -i_1 = 0.33$  (of course for these parameters the limit cycle still exists).

Figure 3.6 shows a typical trend of the synchronized waveform  $x_1$ , obtained for an entraining signal with  $A_e = 0.1$  and  $T_e = 30(t.u.)$ . The entraining signal is applied starting from  $t = 200(t.u.)$ : the oscillation period of the CNN neuron changes from  $T_{x_1} = 40.2(t.u.)$ , when the forcing signal is not applied, to  $T_{x_1} = 30(t.u.)$ , when the forcing signal is applied.

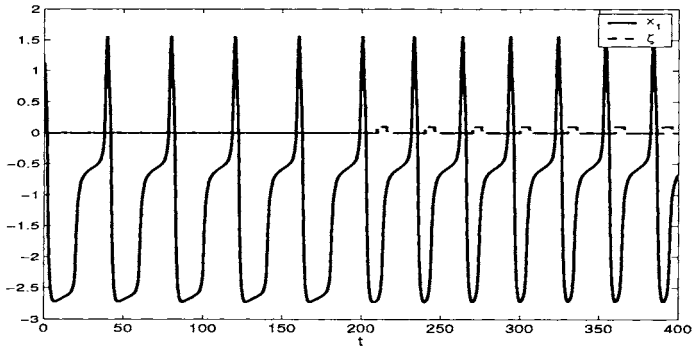


Fig. 3.6 Waveform  $x_1$  of system (3.5) when entrained by a periodic pulse-like signal  $\zeta$  with  $A_e = 0.1$  and  $T_e = 30(t.u.)$ . The forcing signal is applied starting from  $t = 200(t.u.)$ , when the oscillation period is entrained to that of the forcing signal.

The synchronization region was studied numerically. Figure 3.7 shows the Arnold tongues of system (3.5):  $1 : 1$ ,  $2 : 1$ ,  $3 : 1$ ,  $4 : 1$  and  $5 : 1$  synchronization regions were found. The asymmetrical form of the Arnold tongues is due to the fact that the forcing signal acts on system (3.5) by shortening the duration of the slow dynamics, as is evident in Figure 3.6; thus it is impossible to observe synchronization at entraining frequencies higher than the frequency of spontaneous oscillations.



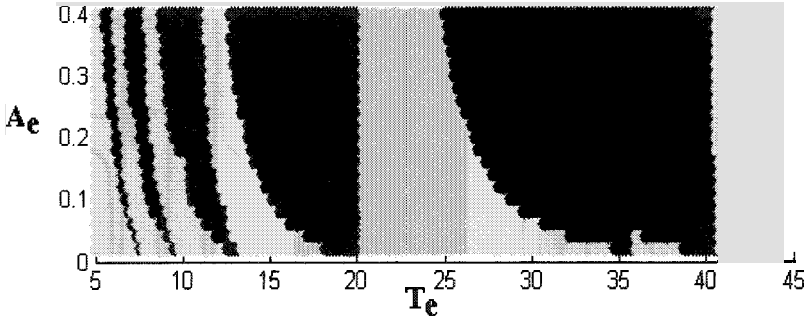


Fig. 3.7 Synchronization regions of system (3.5) driven by a pulse-like periodic forcing signal with amplitude  $A_e$  and period  $T_e$ : the first (from right to left) Arnold tongue refers to 1 : 1 synchronization, the second to 2 : 1 synchronization and so on up to 5 : 1 synchronization.

It should be pointed out that this approach is equivalent to moving the system near the bifurcation illustrated in Sec. 3.1.1. The short pulses of the entraining signal act on the slow part of the dynamics by moving the ghost equilibrium.

After a brief look at the synchronization properties of system (3.5), let us move to the key point of the approach. The ground contact signal acts as the  $\zeta$  signal, i.e. short pulses are applied when the leg is in stance. Let us assume for a while that the robot is walking on a perfectly flat terrain and let us recall that  $y_1$  drives the  $\beta$ -joint, i.e. it regulates the height of the leg. In this case the ground contact signal is high when  $x_1 \leq -1$ . This is a positive feedback during the stance phase. The concept of positive feedback during the stance phase was introduced by Cruse [Cruse *et. al.* (1998a)] and has biological implications. What is the frequency of oscillations when such a signal entrains system (3.5)?

Figure 3.8 shows an example of such a case, when the signal  $\zeta$  is equal to the idealized ground contact, i.e.

$$\zeta = \begin{cases} A_e & x_1 \leq -1 \\ 0 & \text{otherwise} \end{cases} \quad (3.6)$$

As can be noticed, the slow dynamics is almost cancelled by the effects of the positive feedback. The frequency of oscillations is thus the highest frequency possible for the given value of  $A_e$ . Figure 3.9 confirms this:

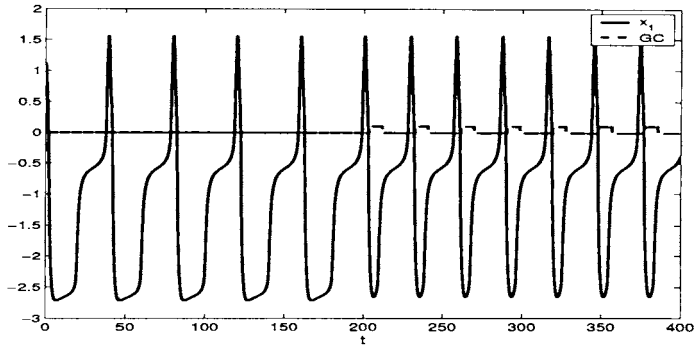


Fig. 3.8 Waveform  $x_1$  of system (3.5) when entrained by signal (3.6) with  $A_e = 0.1$ . This signal is applied starting from  $t = 200(t.u.)$ , when the oscillation period changes from  $T_{x_1} = 40.2(t.u.)$  to  $T_{x_1} = 28.98(t.u.)$ .

it shows the period of oscillations versus the amplitude of the idealized ground contact signal. To match this plot with the 1 : 1 Arnold tongue shown in Fig. 3.7, the period of oscillations is on the  $x$ -axis, even if the period is a function of the applied signal amplitude. Figure 3.9 allows us to draw the conclusion that *the oscillations occur at the boundary of the 1 : 1 synchronization region*. Therefore, this approach leads to a CPG able to show a locomotion pattern modulated by the sensory feedback, in which the stepping frequency is slower in absence of feedback.

Of course, when the robot is walking on rough terrain, the behavior is unpredictable: as shown below this feedback provides the CNN-based CPG with adaptive capabilities.

It is worth remarking that similar results are also discussed in [Gallagher *et. al.* (1996)] even if the underlying approach is quite different. The CPG controller in [Gallagher *et. al.* (1996)] is automatically built up by an evolutionary technique based on a description of the desired behavior. This controller was constructed under the hypothesis that feedback is not always available. The hexapod agent driven by this controller shows a higher stepping frequency (and in general better performance) when sensors are available.

### 3.2.2 CPG with ground contact feedback: results

In order to validate the new strategy proposed, the control scheme has been tested by using the dedicated simulation environment HexaDyn (see also

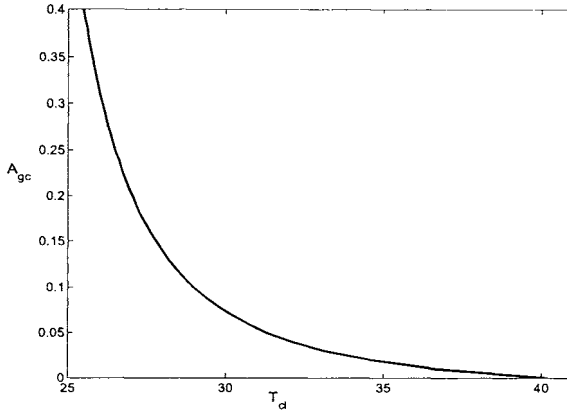


Fig. 3.9 Period of oscillations of system (3.5) entrained by signal (3.6). To match Fig. 3.7 the period of oscillations is on the  $x$ -axis, while the amplitude of  $\zeta$  is on the  $y$ -axis.

Appendix A). The dynamic hexapod robot model is called the HexaDyn robot. HexaDyn includes ground contact sensors: they take into account the vertical projection of the force at the tip of each leg and, if this results in a force towards the ground, the leg touches the ground, otherwise it is considered to be lifted in the air. The signal coming from this simple sensor is not very accurate and sometimes (for example, when the leg is sliding) it does not indicate the correct status of the leg. However, this is sufficient for our purpose and, at the same time, demonstrates the robustness of the approach to sensor faults.

The scheme of the CPG (fast gait with  $\epsilon_f = -0.6$ ) with ground contact feedback is the same as in Fig. 3.3(a) in which each motor-neuron is equipped with a ground contact sensor.

Figure 3.10 refers to the robot walking on flat terrain. The ground contact feedback is activated starting from  $s \simeq 400$  (samples). The stepping frequency is higher in the presence of sensory feedback.

Moreover, an emerging property was noticed: when the feedback is activated, the motor-neuron signals rapidly synchronize. Figure 3.11 illustrates this behavior: when the control is switched on, the waveforms of the motor-neurons belonging to the same tripod are not well-synchronized; when the feedback is switched on, they rapidly synchronize.

To further validate the approach, the behavior of HexaDyn when walk-

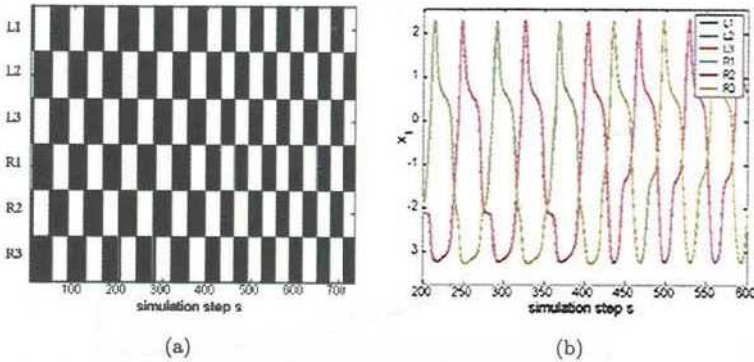


Fig. 3.10 (a) Step pattern of the HexaDyn robot while walking in tripod gait with ground contact feedback and (b) waveforms of the control signals  $x_1$  for each motor-neuron. At  $s \approx 400$ (samples) the ground contact feedback is switched on.

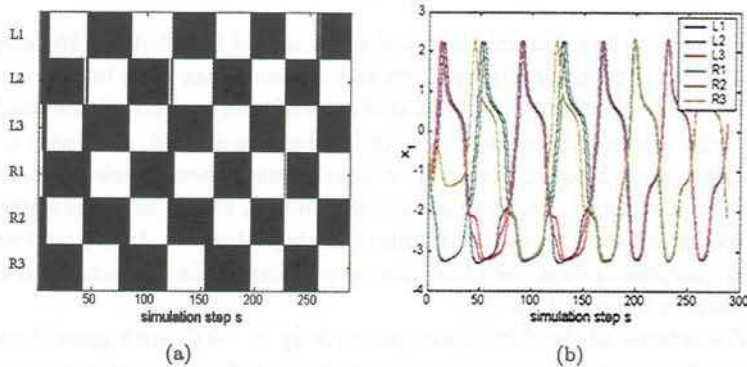


Fig. 3.11 An emerging property of the ground contact feedback: when the control is switched on, the waveforms of the motor-neurons belonging to the same tripod are not well-synchronized; when the feedback is switched on (at  $s \approx 150$ (samples)) these rapidly synchronize. (a) Step pattern of the HexaDyn robot and (b) waveforms of the control signals  $x_1$  for each motor-neuron.

ing on rough terrain was considered. In most cases the locomotion pattern is almost unchanged, as shown in Fig. 3.12(a), but sometimes the coordination between leg controllers slightly changes to match the terrain characteristics, as can be seen in Fig. 3.12(b).

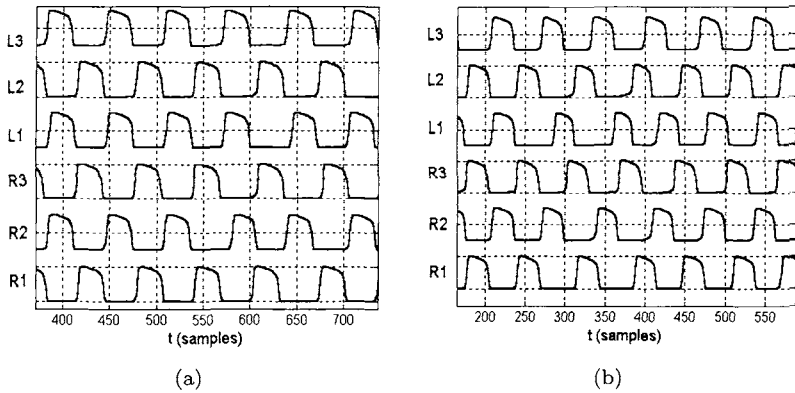


Fig. 3.12 Waveforms of the control signals  $y_1$  for each motor-neuron of HexaDyn walking on rough terrain (after appropriate scaling these signals are used to drive the robot's  $\beta$ -joints, so vertical units are arbitrary). Two different cases are presented: (a) the locomotion pattern is almost unchanged; (b) the coordination between leg controllers changes to match the characteristics of the terrain.

### 3.3 Reflex implementation

When insects walk on irregular terrain, they move in a very effective way by using a variety of local leg reflexes [Pearson and Franklin (1984)]. Insects are able to deal with irregular surfaces, small elevated steps, ditches, poor supports and so on. These reflexes can provide robotic solutions to build up autonomous structures able to negotiate uneven terrain [Espenschied *et. al.* (1996)]. Indeed, reaching areas that cannot be accessed by wheeled vehicles is a clear advantage of walking robots: this can be achieved by implementing local reflexes.

A CPG including two fundamental local reflexes is here introduced.

#### 3.3.1 The elevator reflex

When during the return stroke a leg encounters an obstacle blocking its trajectory, an elevator reflex is triggered [Pearson and Franklin (1984)]: the leg rapidly retracts to disengage from the obstacle, then lifts the foot and swings over the obstacle.

This reflex can be implemented in motor-neuron (2.5) by allowing the bias  $i_2$  in Eq. (2.5) to be expressed by the following relation:

$$i_2 = 0.3 + \xi \quad (3.7)$$

where  $\xi$  is a short pulse (of amplitude  $A_\xi = -2.2$ ) triggered by the obstacle. When such a signal is applied the behavior of system (2.5) changes as shown in Fig. 3.13. The trajectory is slightly modified. If we assume that the signal driving the  $\beta$ -joint is  $x_1$  saturated only at the low level ( $x_1 = -1$ ), the leg is elevated when the reflex is triggered. Moreover, to allow larger excursion of each leg, it has been assumed that the high saturation point of  $y_1$  is  $y_1 = 2$  instead of  $y_1 = 1$ .

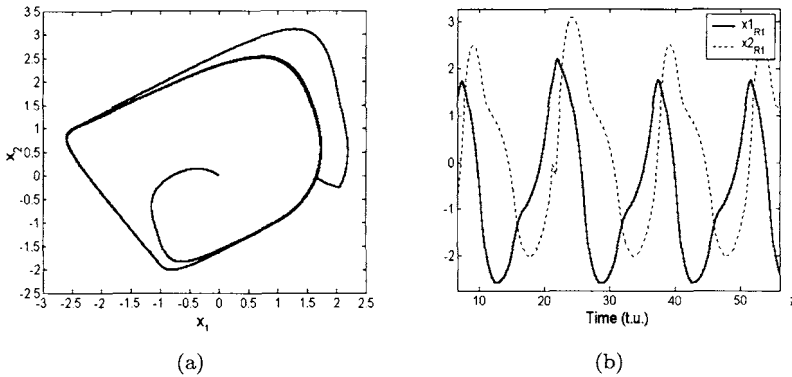


Fig. 3.13 Behavior of a CNN motor-neuron when the elevator reflex is triggered: (a) phase plane  $x_1 - x_2$ ; (b) waveforms of  $x_1$  and  $x_2$ .

The behavior of the CPG is illustrated in Fig. 3.14. As can be noticed the leg R1 lifts higher when the reflex is triggered. A very efficient way to implement a sensor for the elevator reflex is reported in [Klaassen *et. al.* (2002)]. By measuring the current of the joint motor it is possible to monitor whether the leg encounters an obstacle during the swing phase.

### 3.3.2 The searching reflex

When support is missing, as occurs for example if there is a hole in the terrain surface, insects adopt the strategy of rapidly moving the leg to search for additional support [Pearson and Franklin (1984)]. This behavior is called the *searching reflex*. Insects randomly search for possible foothold positions in increasing areas.

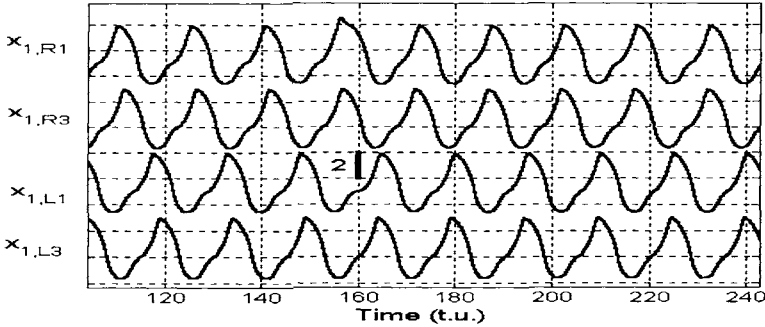


Fig. 3.14 Behavior of tripod gait ( $\epsilon_f = -0.1$ ) when the elevator reflex is triggered. The waveforms of  $x_{1,R1}$ ,  $x_{1,R3}$ ,  $x_{1,L1}$  and  $x_{1,L3}$  are shown.  $x_{1,R2}$  and  $x_{1,L2}$ , not shown, follow the trend of  $x_{1,L3}$  and  $x_{1,R3}$ , respectively.

The searching reflex is introduced in the CNN-based CPG scheme by adopting a chaos-driven search for a foothold: when the expected location of the foothold is not met, the trajectory changes from periodic to chaotic as an effect of the activation of the reflex. This enables the leg tip to randomly explore the area below to search for a foothold position.

To this end the basic motor-neuron has to include the possibility of varying its behavior from periodic to chaotic as some parameters are changed. Zou and Nosssek showed that chaotic behavior can be obtained in a nonautonomous CNN consisting of two cells and driven by a sinusoidal input [Zou and Nosssek (1991)]. The CNN has an opposite-sign template like motor-neuron (2.5), which can be slightly modified to build a CPG with the searching reflex according to [Zou and Nosssek (1991)]. System (2.5) is therefore rewritten to include a sinusoidal input as follows:

$$\begin{cases} \dot{x}_1 = -x_1 + (1 + \mu)y_1 - sy_2 + i_1 + A \sin \omega t \\ \dot{x}_2 = -x_2 + sy_1 + (1 + \mu)y_2 + i_2 \end{cases} \quad (3.8)$$

where

$$y_i = 0.5(|x_i + 1| - |x_i - 1|) \quad i = \{1, 2\}$$

Here two different sets of parameters are considered. For the following set of parameters:

$$\mu = 1; s = 1.2; i_1 = -0.14; i_2 = 0.14; A = 0; \omega = \frac{2\pi}{T}; T = 4 \quad (3.9)$$

a stable limit cycle appears, while for the following parameters

$$\mu = 1; s = 1.2; i_1 = 0; i_2 = 0; A = 4.04; \omega = \frac{2\pi}{T}; T = 4 \quad (3.10)$$

a chaotic attractor is evident. The behavior of the cell is illustrated in Fig. 3.15.

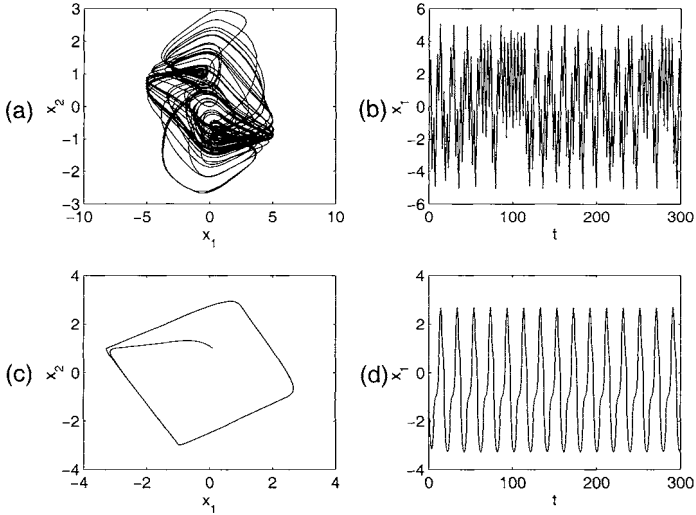


Fig. 3.15 Behavior of system (3.8) for different sets of parameters: (a)-(b) parameters as in (3.10); (c)-(d) parameters as in (3.9).

Motor-neuron (3.8) can be used to build a CPG as shown in Fig. 3.16, where slow gait (with  $\epsilon_s = 0.2$ ) was considered. The underlying idea is that when contact is not found, the sinusoidal input is switched on, allowing the cell to behave chaotically. In the meanwhile the gait is stopped: by adopting the strategy discussed in Sec. 3.1 it is possible to stop each leg either in the stance contact position or at the lift-off point.

### 3.4 Speed control

The control of walking speed in insects is complex. In order to walk faster, insects increase the velocity of joint movements, in particular shortening the stance phase. This may result in adopting a totally different gait. Moreover, in some cases [Watson and Ritzmann (1998)] new motor-neurons may be



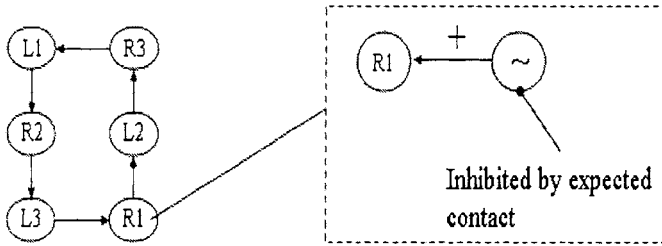


Fig. 3.16 Scheme of the CPG including the searching reflex.

recruited at high speeds. The transition between slower and faster gaits is continuous and very complex.

Taking inspiration from the real world, the approach depicted here is much more simplified. As mentioned above, different connections between the motor-neurons account for the possibility of changing the locomotion pattern. To vary the walking speed without changing the locomotion pattern another mechanism is used. This is based on changing the frequency of the oscillations of the CNN neurons constituting the CPG. It can be accomplished in a very simple way by adopting a switched-capacitor technique to implement the CPG. In this sense the control strategy proposed here is strictly related to the VLSI technique adopted. The switched-capacitor (SC) technique [Gregorian and Temes (1986)] is a key factor to obtain a simple control strategy: even if there is clearly nothing analogous to such an approach in the biological world, the technique can be said to be bio-inspired given its features of immediateness and simplicity, and since the core of the control is analog.

### 3.4.1 Speed of the gait

In order to illustrate the idea underlying speed control, let us rewrite the equation of the second-order CNN neuron (2.5) as follows:

$$\begin{cases} \frac{dx_1}{dt} = \frac{1}{\tau}(-x_1 + (1 + \mu)y_1 - sy_2 + i_1) \\ \frac{dx_2}{dt} = \frac{1}{\tau}(-x_2 + sy_1 + (1 + \mu)y_2 + i_2) \end{cases} \quad (3.11)$$

with

$$y_i = \frac{1}{2}(|x_i + 1| - |x_i - 1|) \quad i = \{1, 2\}$$

This introduces a time-scaling factor in Eq. (2.5) that does not change anything but the frequency of the oscillations of the CPG cell. The parameter  $\tau$  in Eq. (3.11) constitutes the time-scaling factor; when the cell (3.2) is implemented in an electronic circuit, this is usually given by  $\tau = RC$ .

In switched-capacitor based integrated circuits resistors are implemented by means of switched capacitors. An equivalent resistor circuit can be implemented by a capacitor  $C_R$ , switched by a clock with a frequency  $f_c = \frac{1}{T}$ . It approximates a resistor of the following value:

$$R = \frac{T}{C_R} \quad (3.12)$$

The time-scaling factor is therefore given by:

$$\tau = T \frac{C}{C_R} \quad (3.13)$$

Equation (3.13) is fundamental to achieve two features of locomotion control.

- The control approach is suitable for different kinds of actuators requiring different actuating frequencies. For instance, it can be used to drive a walking robot actuated by servo-motors and with a stepping frequency in the order of magnitude of Hz as well as a micro-robot using piezoelectric actuators operating in a 60-90 Hz frequency range.
- The speed of a given locomotion control can be changed by slightly varying the frequency of the clock.

In other words, large variations in the clock frequency lead to the possibility of controlling different kinds of actuators, while slight changes address the speed control issue.

Moreover, Eq. (3.13) allows large time constants to be integrated. The typical stepping frequency of insects (and of walking robots) is in the order of magnitude of Hz. This could lead to large capacitors that may require either large silicon areas or external capacitors. Neither solution is appealing. Instead, by using switched-capacitor based integrated circuits, it is possible to obtain low stepping frequencies. For instance, by selecting  $\frac{C}{C_R} = 10$  and a clock frequency of  $f_c = 100\text{Hz}$  the time scale factor is equal to  $\tau = 0.1\text{s}$ , the oscillations of the CNN neuron (3.11) have a frequency equal to  $f_o \simeq \frac{1}{8\tau}$ , and a stepping frequency in the order of magnitude of Hz is achieved.

### 3.4.2 *Locomotion pattern*

The CNN-based CPGs for the most common gaits for hexapod walkers are described in Sec. 2.6. The choice of the right locomotion pattern can be addressed by a higher level control. This can be based on Turing patterns (as illustrated in Chap. 6) and can use any source of information to establish which speed has to be adopted. Hence our model provides two different strategies: switching between a given set of possible locomotion patterns and continuous variation of the speed of the pattern.

## 3.5 The CNN-based CPG VLSI chip

### 3.5.1 *The hybrid approach*

The strategies discussed previously allow control of the direction of the robot, its walking speed and the choice of the gait and lead to the so called *hybrid control* illustrated in Fig. 3.17. The core of this control scheme is analog and consists of the CNN devoted to locomotion pattern generation. The behavior of this CPG is modulated by signals coming from sensory feedback or high level control. These signals can be processed by a digital controller. Since the gait generation is entrusted to the CPG analog core, this strategy allows the use of a very simple and cheap digital micro-controller and, at the same time, conjugates the flexibility of digital control with the power of analog parallel processing. In other words, the strategy exploits the peculiar capabilities of the digital and analog world, delegating feedforward gait generation (the most difficult task for a micro-controller, since the number of actuators of the locomotion system can be high) to the analog core and the feedback control law to the digital control.

The CNN-based CPG VLSI chip implements the analog core of the hybrid approach to locomotion control and as inputs receives all the signals to modulate the behavior of the CNN-based CPG according to the above considerations. The analog CPG is designed using a switched-capacitor technique. The clock frequency of the switches can be regulated to obtain suitable control of the stepping frequency. Moreover, different connections among the six motor-neurons of the CPG can be selected in order to obtain different locomotion patterns and the biases of the middle cells are regulated by external signals in order to change the direction of the robot.

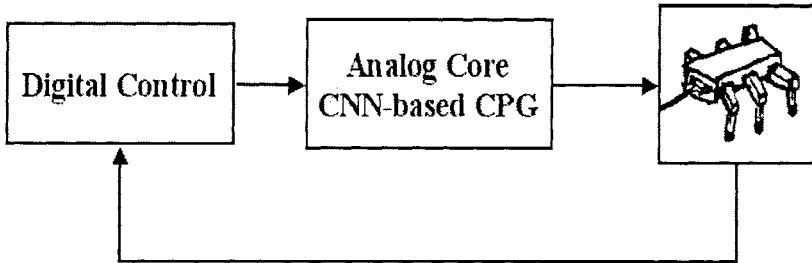


Fig. 3.17 Scheme of hybrid control: the analog core (the CNN-based CPG) is controlled by a digital controller modulating its behavior according to the sensory feedback.

### 3.5.2 The VLSI Circuit Design

The main blocks of the switched-capacitor chip are:

- a phase generator;
- six second-order circuits implementing Eq. (3.11) (the motor-neurons);
- three different sets of switchable connections for fast gait, medium gait and slow gait;
- a gait selector block.

Since the switches are composed of an NMOS transistor and a PMOS device in a parallel arrangement (i.e. in the well-known CMOS transmission gate configuration [Rabaey (1996)]), the switched-capacitor circuit requires the availability of two nonoverlapping clocks (each with two complementary phases). The purpose of the phase generator block is to generate the two-phase clock starting from the external sinusoidal clock the chip is provided with.

The schematics of a single motor-neuron is shown in Fig. 3.18, the design follows the operational amplifier implementation of the CNN cell (see Appendix B and [Manganaro *et. al.* (1999)]) and the switched-capacitor design technique [Gregorian and Temes (1986); Johns and Martin (1997)]. The values of the capacitors in the circuit were chosen to match the parameters given in Table 3.1 minimizing the area occupation. These values are given in Table 3.2.

The operational amplifiers involved in the motor-neuron schematic are canonical two-stage CMOS operational transconductance amplifiers

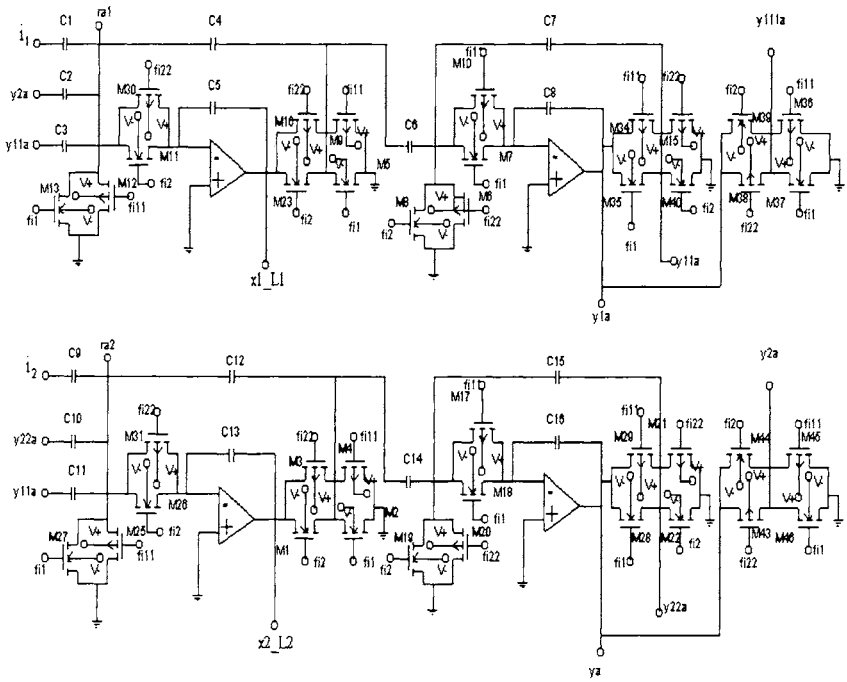


Fig. 3.18 Schematic of the switched-capacitor circuit implementing a CNN motor-neuron.

Table 3.2 Capacitor values of the circuit in Fig. 3.18.

$C_1, C_2, C_7, C_8, C_9, C_{11}, C_{15}, C_{16}$	$0.1pF$
$C_4, C_6, C_{12}, C_{14}$	$0.3pF$
$C_5, C_{13}$	$1pF$
$C_3, C_{10}$	$0.15pF$

[Rabaey (1996); Johns and Martin (1997)] with dominant-pole compensation, static gain  $A \approx 55dB$ , phase margin  $M_\phi = 60^\circ$  and unity-gain frequency  $f_T = 15MHz$ .

In order to implement different gaits, three different sets of connections have been implemented, adopting the topologies discussed in Sec. 2.6. These connections are switchable and are activated according to the value of an analog input, the Gait Select ( $GS$ ) signal. This signal constitutes the input of a comparator block whose outputs select the switchable connections, thus establishing the locomotion gait.

The characteristics of the chip are shown in Table 3.3. The inputs of the chip are the control signals  $GS$ ,  $i_{2,L2}$  and  $i_{2,R2}$  (bias of the middle motor-neurons to allow direction control), while the outputs are the two state variables of each motor-neuron.

Table 3.3 Characteristics of the CNN-based CPG chip.

Power Supply	$\pm 2.5V$
Power Consumption	$50mW$
Pins	19
Area	$1400\mu m \times 1200\mu m$
Clock (sinusoidal)	amplitude $1.5V$ , frequency $f_c$
3 Inputs	$i_{2,L2}$ , $i_{2,R2}$ , $GS$
12 Outputs	$x_{1,L1}$ , $x_{2,L1}$ , $\dots$ , $x_{1,R1}$ , $x_{2,R3}$

The technology adopted is CMOS AMS  $0.8\mu m$ . The layout of the chip is shown in Fig. 3.19. A photo of the first prototype is shown in Fig. 3.20.

### 3.5.3 Experimental results

The experimental results illustrate the key points of the control approach discussed above. These results were obtained on the first chip prototype. All the data was acquired using a data acquisition board (National Instruments AT-MIO 1620E) and successively plotted using MATLAB tools.

- *Range of oscillation frequency*

First of all, the behavior of the single cell at different clock frequencies is illustrated. Here and in what follows, if not explicitly stated, it is assumed that the connections among the cells are set to fast gait. Figure 3.21 shows the trends of the two state variables

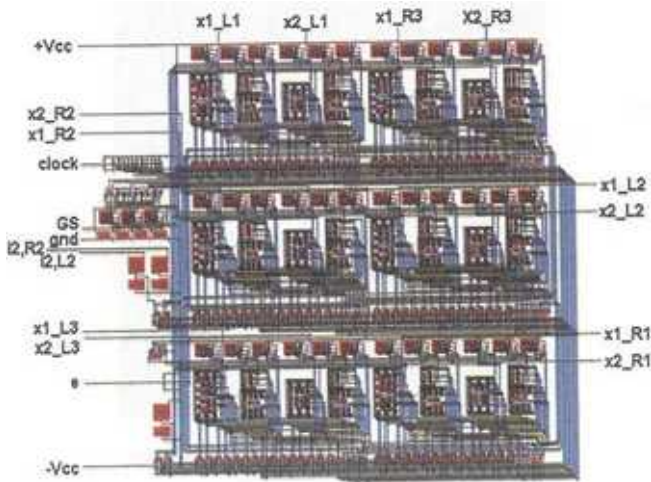


Fig. 3.19 Layout of the CNN-based CPG chip.

$x_1$  and  $x_2$  of the motor-neuron L1 at two different clock frequencies (Fig. 3.21(a) refers to  $f_c = 10kHz$ , Fig. 3.21(b) to  $f_c = 100Hz$ ) and the phase plane  $x_1 - x_2$  for the case of  $f_c = 100Hz$  (the case of  $f_c = 10kHz$  is practically indistinguishable); the corresponding oscillation frequencies are  $f_o = 132Hz$  and  $f_o = 1.32Hz$ , respectively. These two cases demonstrate the suitability of the approach to control different actuators (for instance servo-motors and piezo-electrics). Moreover the low stepping frequency of the latter case is experimental confirmation that this technique allows large time constants with relatively small capacitances.

The frequency range was further investigated by measuring the period of oscillations versus the clock frequency in the interval  $100mHz - 500kHz$ . Figure 3.22 illustrates the results of the analysis, showing the suitability of the approach in the considered range. These results were obtained by considering fast gait, storing all the data and then computing the period of oscillations. The specifications of the data acquisition board do not allow us to consider clock frequencies greater than  $f_c = 500kHz$ . However, an analysis on the oscilloscope reveals that the operating range is even wider; at  $f_c = 3MHz$  the chip outputs are saturated.

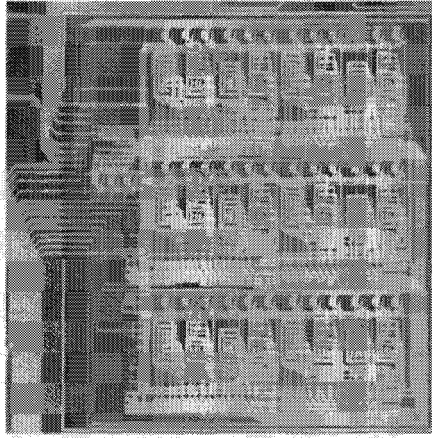


Fig. 3.20 Photo of the CNN-based CPG chip.

- *Speed control*

Speed control can be addressed by providing slight changes in the clock frequency. As an example we considered small variations of the clock frequency around  $f_c = 100\text{Hz}$ , since this value represents a suitable value for control of the hexapod robot. The results are shown in Fig. 3.23.

Table 3.4 shows a comparison between the simulated and measured periods. As can be noticed, the agreement between simulation and experimental results is good.

Table 3.4 Comparison between simulated and measured oscillation period at different clock frequencies.

$f_c$ [Hz]	$T_{sim}$ [ms]	$T_m$ [ms]	$\frac{T_m - T_{sim}}{T_{sim}}$
90	780	845	8.3%
100	700	760	8.6%
110	630	690	9.5%

- *Locomotion gait*

The CNN-based CPG is able to generate different patterns of locomotion. However, in the first chip prototype only fast gait was implemented. Figure 3.24 shows the waveforms of  $x_1$  for each cell at two different clock frequencies. As can be noticed the signals of



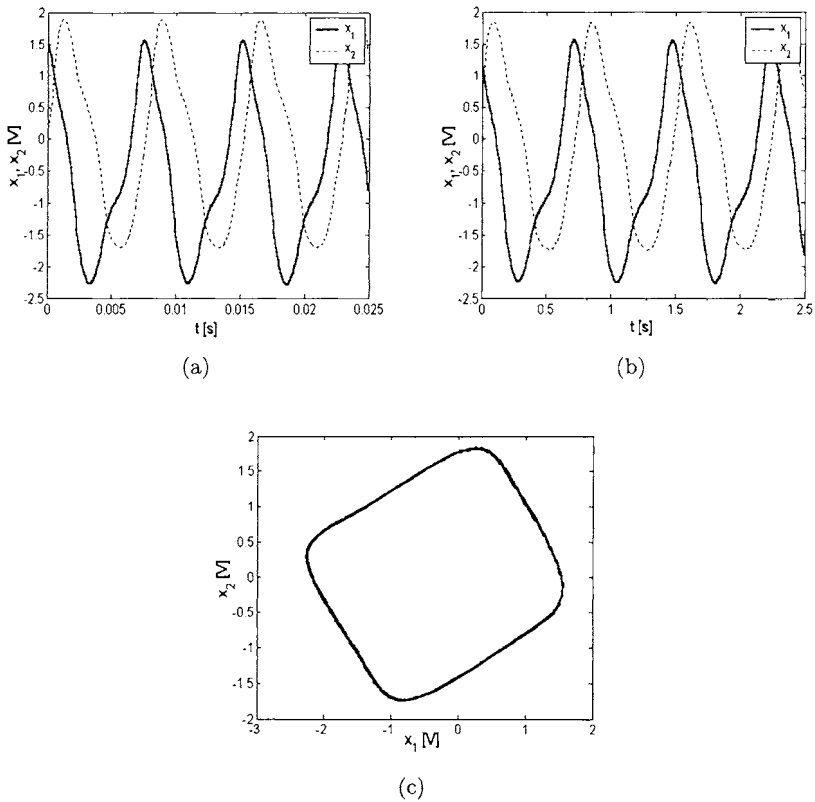


Fig. 3.21 Waveforms of the two state variables  $x_1$  and  $x_2$  of the motor-neuron L1 at two different clock frequencies  $f_c = 10kHz$  (a) and  $f_c = 100Hz$  (b). (c) Phase plane  $x_1 - x_2$  with  $f_c = 100Hz$ .

the two tripods are perfectly synchronized. The good agreement between simulations and experimental results for fast gait guarantees the possibility of obtaining the other gaits.

- *Direction control*

The two chip inputs  $i_{2,L2}$  and  $i_{2,R2}$  provide a suitable way to control the direction of the robot. The results given here illustrate the possibility of changing the behavior of the CPG by acting on these inputs. One of these two inputs was kept constant at  $2.5V$ , while a periodic square wave (between  $2.5V$  and  $5V$  with a duty cycle of  $\delta = 30\%$  and variable frequency) was applied to the pin corresponding to the other input. The suitability of the approach was verified

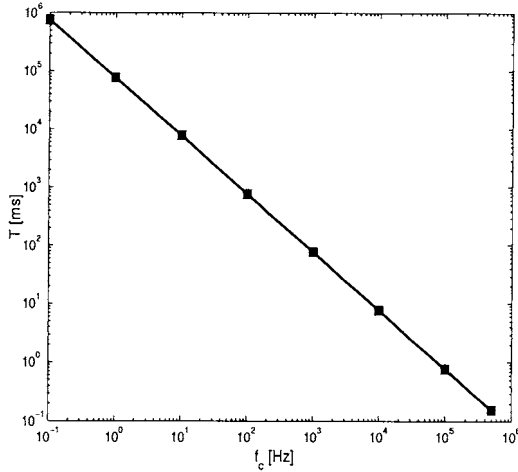


Fig. 3.22 Period of oscillations  $T$  of the fast gait signals versus the clock frequency  $f_c$  (log-log plot).

at different clock frequencies. In particular, Figs. 3.25 and 3.26 refer to  $f_c = 100Hz$ . In this case the frequency of the square wave signal applied to the input  $i_{2,R2}$  is  $f_i = 80mHz$ . Figure 3.25 shows that, when the square wave signal is high, the movements of the leg  $R2$  stop since the state  $x_{1,R2}$  of the motor-neuron  $R2$  is less than  $-1$ , the output is  $y_{1,R2} = -1$ , and thus the leg is on the ground. Figure 3.26 shows the waveforms superimposed on the same plot. This allows us to appreciate the short duration of the transitory phase: when the input signal emulating the sensor output goes down, the locomotion gait is rapidly recovered.

In Chap. 5 further experimental results (on a hexapod robot controlled by the chip) are shown.

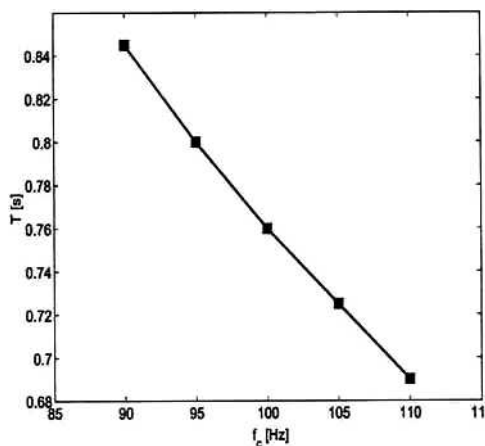
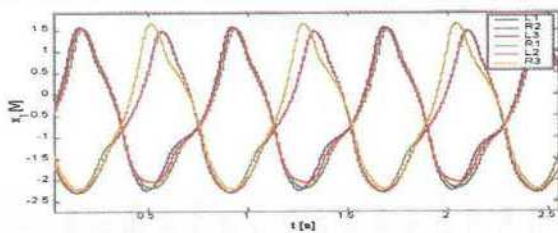
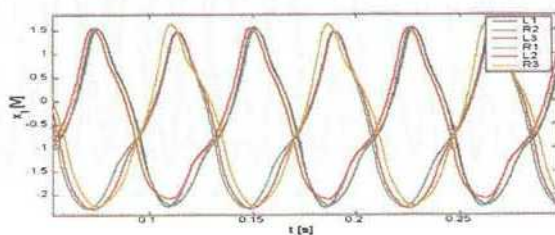


Fig. 3.23 Period of oscillations  $T$  of the fast gait signals versus the clock frequency  $f_c$  (slight changes).



(a)



(b)

Fig. 3.24 The fast gait at different clock frequencies: trend of  $x_1$  for each CNN motor-neuron. (a)  $f_c = 100\text{Hz}$ ; (b)  $f_c = 1\text{kHz}$ .

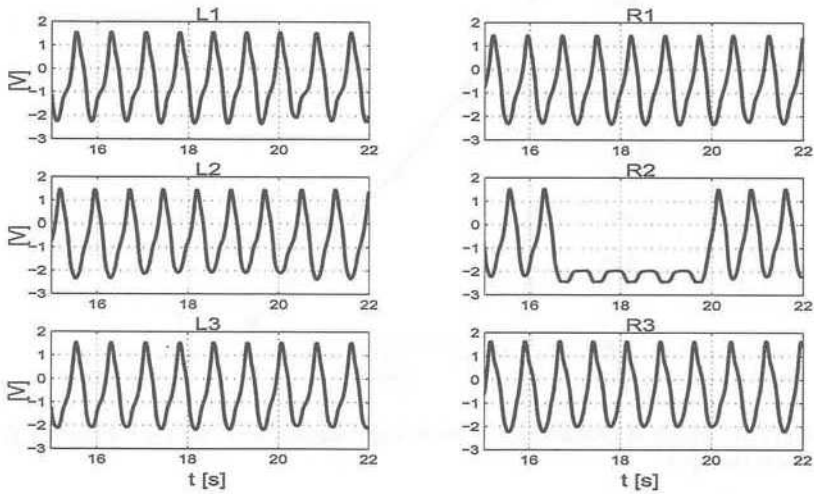


Fig. 3.25 Direction control at  $f_c = 100\text{Hz}$ : waveforms of  $x_1$  for each cell. When the square wave signal is high, the oscillations of leg  $R2$  stop.

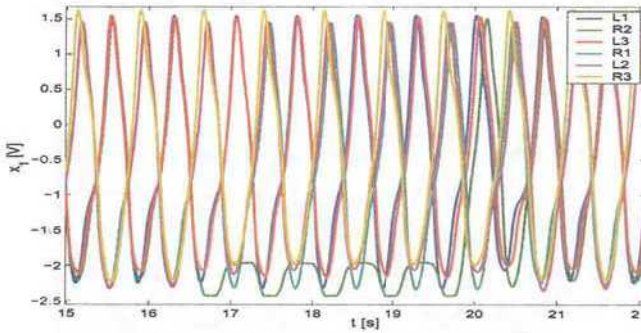


Fig. 3.26 Direction control: recovery of the locomotion pattern (fast gait) after the input  $i_{2,R2}$  goes down.

## Chapter 4

# Decentralized locomotion control

In this Section decentralized locomotion control inspired by the stick insect model is dealt with. In this model the locomotion pattern is the result of a set of local influences among leg controllers whose behavior is regulated by reflexes triggered by sensors. In this sense this scheme is totally different from the CPG, in which the generation of the driving signals does not rely on sensory feedback.

Two different implementations of decentralized locomotion control are presented here. The first is based on CNNs, the second on integrate-and-fire neurons. The two approaches share the idea of using nonlinear dynamical systems at the low level of locomotion control. This implementation strategy is also common to the CNN-based CPG presented in the previous Chapters. The capability to solve the complex problems of legged robot locomotion by means of distributed control in which the computational effort is divided between the self-organizing nonlinear units is a peculiar feature of this implementation strategy. Moreover, the intrinsic self-organization of the networks makes the system robust to parameter changes and faults.

### 4.1 CNN-based decentralized control model

The stick insect (Fig. 4.1) is a fascinating insect that slowly walks by adopting a large variety of reflexes. Its behavior is usually given as an example of an approach for locomotion control totally different from the CPG. The locomotion control is based only on reflexes.

In fact, experiments carried out on the stick insect by Cruse [Cruse *et. al.* (1998a)] led to the emergence of a locomotion pattern from local influences among the networks of neurons devoted to the control of each leg. This is in contrast with a number of findings in other animals, in which



Fig. 4.1 A photo of a stick insect.

the locomotion pattern is assumed to be the result of hierarchical control and its generation takes place at the neural level of the CPG. The models used by Cruse are essentially implemented via artificial neural networks, based on a static neuron model.

This Chapter introduces a new general methodology for the control of locomotion patterns in hexapod robots, applied here to reproduce the motion of stick insect locomotion. This approach is inspired by the decentralized approach and introduces a network of locally coupled nonlinear dynamical systems able to organize themselves so as to show the same complex behavior as the stick insect locomotion generator. Particular attention is also paid to the possibility of directly realizing the control scheme in a hardware framework. This is to fill the gap left by most of the methodologies introduced up to now, which stop at a software level, neglecting implementation issues. This approach, on the contrary, focuses directly on the hardware realization of the locomotion control.

The basic dynamical unit consists of the second-order nonlinear system (2.5). This choice was motivated by an analogy between this cell and an assembly of biological neurons forming an attractor network. The characteristics of the spatial-temporal dynamics arising in arrays of cells of this type are also qualitatively the same as those arising in pools of biological neurons. It will be shown that the dynamic capabilities of the cell can be efficiently exploited to control the leg kinematics of an insect-like hexapod robot. In particular, the limit cycle behavior of the cell, as well as its slow-fast dynamics, will be preserved in order to periodically drive the

leg; at the same time the interactions among the other cells will allow the emergence of the locomotion pattern. This mechanism will be formalized in a very simple way in order to guarantee the feasibility of the hardware realization. The proposed approach implements a dynamical leg controller structure, at the same time preserving the capabilities of both the stance controller network and the swing controller network, which strictly reflect the biological behavior [Cruse *et. al.* (1998b)].

The influences between the cells are the same as used in [Cruse *et. al.* (1993)]. However, the parameter values were changed, since the core of the leg controller is in this case a dynamical system, acting in a very different way with respect to the previous approach. The influence parameters were found by using both a heuristic procedure and genetic algorithms.

The focus of this Section is to formalize a decentralized locomotion control system using the CNN paradigm, in order to generate locomotion rhythms based on sensory feedback to provide suitable controllers for bio-inspired robots [Arena *et. al.* (2003b)]. The approach shows its suitability to drive six legs in an organized way, in accordance with the model of the stick insect. The simulation results obtained reveal that the network generated is able to reproduce both *tripod* and *tetrapod gait* and shows the capabilities of pattern recovery in the event of leg jamming due to external disturbances.

#### 4.1.1 *The decentralized control paradigm*

The decentralized control paradigm is based on a distributed network of neural controllers, with no hierarchy, that are able to generate proper motions based on reflexes and local influences (unlike the CPG, it strictly needs sensory feedback for pattern generation, while the locomotion pattern generation in the CPG takes place even in absence of sensory feedback). This type of control is realized in [Cruse *et. al.* (1998a)], through the so-called Walknet model. Walknet is based on a controller network for each leg and on a set of local influences acting on two kinematic parameters: the posterior extreme position (PEP) and the anterior extreme position (AEP) of the leg contact point. The AEP represents the point when the leg touches the ground, i.e. the transition from the return stroke to the power stroke, while the PEP represents the transition from the power stroke to the return stroke, i.e. the point when the leg is lifted in the air.

According to the Walknet model, each leg controller is divided into three main blocks: the *selector net*, the *swing net* and the *stance net*. The swing

net is devoted to controlling the trajectory of the leg while in the swing phase, whereas the stance net controls the leg trajectory when the leg is in the stance phase. On the basis of sensorial inputs and proprioceptive information, the selector net establishes which of the two other nets should actively control the leg. The inputs to this selector net are the ground contact (GC) signal, indicating when a leg is on the ground, and the PEP signal, which indicates if the PEP has been reached or not.

A set of local influences acting on the kinematic parameters (AEP, PEP) of each leg controller has been shown to be suitable to explain the emergence of a well-defined gait in stick insect locomotion [Cruse *et. al.* (1998a)]. Experiments reveal that the stick insect adopts two gaits to deal with different environmental conditions: tripod gait and tetrapod gait. The former allows higher speed and is characterized by two alternating leg tripods sustaining the body. In tetrapod gait, in contrast, there are at least four legs in the stance phase at the same time.

The characteristics of these two patterns can be defined through the concepts of *cycle time*, *duty factor* and *phase lags* introduced in Sec. 2.5. For sake of commodity and completeness, let us now indicate the leg phases with capital letters  $\Phi_i$  and let us recall that using these parameters, it is possible to characterize tripod gait as a periodic gait with the same duty factor for all the legs ( $df_i = 0.5$ ) and the following phase lags:

$$\Phi_{L2} = 0.5; \Phi_{L3} = 0; \Phi_{R1} = 0.5; \Phi_{R2} = 0; \Phi_{R3} = 0.5 \quad (4.1)$$

where the legs are numbered from front to rear and labelled as left (L) or right (R).

In the stick insect another gait is experimentally observed, the tetrapod gait. In this investigation, we concentrate on an intermediate form of tetrapod gait (slightly different from the medium gait discussed in Sec. 2.6) characterized by  $df_i = 2/3$  and the following phase lags:

$$\Phi_{L2} = 0.75; \Phi_{L3} = 0.5; \Phi_{R1} = 0.5; \Phi_{R2} = 0.25; \Phi_{R3} = 0. \quad (4.2)$$

Even in this case it should be noticed that although often regarded as two different gaits, tripod gait is a special form of tetrapod gait. What is called tripod gait is the fast version of tetrapod gait [Graham (1985)]. This means that there is a continuous transition between slow tetrapod and fast tripod gait and all intermediate forms are stable versions of tetrapod gait.



In more detail, six local influences are identified [Cruse (1990)]: a leg in return stroke inhibits the start of the return stroke in the anterior leg (first influence, rostrally directed, i.e. from rear to front); when the leg begins its power stroke, it excites the start of the return stroke in the anterior leg (second influence, rostrally directed); while in stance a leg excites the start of the return stroke in the posterior leg (third influence, caudally directed, i.e. from front to rear); the position in which a leg touches the ground is a target for the posterior leg (targeting influence, caudally directed); the fifth influence is caused by increased loads and given the purely kinematic nature of our model is not considered at the moment, nor is the sixth influence, which represents the treading-on-tarsus reflex.

The selector net, the swing net and the stance net were realized in [Cruse *et. al.* (1998a)] by using artificial neural networks with static neurons. In this work the decentralized control paradigm is implemented as a space distributed architecture of nonlinear dynamical systems representing the leg controllers. This formalization led us to represent the decentralized control through the CNN paradigm. The peculiarities of this system will be examined in the next Section.

#### 4.1.2 *The CNN leg controller*

In the hierarchical control of hexapod robots illustrated in Chap. 2, the periodic movement of each leg is controlled by the oscillating variables of a nonlinear dynamic system. This system corresponds to two coupled motor units responsible for driving the two joints of a leg. The biological system consists of a large number of sensory cells, inter-neurons, and motor-neurons. The CNN neuron has only two control signals that alternate between low and high excitation, needed to assure the correct movement of the leg in stance and in swing.

The dynamics of the motor-neurons is described by Eq. (2.5). It is worth recalling that the output variables,  $y_1$  and  $y_2$ , after appropriate scaling as in Eq. (5.1), are used to drive the joint actuators. More precisely, the leg of the hexapod robot used to illustrate the decentralized locomotion is made up of two links, as shown schematically in Fig. 4.2, and has two DOF actuated by servomotors driven by the variables of system (2.5). Let us recall that the variable  $y_1$  is used to drive the  $\beta$ -joint, therefore regulating the height of the leg, while the variable  $y_2$  drives the  $\alpha$ -joint.

As discussed in Sec. 4.1.1, the decentralized control is based on two kinematic parameters, AEP and PEP, which can be identified on the phase

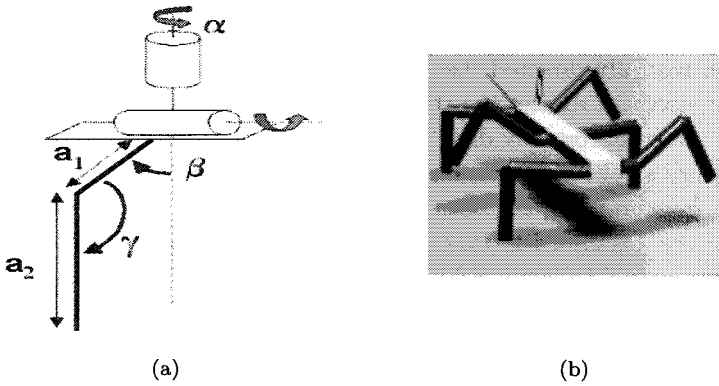


Fig. 4.2 (a) Leg model (two actuated joints) and (b) the hexapod model for decentralized locomotion control.

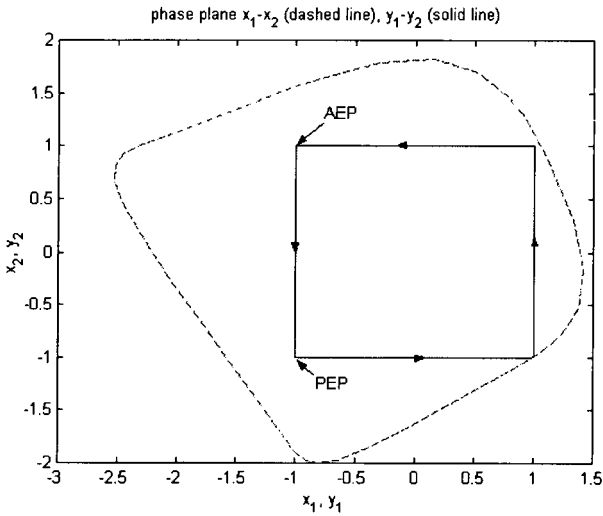


Fig. 4.3 The limit cycle of system (2.5) with nonlinearity (2.6) in the phase planes  $x_1 - x_2$  (dashed line) and  $y_1 - y_2$  (solid line).

plane in Fig. 4.3 where the limit cycle of the CNN neuron in both the phase planes  $x_1 - x_2$  and  $y_1 - y_2$  is shown. To allow these parameters to change, instead of considering nonlinearity (2.6), the following nonlinearity can be taken into account:

$$\begin{aligned}
 y_1 &= \begin{cases} k_1 & \text{if } x_1 \geq k_1 \\ x_1 & \text{if } k_2 < x_1 < k_1 \\ k_2 & \text{if } x_1 \leq k_2 \end{cases} \\
 y_2 &= \begin{cases} k_3 & \text{if } x_2 \geq k_3 \\ x_2 & \text{if } k_4 < x_2 < k_3 \\ k_4 & \text{if } x_2 \leq k_4 \end{cases}
 \end{aligned} \tag{4.3}$$

That is, instead of a clipping function with fixed upper and lower bounds of 1 and -1, respectively, we now have variable upper bounds ( $k_1, k_3$ ) or lower bounds ( $k_2, k_4$ ). Introducing this nonlinearity in system (2.5) leads to an oscillating behavior that can be modified by acting on the parameter vector  $\mathbf{K} = [k_1 \ k_2 \ k_3 \ k_4]$  of the nonlinearity. Figure 4.4 illustrates some examples showing how the original limit cycle (obtained for  $\mathbf{K}=[1 \ -1 \ 1 \ -1]$ ) is modified as a function of  $\mathbf{K}$ .

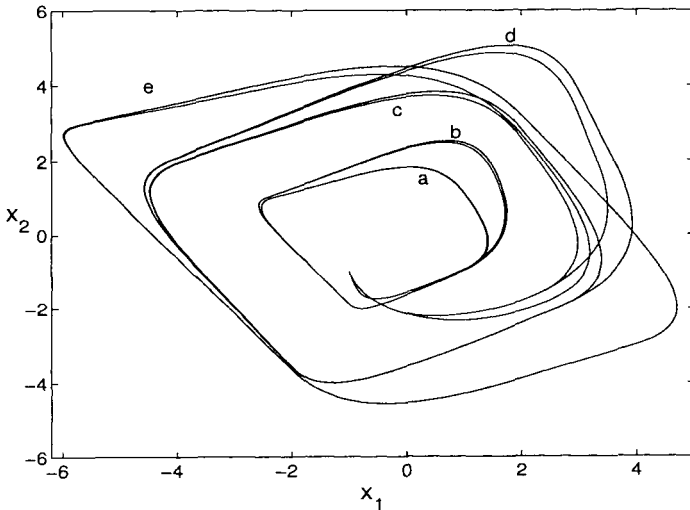


Fig. 4.4 Limit cycle of system (2.5) with nonlinearity (4.3) for different values of  $\mathbf{K}$ :  $\mathbf{K}_a=[1 \ -1 \ 1 \ -1]$ ;  $\mathbf{K}_b=[2 \ -1 \ 1 \ -1]$ ;  $\mathbf{K}_c=[2 \ -2 \ 2 \ -2]$ ;  $\mathbf{K}_d=[3 \ -2 \ 2 \ -2]$ ;  $\mathbf{K}_e=[2 \ -2 \ 3 \ -3]$ .

However, not all the possible values of  $\mathbf{K}$  are such that the limit cycle behavior is maintained. The range of suitable values of  $\mathbf{K}$  can be investigated by considering the conditions for which the Poincaré-Bendixson Theorem holds. The analysis follows that of Chap. 2; in this case the com-

putation refers to the saturation points that are now variable. In order to have a periodic orbit,  $k_3$  and  $k_4$  should be chosen so as to exclude stable equilibrium points in the other remaining regions ( $D_1, \dots, D_8$ ). Solving the system given in Eq. (2.5) in each of these 8 regions, the four non-trivial conditions assuring the non-existence of equilibria inside them lead to the following relations as functions of  $k_3$  and  $k_4$ , which guarantee the existence of a periodic orbit inside  $R$ :

$$\begin{aligned} \frac{-sk_2 - i_2}{\mu} < k_3 < \frac{\mu k_1 + i_1}{s} \\ \frac{\mu k_2 + i_1}{s} < k_4 < \frac{-sk_1 - i_2}{\mu} \end{aligned} \quad (4.4)$$

These parameters can therefore be modulated around fixed values, depending on the state of the other legs. In particular, it is possible to act on  $k_3$  and  $k_4$  to vary the AEP and the PEP, as shown in detail in Sec. 4.1.3.1. In fact, the AEP is reached when the variable  $x_2$  passes the saturation point  $k_3$  and  $y_1$  reaches the low saturation limit  $k_2$ . The stance phase begins at this point. It ends when the PEP is reached: this occurs when the variable  $y_2$  reaches its inferior saturating point  $k_4$ . Therefore, to change PEP or AEP, the parameters  $k_4$  or  $k_3$  have to be varied. Their modulation rules are derived following the influences known to control stick insect walking [Cruse (1990)]. In particular, to match the method adopted in Walknet and that based on CNNs, the local connections among the CNN cells influence the parameters  $k_3$  and  $k_4$ . Therefore, for each cell, these two parameters modulating its clipping function are themselves functions of the neighboring cell state variables.

### 4.1.3 The whole control system and results

#### 4.1.3.1 CNN Decentralized Control

In this Section it is shown that the CNN paradigm is suitable for the implementation of the whole locomotion control system. To this end the local influences are re-examined. An overview of the whole control system, according to the Walknet model, is given in Fig. 4.5, where the influences among the CNN cells are indicated. These have been assumed to be of the same type as those used in other works dealing with the decentralized control of the stick insect [Cruse (1990)]. Figure 4.6 shows the nonlinearities involved in the local influences: the subscripts  $i$  and  $c$  refer to ipsilateral (between legs on the same side of the insect) and contralateral (between cor-

responding opposite legs, for example R2 and L2) influence, respectively, while the inputs of these nonlinearities will be discussed in detail below. The parameters appearing in the nonlinearity graphs were chosen by using an optimization strategy, as will be explained later on. The inputs of the influences are Ground Contact (GC), which is a binary signal coding if a leg touches the ground or not; the (normalized) displacement of the leg in the motion direction,  $Y$ ; and a speed parameter,  $v$ , indicating the speed of the insect. The input of the first type of influence is GC, the input of influence 2 is the product of GC and the displacement  $Y$ , the input of the third influence is  $X = Y_{R1}GC_{R1} + \frac{v-0.5}{2}$  or  $X = Y_{R1}GC_{R1} + i_{log} \log(5v - 1.5)$  (contralateral and ipsilateral, respectively), and the input of influence 4 is the product of GC and  $Y$ .

It should be pointed out that for influence 2 the nonlinearity follows a low-pass filter cascaded by a high-pass filter, and therefore the whole influence nonlinearity will be referred to as  $\tilde{h}_2(\cdot)$ .

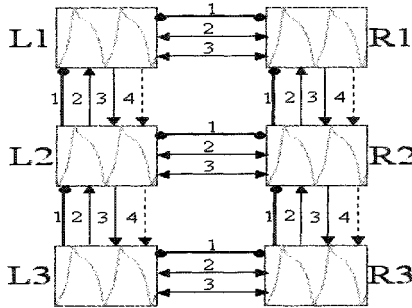


Fig. 4.5 Scheme of local influences: arrows indicate excitatory influences, while dots indicate inhibitory influences. Influences acting on the PEP are indicated with solid lines, those on the AEP with dashed lines.

The local influences act on the leg controller, represented by Eq. (2.5) and (4.3), modifying  $k_3$  as regards the influence on the AEP (influence 4) or  $k_4$  as regards the influences on the PEP (influences 1-3), according to the scheme shown in Fig. 4.5. Therefore, the output nonlinearity  $y_2$  of each cell (while the output  $y_1$  is not affected by the neighboring cells) can be written as:

$$y_{2,ij} = f(x_{2,ij}, k_{3,ij}, k_{4,ij}) \quad (4.5)$$

which constitutes a slight generalization of the CNN output definition. The parameters  $k_3$  and  $k_4$  are modulated by the values of the outputs of neighboring cells and the sensory feedback. In more detail, the dependence of  $k_3$  and  $k_4$  on the neighboring cells can be explained by considering the scheme in Fig. 4.5 and the nonlinearities in Fig. 4.6. For example, let us consider the controller of leg L1. The leg controller receives contralateral influences from the controller of leg R1 (influences 1-3) and from the ipsilateral leg L2 (influences 1-2). All these influences act on the PEP. Therefore, the nonlinearity  $y_{2,L1}$  is given by:

$$y_{2,L1} = \begin{cases} \bar{k}_3 + \Delta_{k_3,L1} & \text{if } x_{2,L1} \geq \bar{k}_3 + \Delta_{k_3,L1} \\ x_{2,L1} & \text{if } \bar{k}_4 + \Delta_{k_4,L1} < x_{2,L1} < \bar{k}_3 + \Delta_{k_3,L1} \\ \bar{k}_4 + \Delta_{k_4,L1} & \text{if } x_{2,L1} \leq \bar{k}_4 + \Delta_{k_4,L1} \end{cases} \quad (4.6)$$

where  $\Delta_{k_3,L1} = 0$  since no influence acts on the AEP of L1 and

$$\Delta_{k_4,L1} = h_{1c}(GC_{R1}) + \tilde{h}_{2c}(Y_{R1}GC_{R1}) - h_{3c}(Y_{R1}GC_{R1} + \frac{v-0.5}{2}) + h_{1i}(GC_{L2}) + \tilde{h}_{2i}(Y_{L2}GC_{L2}).$$

The output nonlinearities of the other cells of the CNN implementing the whole decentralized control can be obtained in a similar fashion.

As can be noticed, all the nonlinearities involved in the CNN model of decentralized control are piecewise linear with flat end segments. This ensures the stability of the control system.

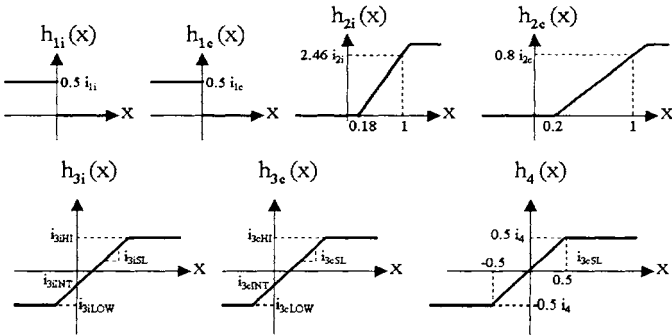


Fig. 4.6 Nonlinearities of local influences between leg controllers.

#### 4.1.3.2 Choice of the parameters of the model

For the first and second influence four parameters should be taken into account:  $i_{1i}$ ,  $i_{1c}$ ,  $i_{2c}$  and  $i_{2i}$ . These describe the strength of the effect of influence 1 and influence 2 acting between ipsilateral legs or between contralateral legs, respectively. For the third influence four parameters for the ipsilateral and four for the contralateral influence were taken into account. Two of these four parameters describe the strength of the dependency on the position of the leg sending the information (slope, ordinate offset:  $i_{3SL}$  and  $i_{3Int}$ ), and two ( $i_{3Hi}$  and  $i_{3Low}$ ) describe an upper and lower bound limiting this effect. Finally, two further parameters ( $i_4$  and  $i_{log}$ ) were included, namely the strength of the fourth influence and the strength of the pattern dependency on speed. The value of  $k_2$  was also varied. Altogether 15 parameters were allowed to vary.

As discussed above, in the stick insect there is a continuous transition between the two locomotion patterns as a function of the speed parameter: when the speed parameter  $v$  is in the range of 0.5-0.7, tripod gait is the pattern adopted by the stick insect, while when the speed parameter assumes values in the range of 0.35-0.5 tetrapod gait arises. We considered the problem of finding a set of parameters for tripod gait and tetrapod gait separately. This was the first step to find a single set of parameters for both patterns and allows us to show that local influences are suitable even when a dynamical system is assumed as the leg controller. This is a first step that allows us to prove that local influences can be a suitable way to model insect gait, even when the dynamics of each cell changes completely.

As a first approach we applied a heuristic (trial and error) procedure to find a set of parameters. This was shown to be suitable for tripod gait.

To find suitable parameters for tetrapod gait, a heuristic procedure was not sufficient, so genetic algorithms [Goldberg (1989)] were used. They are an optimization algorithm based on the paradigm of natural selection. The algorithm operates on a population of individuals: each individual is represented by a string containing the parameter set. After the random generation of the initial population, the algorithm evolves through three operators: selection (survival of the fittest), crossover (mating between individuals) and mutation (introducing random modifications) to achieve a population containing the optimal solution with respect to a fitness function which will be specified below. For each set of parameters a simulation of the model was carried out. To allow a degree of randomness and at the same time to avoid starting from unnatural initial conditions, fixed values

with small random additive terms (10% of the nominal value) were chosen as initial conditions for this simulation. The fitness function takes into account how well the simulation matches the given pattern. If  $\Phi_{L2}$   $\Phi_{L3}$   $\Phi_{R1}$   $\Phi_{R2}$   $\Phi_{R3}$  indicate the phase lags of the target pattern and  $\phi_{L2}$   $\phi_{L3}$   $\phi_{R1}$   $\phi_{R2}$   $\phi_{R3}$  indicate the phase lags obtained in the simulation, the fitness function  $f$  is chosen as follows:

$$f = \sum_i \sum_{k=1}^n (\phi_i(k) - \Phi_i(k))^2 \quad (4.7)$$

where

$$i \in \{L2, L3, R1, R2, R3\}$$

and the index  $k$  takes into account an average of the last  $n$  periods of oscillations.

Similar results can be obtained if, in addition to Eq. (4.7), a further term is taken into account, i.e. the error achieved on the duty factor.

Figures 4.7 and 4.8 show the gaits obtained by simulating the model. Figure 4.7 refers to tripod gait with phase lags as in Eq. (4.1) and parameters obtained by trial and error as shown in Table 4.1, while Fig. 4.8 shows tetrapod gait obtained by considering the phase lags as in Eq. (4.2) and by using the parameters obtained with genetic algorithms and shown in Table 4.2.

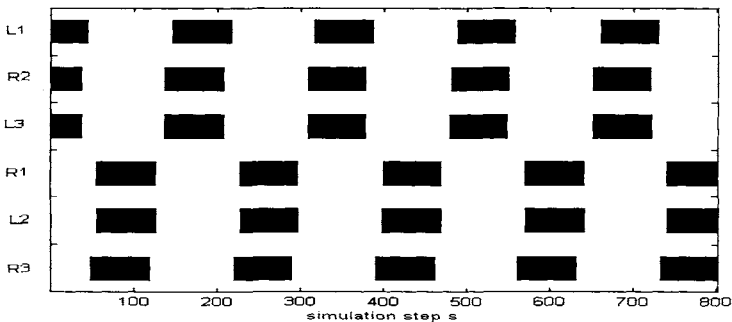


Fig. 4.7 The step pattern of tripod gait as produced by the CNN model of the locomotor control system of the stick insect. The six traces indicate the six legs (from the top: L1, R2, L3, R1, L2, R3); black boxes denote the swing phase.



The pattern shown in Fig. 4.8 was obtained by running genetic algorithms with a population of 30 individuals for 100 generations. With the choice of the fitness function as in Eq. (4.7), minima of the problem are searched for. The fitness of the best individual is  $f = 5.9 \cdot 10^{-4}$ . The results obtained fit the tetrapod gait of the stick insect very well, like those obtained with Walknet.

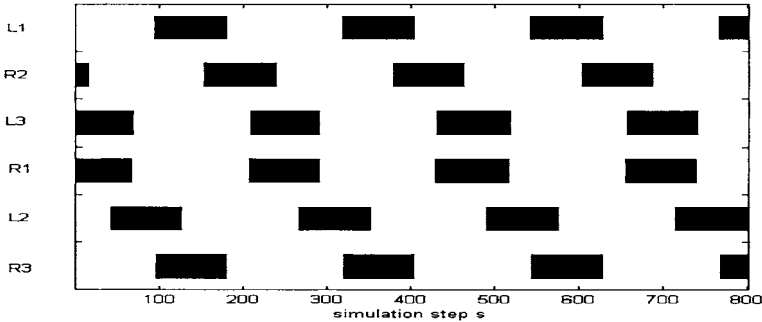


Fig. 4.8 The step pattern of tetrapod gait as produced by the CNN model of the locomotor control system of the stick insect (parameters obtained running genetic algorithms with a population of 30 individuals for 100 generations). The six traces indicate the six legs (from the top: L1, R2, L3, R1, L2, R3): black boxes denote the swing phase.

Table 4.1 Parameters of the CNN controller: tripod ( $k_2 = -3.8$ ).

parameter	$i_{1i}$	$i_{1c}$	$i_{2c}$	$i_{2i}$	$i_{3cHi}$	$i_{3cInt}$	$i_{3cLow}$
value	3	2	8	4	0	0	-3
	$i_{3cSL}$	$i_{3iHi}$	$i_{3iInt}$	$i_{3iLow}$	$i_{3iSL}$	$i_4$	$i_{log}$
	3	0	0	-1	1	1	0.2

Table 4.2 Parameters of the CNN controller: tetrapod ( $k_2 = -4.5$ ).

parameter	$i_{1i}$	$i_{1c}$	$i_{2c}$	$i_{2i}$	$i_{3cHi}$	$i_{3cInt}$	$i_{3cLow}$
value	1.3647	1.6824	0	0.5882	0.5000	0	-2.7235
	$i_{3cSL}$	$i_{3iHi}$	$i_{3iInt}$	$i_{3iLow}$	$i_{3iSL}$	$i_4$	$i_{log}$
	0.5000	5.0000	0	-0.5000	10.0000	0	0.1459

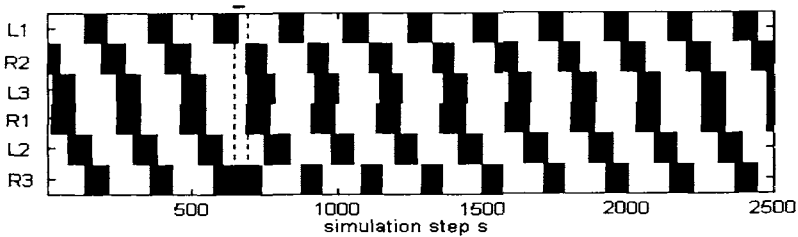
#### 4.1.3.3 Robustness of the CNN Decentralized Controller

The control system with the parameters obtained as described above was simulated in different conditions to test its robustness. The results discussed here deal with tetrapod gait (in all the simulations the parameters are kept fixed to the values given in Table 4.2).

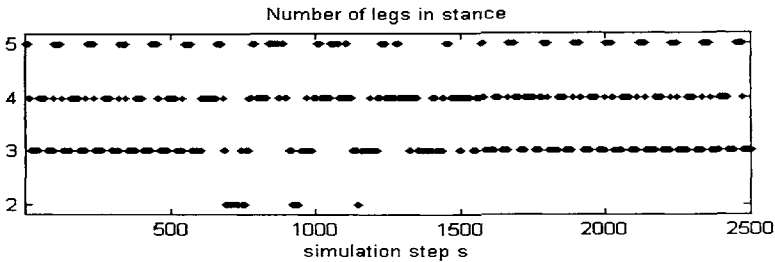
The first experiment deals with conditions similar to those reported in [Cruse *et. al.* (1998a)] and is devoted to testing the behavior of the system when a leg is blocked for a short time due to external disturbances. The model presented here shows the same capability of pattern recovery as in Walknet. The step pattern resulting from this experiment is shown in Fig. 4.9(a), while Fig. 4.9(b) shows the number of legs that are in stance at the same time as a function of the simulation step  $s$ . As can be noticed, at approximately  $s = 1100(\text{samples})$  the right middle leg ( $R2$ ) should begin its swing phase, but it is externally forced to remain in stance. After a few cycle periods the pattern is recovered. A measure of the error induced by this disturbance in the gait, expressed in terms of the fitness function in Eq. (4.7) with  $n = 1$ , was considered. This is shown in Fig. 4.10 as a function of the gait cycle of the insect model: the leg  $R2$  is blocked at the fourth cycle causing a sudden increase in the phase lag error; after 4 cycles the phase lags fit those of the ideal tetrapod gait (4.2) very well.

Attention was also paid to the minimum number of legs in stance during the whole simulation. This is fundamental since static stability is assured if three (with at least one leg for each side) or more legs are in stance. Analysis of Fig. 4.9(b) reveals that the period when only two legs are on the ground corresponds to a small fraction of a swing phase and takes place soon after the disturbance.

Finally, the robustness of the control system to different initial conditions of the CNN cells was tested. The parameter set was found by running genetic algorithms with random perturbations around nominal values of about 10%. This leads to the parameter set in Table 4.2, thus the control system already includes some robustness properties. To further investigate the robustness of the control system with parameters fixed to the values in Table 4.2, random perturbations (ranging from 0% to 80%) on the initial conditions of the CNN were considered. For each value of random perturbations (ranging from 0% to 80%) a set of 10 simulations of the control system was performed. The results are illustrated in Fig. 4.11, where the logarithm of the fitness function value as in Eq. (4.7) is given for the 10 trials for each of the different values of the random perturbation on the



(a)



(b)

Fig. 4.9 Recovery of tetrapod gait after the blocking of a leg: leg *R2* is blocked for the period indicated by the two dashed marks: (a) the step pattern (black boxes denote the swing phase); (b) number of legs in stance versus the simulation step *s*.

initial conditions.

As can be noticed, the pattern is quite insensitive to the initial conditions. Only large perturbations (80%) lead to not well-defined patterns (when the fitness function takes values above  $10^{-2}$ ), in which in any case a form of locomotion is still present.

Static stability was also taken into consideration in the investigations into the system's robustness to different initial conditions. For some initial conditions, the gait pattern may not be well defined in the first part of the simulation and for short periods (not longer than those examined in the case of leg blocking) only two legs are in stance. However, this is a conservative condition, and a more accurate analysis including dynamic constraints should be performed to investigate when this condition can cause the robot to fall. In any case, this can be taken into account in the choice of the fitness function.

The experiments discussed here show an important feature of the control system: its ability to deal with disturbances. Of course, even if the

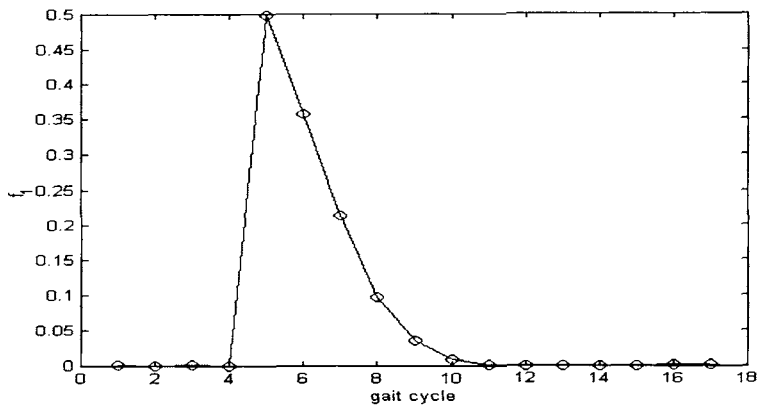


Fig. 4.10 Recovery of tetrapod gait after the blocking of a leg: error  $f_1$  between target and actual pattern versus gait cycles.

experiments deal with a particular kind of disturbance, the recovery capabilities are independent of the source of disturbance. Moreover, the genetic algorithm procedure gives a population of possible solutions among which the best is chosen, showing the intrinsic robustness of the evolution-based method in finding the parameters. The CNN neuron has also been shown to be robust to parameter changes [Arena *et. al.* (1999)]. All these considerations guarantee the feasibility of a hardware realization of the locomotion control.

## 4.2 Integrate-and-fire neurons and decentralized control

The aim of this Section is to propose another implementation of decentralized locomotion control. In particular this realization has two objectives:

- to study a network of spiking neurons and their suitability for locomotion control;
- to implement a scheme totally based on reflexes.

It is well-known that spiking neurons are involved in the neural control of locomotion [Orlovsky *et. al.* (1999)]. Therefore using spiking neurons adds biological plausibility to the control scheme. Moreover, this allows to realize an analog control system totally relying on reflexes, i.e. there is no

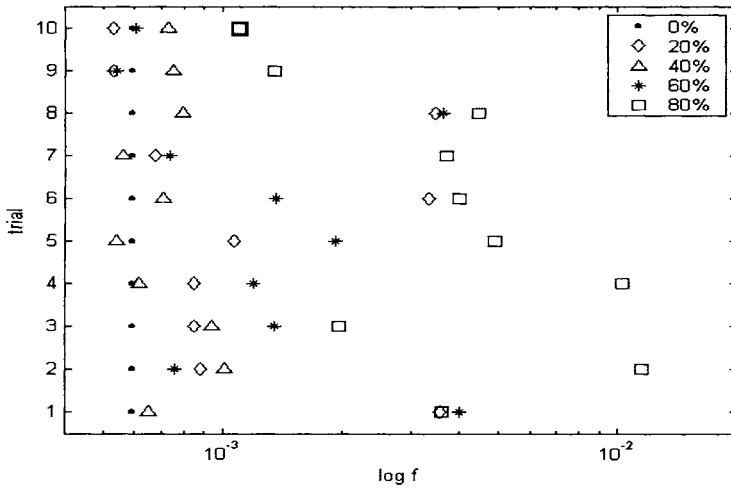


Fig. 4.11 Robustness of the pattern with respect to initial conditions: 10 trials were performed for each value of the perturbation percentage.

intrinsic oscillation in the leg controller (the swing/stance periodic sequence is induced by sensors), whereas in the CNN model each leg controller is still an oscillator. Therefore, the new scheme essentially differs from the CNN model in the leg controller design, whereas interlimb coordination is achieved by means of the same local interactions. The rhythmic movements are the result of the interaction between the leg dynamics and the control system. This leads to a totally different leg controller design, that strictly requires sensory feedback for the generation of leg movements.

To implement spiking neurons the well-known integrate-and-fire (IF) model is assumed. It should be remarked that IF neurons have already been used in bio-inspired systems [Lewis *et. al.* (1998)], where they constitute the last stage of a CPG for bipedal locomotion. The hierarchical organization of the motor control of [Lewis *et. al.* (1998)] consists of a pair of mutually inhibited oscillators driving two IF motor-neurons. The aim of our work is to further exploit spiking neurons for locomotion control and to build up a network consisting exclusively of IF neurons able to control the locomotion of a hexapod robot.

### 4.2.1 *The leg controller*

The model for the leg controller is now presented. As discussed above, this is the core of the new scheme. The model does not reproduce any particular leg controller of a given insect, but relies on a few biological motivations, common to many arthropods and which will be examined below.

The fundamental unit of the leg controller is the IF neuron, a very simple model of a spiking neuron. No intrinsic oscillations are included in the model and the rhythm of the locomotion is induced by the status of the sensors.

#### 4.2.1.1 *Biological motivations*

The neural control of muscles takes place in a very different way in invertebrates and vertebrates. While in vertebrates a number of neurons innervate the muscles, in arthropods typically few neurons innervate the muscles. For example, as regards the muscles controlling the extension of the femur, the studies carried out by Pearson and Iles [Pearson and Iles *et. al.* (1971)], based on electromyograms, reveal that only two motor-neurons (fast and slow depressors) innervate these muscles.

It should also be remarked that animal movements are very often regulated by pairs of antagonistic muscles. The main blocks present in the model of the leg controller considered here reflect these considerations. They are: a “muscle block”, that plays the role of either the pair of antagonistic muscles or (in the robot) the servo-motor controlling the joint; the “motor-neurons”, two for each joint (depressor and extensor), and the “inter-neurons”. The motor-neurons drive a muscle block, while the inter-neurons are not connected to any joint. When a motor-neuron emits a spike, the muscle block integrates the spike and the joint moves according to the direction established by the kind of motor-neuron (depressor or extensor).

#### 4.2.1.2 *The integrate-and-fire neuron*

The dynamics of accumulation and fire is common to different phenomena. Figure 4.12(a)-(c) shows a series of examples: a vessel that is slowly filled by water and is quickly emptied when the water reaches the threshold level; a neuron whose electric potential across its membrane slowly charges until it emits a spike (a huge current pulse discharging the membrane potential in a very short time) and the van-der Pol relaxation generator, in which the charge of the capacitor slowly grows until the threshold for the gas discharge

in the neon lamp is reached [Pikovsky *et. al.* (2001)]. The common feature of these systems is the presence of two different time scales: slow and fast motion alternate within an oscillation period. These systems are known as relaxation oscillators. The main building block of our leg controller, the neuron, represents a relaxation oscillator.

The equations of the IF neuron are the following:

$$\dot{x} = I_{sp} + \sum I_{syn} + I_{dis}h(x) \quad (4.8)$$

where  $x$  is the membrane potential,  $I_{sp}$  is a spontaneous current,  $I_{syn}$  is the synaptic current from other neurons,  $I_{dis}$  is the discharge current and  $y = h(x)$  is a hysteresis:

$$y = h(x) = \begin{cases} 1 & x \geq Th_H \\ 0 & x \leq Th_L \end{cases} \quad (4.9)$$

where  $Th_L$  and  $Th_H$  indicate the low and high threshold respectively.

By choosing  $I_{dis} \gg I_{sp}$  the dynamics of the neuron, in absence of other currents, is regulated by two different time scales: the small  $I_{sp}$  slowly charges the membrane, while the large  $I_{dis}$  rapidly discharges the membrane. It offers a good example of a relaxation oscillator. Moreover, far from reproducing the complex current exchanges occurring in the membrane of real neurons, they capture their essential dynamics, offering a biologically plausible model more than an accurate, but complex model.

The typical trends of the variables  $x$  and  $y$  are shown in Fig. 4.12(d). As can be noticed, the trend of the variable  $y$  is characterized by spikes occurring at instants when  $x$  reaches the threshold value. As mentioned above, spiking neurons are involved in the neural control of locomotion. This is the reason why we will think of the variable  $y$  as the output of the neuron and consider synaptic connections between the neurons realized through this output.

#### 4.2.1.3 Scheme of the leg controller

The leg controller provides the leg with the signals for proper locomotion. The leg has two DOF ( $\alpha$ -joint and  $\beta$ -joint in Fig. 4.2). The two phases, stance and swing, alternate on the basis of the sensory feedback. Two sensory signals are fundamental, the GC and the PEP-AEP. Both signals are binary coded: the GC is high when the leg tip touches on the ground,

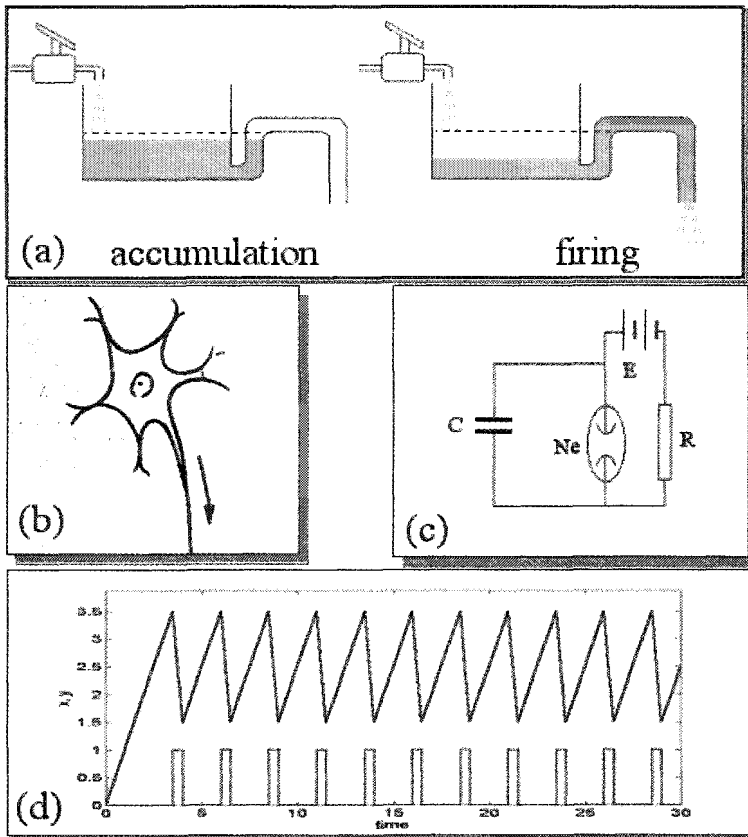


Fig. 4.12 (a)-(c) Examples of relaxation oscillator. (d) Slow-fast dynamics of the relaxation oscillator.

thus indicating that the stance phase should begin; the PEP-AEP is high when the PEP is reached, thus indicating that the swing phase should begin.

During the stance phase the motor-neuron PS (power stroke neuron) is spiking, while during the swing phase the motor-neuron RS (return stroke neuron) is active. These motor-neurons drive the  $\alpha$ -joint. Since during stance and swing this joint has to be moved in opposite directions, these motor-neurons act on the joint with opposite signs, meaning that when, for instance, RS emits a spike, the joint moves according to the forward direction of robot motion. Hence, the activities of these two neurons in



a certain sense define the beginning and the end of the power stroke and return stroke, the pattern of their activities being characterized by an alternating bursting of the two neurons.

This behavior is due to the connections between sensors and neurons present in the leg controller network (shown in Fig. 4.13). These connections were established on the basis of the following considerations. For instance, since the GC indicates the beginning of the power stroke, it seems clear that it should excite the PS neuron and inhibit the RS neuron. Analogously, the PEP-AEP signal should excite the RS neuron (the return stroke initiates since the leg has reached the PEP) and inhibit the PS. Mutual inhibition between neuron RS and PS reinforces the anti-phase bursting of these neurons. The second actuated joint (the  $\beta$ -joint) is driven by two other motor-neurons, labelled E (extensor, lifting the leg) and F (flexor, lowering the leg). Neuron E should clearly be inhibited by the neuron PS, since the leg should not lift during the power stroke.

The leg controller model includes other neurons. These are interneurons, since they do not directly synapse any joint actuator. Their role is, however, important. For instance, neuron  $E_{IN}$  is excited by neuron E and inhibits it. This kind of connection is quite common in real neural controllers [Orlovsky *et al.* (1999)] and results in a pattern of activities, in which neuron E emits a burst of spiking and then stops. In our model it has been assumed that neuron  $E_{IN}$  is a “slow” neuron: it is slowly charged by the excitatory input provided by the synapse with neuron E.

The behavior of the whole network is illustrated in Fig. 4.14. A rhythmic pattern is evident. It is induced by the rhythmic activation of the sensory feedback signals GC and PEP-AEP. Moreover, the pattern is periodic since it is assumed that the leg is on flat terrain and the outputs of GC and PEP-AEP are regular.

Each connection in Fig. 4.13 is characterized by a weight and a sign, negative if the synapse is inhibitory (indicated with a filled dot in Fig. 4.13) or positive if the synapse is excitatory (indicated with an arrow). The weights of these connections constitute the parameters of the leg controller and were obtained by trial and error according to the considerations outlined above. Table 4.3 gives these parameters, while Table 4.4 deals with the parameters of the neurons of the leg controller: spontaneous current  $I_{sp}$ , discharging current  $I_{dis}$  and threshold values according to Eq. (4.8).

Finally, Fig. 4.15 gives the typical trends of the PS and RS outputs and of the position  $xL1$  of the tip of leg L1 in the direction of motion. The position  $xL1$ , varying between AEP and PEP, goes from AEP to PEP in

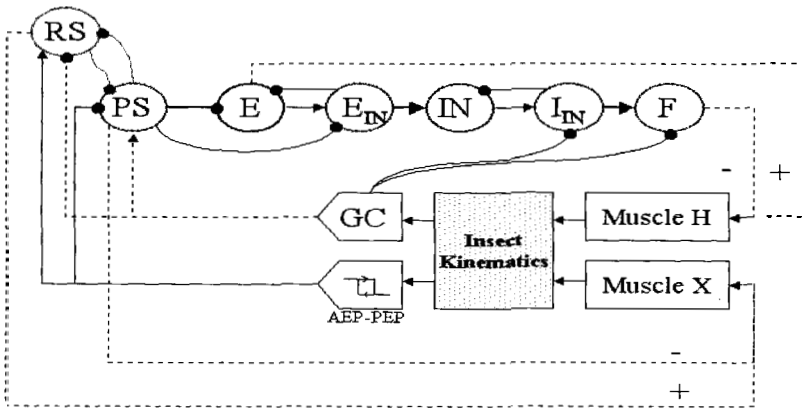


Fig. 4.13 Scheme of the IF leg controller.

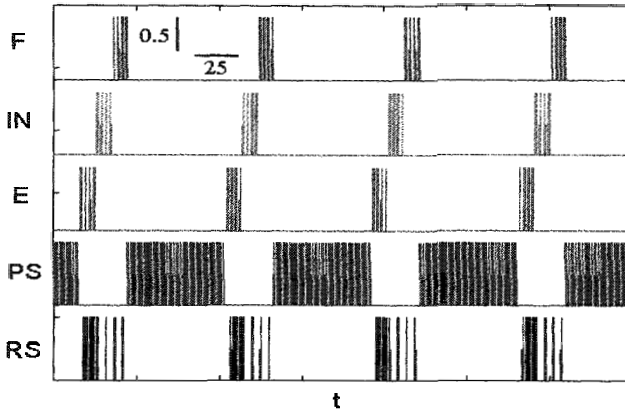


Fig. 4.14 Signals from the leg controller: the output of the motor-neurons RS and PS controlling leg protraction and retraction, the output of motor-neuron E and F controlling the extension of the leg and the inter-neuron E are shown (normalized units are assumed).

the stance phase (when neuron PS fires) and from PEP to AEP during the first phase of the return phase (when neuron RS fires). Two complete stepping movements are shown.

Table 4.3 Weights of the connections involved in the scheme in Fig. 4.13 and leading to the behavior shown in Fig. 4.14.

From	RS	PS	E	E <sub>IN</sub>	IN	I <sub>IN</sub>	F	GC	AEP, PEP
RS	0	-4	0	0	0	0	0	-1.5	1.5
PS	-4	0	0	0	0	0	0	1.5	-1.5
E	0	-4	0	-1	0	0	0	0	0
E <sub>IN</sub>	0	-1	1	0	0	0	0	0	0
IN	0	0	0	1	0	0	0	0	0
I <sub>IN</sub>	0	0	0	0	1	0	0	-1	0
F	0	0	0	0	0	1	0	-1	0

Table 4.4 Parameters of the neurons constituting the IF leg controller.

Neuron	$I_{sp}$	$I_{dis}$	$Th_L$	$Th_H$
RS	0.5	5	0	3.5
PS	0.5	5	0	3.5
E	1	5	1.5	3.5
E <sub>IN</sub>	0	0.05	1.5	3.5
IN	0	5	1.5	3.5
I <sub>IN</sub>	0	0.005	1.5	3.5
F	0	5	1.5	3.5

#### 4.2.1.4 The elevator reflex

As discussed previously, there are a number of local reflexes which are fundamental for proper locomotion on rough terrain. The IF leg controller can easily be modified to include other reflexes, such as the elevator reflex.

To this end two new neurons are introduced in the leg controller network. In Fig. 4.16 the connections between the previously introduced neurons (R and R<sub>IN</sub> with parameters as in Table 4.5) and the other neurons of the leg controller, as well as the weight of these connections are drawn. The elevator reflex sensor represents a generic sensor whose output is normally high and goes down when the leg touches an obstacle. Neuron R is inhibited by neuron PS since the elevator reflex occurs only during the return stroke. When the reflex is activated, neuron R begins firing. It excites neuron PS, and the leg rapidly retracts. Then neuron R stops firing and a return stroke begins (this is induced by neuron R<sub>IN</sub>). As can be noticed in Fig. 4.17 showing the neuron outputs of a typical case, the leg lifts higher than normal. Figure 4.17 refers to a simulation in which only one leg of the dynamic hexapod model (shown in Fig. 4.16(a) and realized in VisualNas-

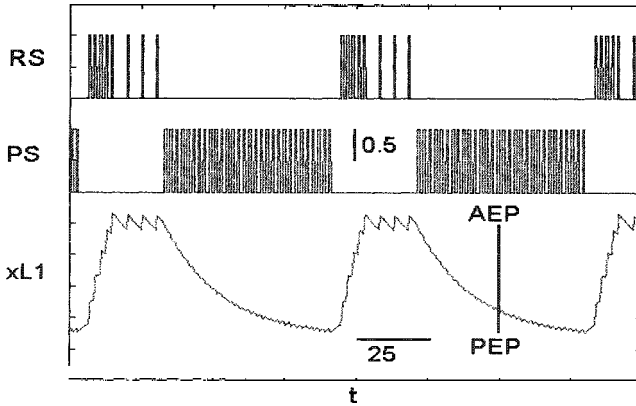
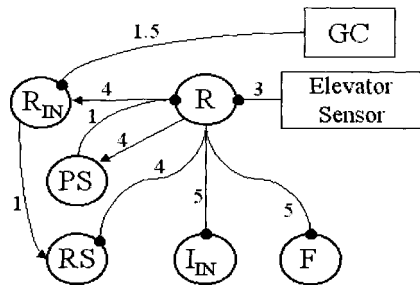


Fig. 4.15 Signals from the leg controller: the position of the tip of the leg  $L1$  in the direction of motion ( $xL1$ ) varies between AEP and PEP: it goes from AEP to PEP in the stance phase (when neuron PS fires), while it goes from PEP to AEP during the first phase of the return phase (when neuron RS fires). Two complete stepping movements, referring to tetrapod gait, are shown.

tran) is controlled; the obstacle is on a conveyor belt moved by the leg of the still hexapod.



(a)



(b)

Fig. 4.16 (a) Dynamic model of the hexapod. (b) Scheme of the IF network implementing the elevator reflex.

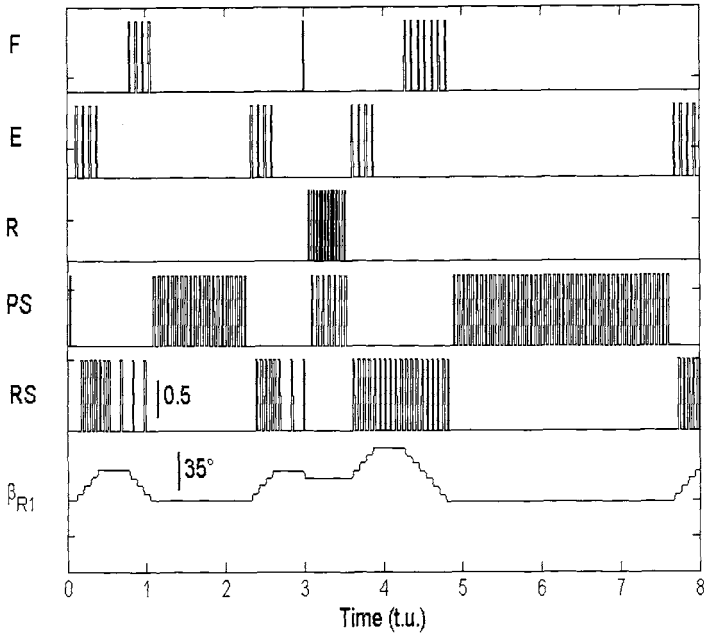


Fig. 4.17 Waveforms of the neuron outputs of the leg controller when the elevator reflex is activated. The trend of the signal controlling the  $\beta$ -joint is given to show how the leg lifts higher than normal.

Table 4.5 Parameters of the IF neurons for elevator reflex.

Neuron	$I_{sp}$	$I_{dis}$	$Th_L$	$Th_H$
R	3	10	0	3.5
R <sub>IN</sub>	0	2.5	1.5	3.5

#### 4.2.2 The whole control scheme

The leg controllers are coordinated by means of the influences described previously and shown in Fig. 4.6. Of course, the parameters have to be adapted to the new dynamics of the leg controller. This was done by following the same approach as that used for the CNN model. However, to account for both tetrapod and tripod gait the fitness function (4.7) was

modified. Now a longer simulation is considered to evaluate the fitness of an individual. In the first part of the simulation the parameter  $v$  is fixed to  $v = 0.35$ ; this value is typical of tetrapod gait, so the fitness is evaluated as in Eq. (4.7) with the tetrapod phase lags. Then the speed  $v$  is switched to the value  $v = 0.7$ , typical of tripod gait, and a new term is calculated by considering Eq. (4.7) again and the set of tripod phase lags (4.1). The overall fitness function takes into account the two terms:

$$f = f^{tetra} + f^{tri} = \sum_i \sum_{k=n-3}^n (\phi_i(k) - \Phi_i^{tetra}(k))^2 + \sum_i \sum_{k=m-3}^m (\phi_i(k) - \Phi_i^{tri}(k))^2 \quad (4.10)$$

where  $n$  and  $m$  indicate the last period of oscillations (stepping movement) at the end of the tetrapod and tripod phase, respectively. Therefore, as for the CNN model, an average of four stepping movements for both tetrapod and tripod gait was considered.

It should be remarked that the tetrapod gait considered here is slightly different from Eq. (4.2). The leg  $R3$  and  $L1$  are slightly out of phase. The same holds for  $R1$  and  $L3$ , as follows:

$$\Phi_{L2} = \frac{2}{3}; \Phi_{L3} = \frac{1}{3}; \Phi_{R1} = \frac{1}{2}; \Phi_{R2} = \frac{1}{6}; \Phi_{R3} = \frac{5}{6}. \quad (4.11)$$

Moreover, the velocity  $v$  also acts on the leg controller parameters. The spikes of motor-neuron PS and those of motor-neuron RS are weighted differently according to a parameter (that has been called  $sf$ ) regulating the ratio between the duration of the stance and swing phases. This parameter is a function of the speed of the gait according to  $sf = \frac{v}{1.75}$ .

Table 4.6 shows the optimized parameters of the network of IF neurons, while Fig. 4.18 illustrates a simulation of the step pattern as obtained with the IF controller. The gait changes at  $t = 8000(\text{samples})$ .

Figures 4.19 and 4.20 are a zoom of tetrapod and tripod walking, respectively.

Table 4.6 Influence parameters for the IF controller.

parameter value	$i_{1i}$ 0.41	$i_{1c}$ 0.016	$i_{2c}$ 0	$i_{2i}$ 0.5041	$i_{3cHi}$ 0	$i_{3cInt}$ 0	$i_{3cLow}$ -0.1402
	$i_{3cSL}$ 0.0735	$i_{3iHi}$ 0.1882	$i_{3iInt}$ 0	$i_{3iLow}$ -0.2422	$i_{3iSL}$ 0.0539	$i_4$ 0	$i_{log}$ 0.1

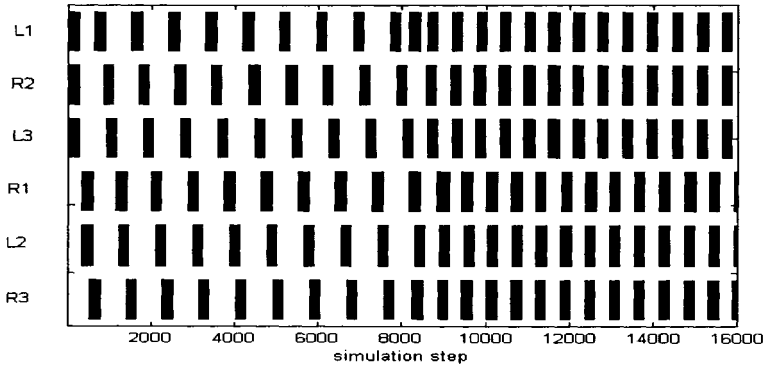


Fig. 4.18 Step pattern of the gait produced by the IF controller: the gait changes from tetrapod to tripod at  $t = 8000$ (samples).

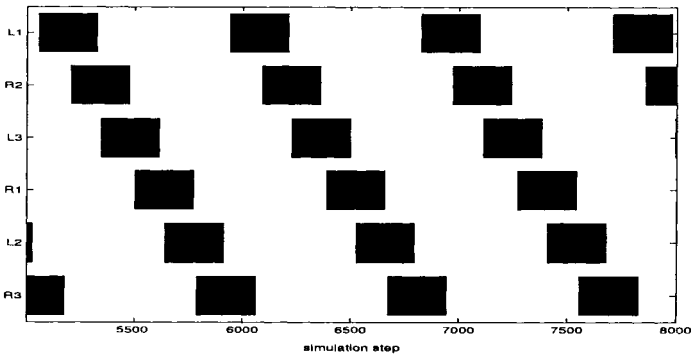


Fig. 4.19 Step pattern of the gait produced by the IF controller: tetrapod gait.

### 4.3 CPG and decentralized control

In stick insect locomotion control as modelled in Walknet, a set of local influences among single leg controllers, realized through artificial neural networks, was used to control the locomotor system, allowing switching among different patterns. In this Chapter a new approach has been introduced: the main features of the Walknet model have been implemented by using dynamical systems to realize the leg controllers. This allows us to conclude that local influences can be a suitable way to model stick insect

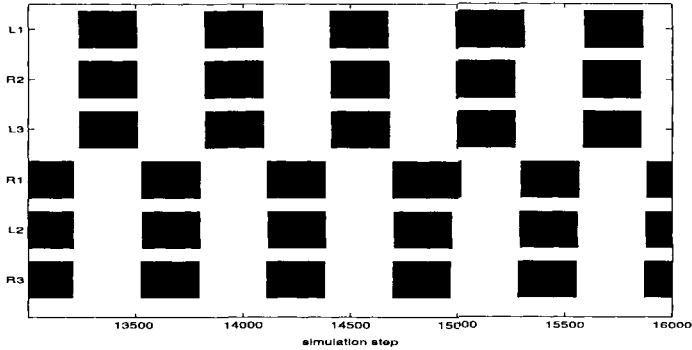


Fig. 4.20 Step pattern of the gait produced by the IF controller: tripod gait.

gait, even when the dynamics of each cell changes completely. In our case the single cell dynamics are much more complex than in the previous decentralized control scheme. The results obtained allow us to state that the self-organized dynamics that regulate insect gait depend on the connections among the cells more than on the single cell dynamics.

The results obtained also open up the way to model other, more complex, local dynamics that could be useful to represent joint motions, while assuring an overall organized spatio-temporal dynamics in the form of a general motion pattern.

The CPG and the decentralized locomotion control presented in this book share two important features:

- the approach is distributed;
- they were implemented by using arrays of nonlinear circuits.

However, the role that feedback plays in the two schemes is very different. The CPG is able to control the robot in absence of feedback, whereas in decentralized locomotion control without feedback there is no coordination among legs (in the CNN-based model) or within a step movement (in the IF-based model). The so-called body instantiation problem [Gallagher *et. al.* (1996)] is therefore quite different in the two schemes. It has been shown that the decentralized model has some recovery capability that makes it robust to noise on the sensors, but of course correct coordination requires, for instance, reliable ground contact sensors. This was also clear in HexaDyn experiments in which interlimb coordination is lost when the ground contact feedback is not correct.



## Chapter 5

# A gallery of bio-inspired robots

In this part of the book several examples of bio-inspired robots are shown. The first example is a lamprey-like robot driven by an RD-CNN. Then several prototypes of hexapod robots are illustrated. The first one, called MTA hexbot I, is an example of an autonomous robot implementing the methodology based on MTA introduced in Chap. 2. This robot has two degrees of freedom for each leg. The second example is a hexapod robot with a more complicate leg design; each leg has now three degrees of freedom. Moreover, the robot is remote-controlled and the direction control discussed on Chap. 3 is implemented. The last example shows a hexapod robot controlled by the CNN-based CPG VLSI chip.

### 5.1 Lampbot: A lamprey robot controlled by RD-CNN

The lamprey is an eel-like fish that swims by rhythmic undulations of its body, which can be divided into a Cerebral Trunk (CT) and a number of Spinal Cord Segments (SCS). Like other fish, a lamprey swims by progressively contracting its muscles via undulatory, wavelike motions from head to tail. Swimming involves coordinated sequences of muscle contractions regulated by the nervous system. The neural control system for swimming in the lamprey spinal cord has been efficiently unravelled by Grillner [Grillner *et. al.* (1991)]. The circuit is quite complicated, but the overall organization can be schematized and a simplified realization is possible. In Grillner's description each segment of the lamprey has its own CPG, which is locally connected via intersegmental connections to the neighboring ones and to the reticulospinal circuits that initiate locomotion. They involve a sequence of activity in consecutive spinal segments. In particular, the speed of motion depends on the phase lag between the consecutive activation of

adjacent segments. Each segment is itself a CPG since it is able to generate bursting activity in the local motor-neurons. In other words locomotion can be initiated at the level of each segment in response to certain sensory inputs. The overall biological systems is therefore a complex activator-inhibitor system, in which organized activity in the reticulospinal circuits directly generates motion patterns which drive the animal segments.

The swimming system as a whole thus represents a complex reaction-diffusion system, which has similar characteristics in a lot of biological cases, among which the examples presented before. Starting from the point that we consider the neuron dynamics as “always active”, we can model it as a nonlinear oscillator locally connected to other equals cells by diffusion rules. As a consequence the model can be once again described by using a RD-CNN as in Eq. (2.16). Each SCS is made up of a number of locally connected neurons. Therefore the whole scheme can be represented as in Fig. 5.1.

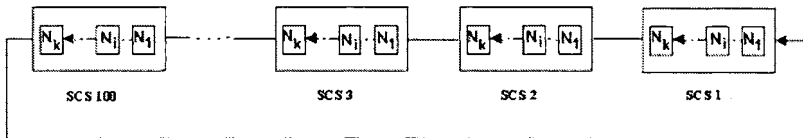


Fig. 5.1 A scheme for implementing the slow swimming CPG.

If suitable initial conditions are imposed, an autowave front will onset: this front will consecutively “visit” all or a fraction of the neurons in each SCS of the ring generating the signals needed to make the lamprey robot swim. In this condition an undulatory motion is caused along the lamprey body and modulated in its speed. In fact, following the connection shown in Fig. 5.1, since all the neurons are connected into the ring, the frequency of the wavefront is the least possible. If some neurons are cut from the ring faster undulatory motion can be easily realized. For instance Fig. 5.2 shows the same configuration, but now there are some particular neurons that also propagate their signals to the first neuron in the following and previous SCS. In this condition the autowave front will visit a fraction of all the neurons in the SCS; therefore the time the wavefront takes to get back to the first SCS will be shorter. Of course the fastest speed is achieved in Fig. 5.3, where only one neuron per SCS is able to fire. In such a way the

speed is increased simply by decreasing the neuron number involved in the progressing firing.

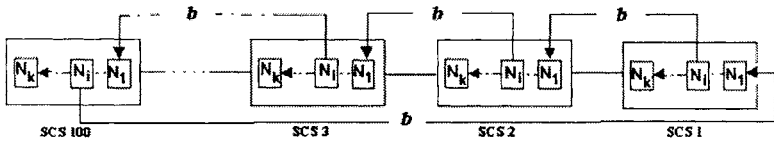


Fig. 5.2 A scheme for implementing the medium swimming scheme.

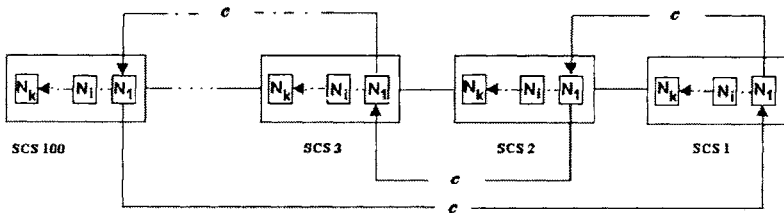


Fig. 5.3 A scheme for implementing the fast swimming scheme.

The lamprey robot, as already outlined, is made up of four actuated segments plus the head and the tail. The material chosen is aluminium. The structure of the robot, called Lampbot, is shown in Fig. 5.4.

It roughly consists of an aluminium backbone, made up of four vertebrae having elliptical shape of axes 16cm and 12cm long and 2 mm thick, and rods (each 20 cm long and 1 cm in diameter) inserted in the centre of each vertebra to realize the whole structure. Motion in each segment is realized by means of pneumatic valves which drive some pneumatic muscles. Each segment is able to perform horizontal motion, since two muscles work as a flexor-extensor couple. Each muscle is directly controlled by a pneumatic valve, whose driving signal comes from the state variable of a particular CNN motor-neuron, exhibiting oscillatory dynamics, coupled by a diffusion template in the ring already discussed. Each vertebra, which accommodates the valves, is also able to rotate, in order to allow spiral motions for downward and upward swimming, as shown in Fig. 5.5.

The rotation of each vertebra with respect to the rod is achieved via a

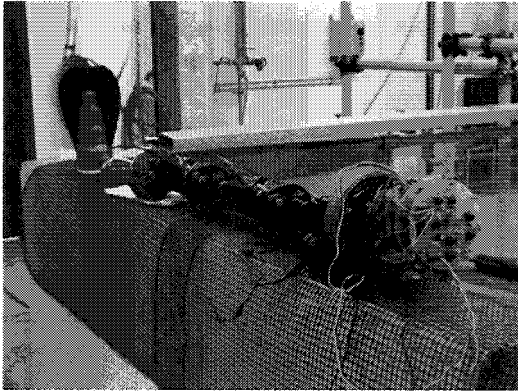


Fig. 5.4 The lamprey robot structure.

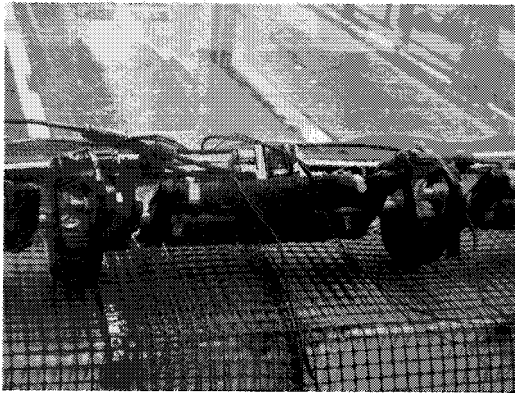


Fig. 5.5 Particular of a Lampbot vertebra: McKibben muscles are used for the actuation.

ball bearing. The rods are connected with each other via joints that allow axial movements. The head of the lamprey robot is built up of plastic pipes, suitably arranged so as to be waterproof. The RD-CNN for the swimming pattern generation is placed inside the head, together with other circuitry that drives the actuators according to the CNN state variables (Fig. 5.6). The lamprey tail is also made of aluminium although it may be made of plastic material so as to produce smoother swimming. The robot was covered with waterproof elastic material typically used by skin-divers. The

sheet was modelled, glued and stitched to allow axial and spiral movements due to its natural elasticity, but to prevent slipping along the vertebrae. Finally, a thin plastic film was placed between the mechanical structure and the artificial skin, to further isolate the structure from unavoidable perspiration.

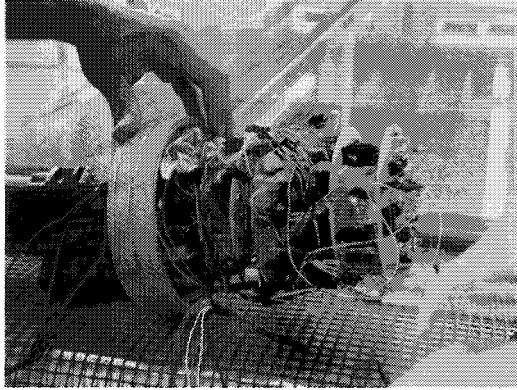


Fig. 5.6 A particular of Lampbot: the electronic circuits are placed inside the head of the robot.

Swimming was actuated by using a pneumatic approach, consisting of Air Muscles driven by electrical valves. In particular, we used McKibben pneumatic artificial air muscles (Fig. 5.5). They consist of an internal rubber tube supported by braided cords that are attached at both ends to realize a tendon-like structure. When the air goes inside the rubber tube, the high pressure pushes against the external shell, and tends to increase its volume. Given the non extensibility (or very high longitudinal stiffness) of the threads in the braided mesh shell, the actuator shortens in relation to its volume increase and produces tension to the applied mechanical load. This physical configuration causes McKibben muscles to have a variable-stiffness spring-like characteristics, nonlinear passive elasticity, physical flexibility, and very light weight compared to other kinds of artificial actuators. To realize our prototype 20 mm muscles were used; they are able to contract, under a pressure of 6 bars and a load of 5 Kg, by 34% of their length. Each muscle is driven by a digital pneumatic valve.

The lamprey-robot is autonomous as regard the electronics, while the power supply is provided from the outside. Figure 5.7 shows Lampbot while swimming in a pool.



Fig. 5.7 Lampbot while swimming in a pool.

## 5.2 MTA hexbot: a hexapod robot controlled by MTA-CNN

The MTA has been implemented on a discrete components board and used to control a six-legged walking robot called *MTA hexbot*. The structure of the MTA hexbot consists of a body made of aluminum and six legs, each one with 2 DOF, actuated by twelve servomotors (Hitec high torque serves HS-945MG with metal gears). A photo of the hexapod robot is shown in Fig. 5.8. The robot is completely autonomous: all the electronics and power supply (three packs of five Ni-Mh batteries 2300mAh, each pack containing five batteries, so that the voltage supply is  $6V = 5 \times 1.2V$ ) are on board. With this power supply the robot is able to carry a payload of 4kg and has an autonomy of about 1h without payload. Some characteristics of the hexapod robot are given in Table 5.1.

Table 5.1 Characteristics of MTA hexbot.

dimensions	31cmx25cmx20cm
weight of the robot	2250g
weight of servos	750g
weight of mechanical parts	600g
weight of batteries	500g
weight of electronic boards	400g
payload	4kg
autonomy	1h

The electronic part consists of 8 different circuits. A brief description of each board is given below (boards are labelled as in Fig. 5.8).

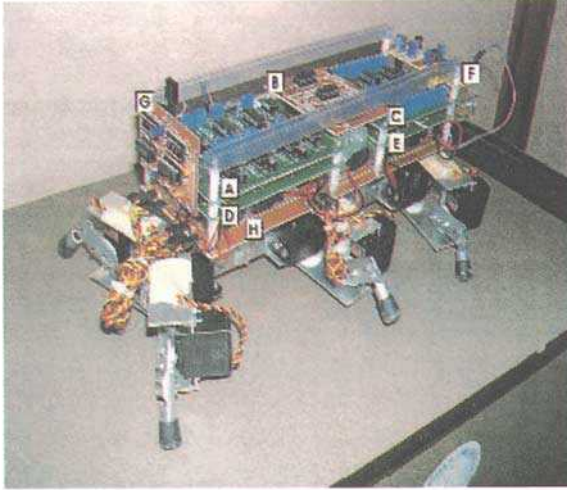


Fig. 5.8 The autonomous six-legged walking robot driven by the CNN-based MTA control system.

- A** - The CNN neurons. Each six second-order dynamical system is realized with OPAMPs. This circuit implements Eq. (2.5) according to the design presented in Appendix B.
- B** - Output saturation. For each leg of the robot the two joints ( $\alpha_{ij}$ -joint and  $\beta_{ij}$ -joint as in Fig. 2.7) are controlled by signals generated from the outputs of the motor-neuron  $ij$  associated with the leg as in Fig. 2.6. More precisely, the relationship between the motor-neuron outputs and the motor control signals can be described by the following equation:

$$\begin{cases} \alpha_{ij} = a_{2,ij} * y_{2,ij} + b_{2,ij} \\ \beta_{ij} = a_{1,ij} * f(y_{1,ij}) + b_{1,ij} \end{cases} \quad (5.1)$$

where  $\alpha_{ij}$  and  $\beta_{ij}$  are used to control the joint position and  $f(\cdot)$  is

the following:

$$f(y) = \begin{cases} y & \text{if } y > k \\ k & \text{if } y \leq k \end{cases} \quad (5.2)$$

where  $k$  can be regulated by a trimmer from -1 to 0. This further saturation of the control signal prolongs the duration of the stance phase (corresponding to the low control signal saturation value) and thus enhances the stability of the locomotion. Board B implements this nonlinearity  $f(\cdot)$ .

- C** - Motor settings. This circuit implements terms  $a_{1,ij}$ ,  $a_{2,ij}$ ,  $b_{1,ij}$ ,  $b_{2,ij}$  in Eq. (5.1) and thus provides the gains and offsets used to modify the output variables for proper control of the legs. The setup is made by using two groups of 12 trimmers (two for each DOF of each leg). These components set the parameters for suitable control of the leg motion.
- D** - Connections between CNN neurons (i.e. templates in Table 2.2) used to obtain fast, medium and slow gait. Analog switches (MAX394) driven by digital signals are used to choose the given set of templates.
- E** - Servomotor control. In order to generate the Pulse Width Modulated (PWM) signals needed to drive the servomotors, a CPLD (XC9572) is used. The CPLD with an external DAC (DAC800) generates a suitable digital ramp and provides the PWM signals by comparing this ramp with the analog signals  $\alpha_{ij}$  and  $\beta_{ij}$  coming from the motor-neuron outputs. In this way PWM signals (i.e. signal with duty cycle proportional to the analog control signals) are generated to drive the robot servomotors. VHDL language is used to program the XC9572 that generates the pulses and controls the maximum range of the duty cycle. It is worth remarking that the design of this board is strictly connected with the choice of the motors used to actuate the robot. The use of a microcontroller to generate PWM signals does not alter the generality of the analog approach. Moreover, a simple low-cost microcontroller can be used for this purpose since the effective computation takes place in the analog CNN.
- F** - Power supply. In order to obtain suitable dual power supply (+5V, -5V), two DC-DC converters (ST755, ST777) are used. The first is an inverter used to generate the negative voltage (-5V), while the second is a step-up converter. The positive power supply is



first decreased to  $4.8V$  by two diodes, and then set to  $+5V$  by the step-up converter. In this way, if the battery voltage decreases the power supply remains stable.

- G - Infrared remote control (4-bit encoder *M145026* on the remote command and decoder *M145027*) to pilot the robot. This allows to switch between the three different locomotion patterns and to change the motion direction, turning left or right.
- H - Printed circuit board for connections between batteries, servomotors and power supply.

The application of MTA-CNN to the hexapod robot offers the advantage of a simple circuitry and flexibility, thus allowing for the implementation of new locomotion patterns.

MTA hexbot was tested indoor on a flat surface with good performance: all three gaits work correctly. Other experiments were carried out outdoors on irregular terrain. The robot was able to work correctly on sloped surfaces in the presence of many small irregularities (such as small rocks). All three gaits work properly, as does direction control. The high ratio payload/weight for this type of legged autonomous robot is remarkable. Some pictures taken in outdoor experiments are shown in Fig. 5.9.

These tests were carried out controlling the direction of MTA hexbot by using the infrared remote control. Of course, the robot may be equipped with sensors making it able to interact with the environment. A simple application is to use the direction control to avoid obstacles detected via distance sensors endowed in the front part of the robot.



Fig. 5.9 Some pictures of MTA hexbot walking on irregular terrain.

### 5.3 MTA hexbot II: a remote-controlled hexapod robot

The mechanical structure of MTA hexbot II is similar to that of MTA hexbot I: the main difference is that each leg has three degrees of freedom in contrast with the two of the first prototype. Another hexapod robot prototype with three degrees of freedom for leg and its attitude control are discussed in Chap. 6.

Referring to Fig. 2.7 adding a further degree of freedom in each leg of the hexapod robot is equivalent to actuate the  $\gamma$ -joint. This joint can be controlled in two ways. A simple solution is just to use  $\gamma_{ij} = -\beta_{ij}$  for each leg of the robot so that the leg tibia remains perpendicular to the ground. This is the case of MTA hexbot II. The second solution is shown in Chap. 6 and consists of a new CPG design to generate an independent signal controlling the  $\gamma$ -joint.

The control system of MTA hexbot II was implemented by using the approach based on CNLab and described in details in Appendix A. Briefly, the control system is implemented on a PC communicating with the robot, while the circuitry carried on the robot only deals with the joint motor control (i.e. the generation of PWM signals), the measurement system and the modules for communicating with the PC. The aim of this approach is twofold: it allows us to develop a platform for testing new control strategies and to build a remote-controlled robot.

This robot allows us to test the direction control illustrated in Chap. 3. The hexapod robot is equipped with two “antennae” detecting possible obstacles. The outputs of these sensors are connected to the motor-neurons R2 and L2 by means of the values of bias  $i_2$ . The left sensor is connected to the motor-neuron controlling the right middle leg (R2), while the right sensor with L2. When one of these sensors, for instance the left sensor, detects an obstacle, the value of the bias  $i_2$  of the corresponding motor neuron (R2) changes from  $i_2 = -0.3$  to  $i_2 = -0.5$ . This parameter variation induces a bifurcation on the dynamical behavior of R2 which, from a limit cycle, collapses into a stable equilibrium point. This stops the oscillations of motor-neuron R2; thus the movement of the corresponding leg is inhibited and the robot turns right.

The two “antennae” detecting the presence of obstacles to be avoided are implemented by two pairs of contact switches. The switches are connected by a thin rubber pipe and the sensor output is the logical OR of the two switch outputs. This allows obstacles to be detected in a 180° orientation range. When one sensor is activated, the corresponding leg is stopped

for a fixed short time interval, allowing the robot to avoid the obstacle. Figure 5.10 shows the trajectory obtained while the robot is walking on a corridor (the wall on the right side and the wooden panels on the left constitute the obstacles). These experimental results confirm the suitability of the remote-controlled approach for real-time control.

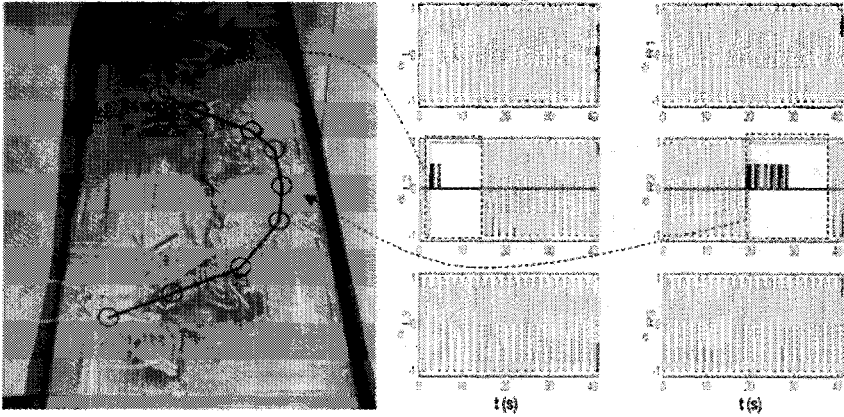


Fig. 5.10 Direction control of the robot: trajectory of the robot and control signals for the swing motor of each leg (for legs R2 and L2 the sensor outputs are also shown in the right panel).

Another interesting test that has been carried out using MTA hexbot II is experimental evaluation of the speed of the robot. Measuring the actual speed of a walking robot is a hard task, because of the nature of legged locomotion and since legged robots are subject to large accelerations. A simple strategy based on recordings from a video camera fixed on the ceiling of the laboratory and the processing of successive frames has the advantage of ease of implementation, but cannot be adopted for an autonomous robot. The strategy we propose is based on allowing the robot move on a patterned floor, and therefore does not require external measurement. On the other hand it cannot be applied to measure the walking speed on uneven terrain: this is still an open problem. The hexapod robot moves on a floor with alternating black and white stripes. The widths of black and white stripes are 5.6 cm and 7.1 cm, respectively. A diode/photoresistance pair is used to detect reflection of an infrared signal and to distinguish the color of the stripes. The sensor is located on the bottom of the robot, and an

average speed is measured. As the algorithm is based on transition between the two colors, it is quite robust. The algorithm was tested on the three different gaits implemented on the robot. Figure 5.11 shows the hexapod robot walking on the floor and the data acquired for the three locomotion patterns. Table 5.2 gives the speeds measured in the three cases.

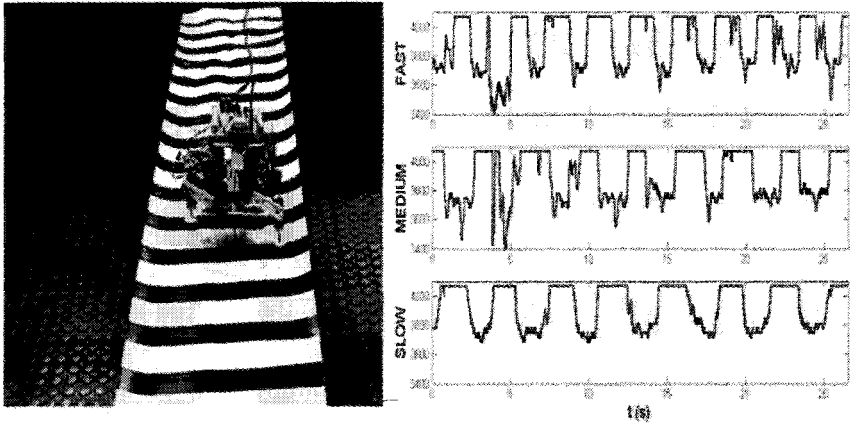


Fig. 5.11 Measured average speed for different locomotion gaits.

Table 5.2 Speed measured for the three gaits.

Fast gait	Medium gait	Slow gait
5.13 cm/s	4.18 cm/s	3.53 cm/s

#### 5.4 MTA hexbot III: a robot driven by the CNN-based CPG VLSI chip

The last example of bio-inspired robot is a hexapod robot controlled by the CNN-based CPG VLSI chip. A photograph of the robot, called MTA hexbot III, is shown in Fig. 5.12. An electronic board was designed to provide the control signals to the chip. For instance as regards the clock, a circuit included in the board is devoted to the generation of a square wave signal with an adjustable frequency. This circuit provides the chip with the clock and allows us to test the speed control discussed in Chap. 3.

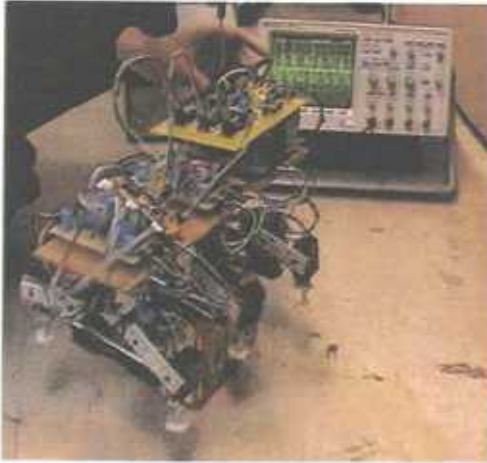


**Fig. 5.12** The electronic board containing the CNN-based CPG VLSI chip mounted on MTA hexbot III.

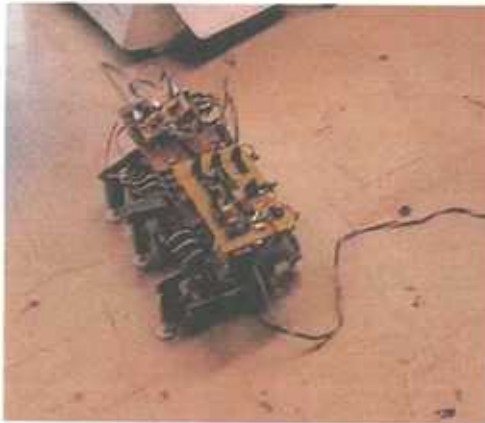
The outputs of the chip, namely the state variables of the CNN motor-neurons, are used to generate the PWM signals driving the servomotors. An electronic board is devoted to the generation of PWM signals. The experimental setup thus consists of only two electronic boards carried on the robot.

The first tests have shown the suitability of the chip, which works perfectly on MTA hexbot III. Fast gait, turning and speed control were successfully tested. For instance, as regards speed control, by tuning the frequency of the clock it is possible to change the speed of the robot. Moreover, this approach can be immediately extended to include feedback signals regulating the clock frequency in an automatic way. Figure 5.13 shows one of the tests carried out for speed control. The waveforms of the chip outputs are also visible: the frequency of their oscillations is regulated by the clock frequency tuned through a trimmer.

In Fig. 5.14 a photograph of MTA hexbot III while walking with fast gait is shown.



**Fig. 5.13** Experimental setup for speed control test. Waveforms of the CNN-based CPG VLSI chip are shown on the oscilloscope.



**Fig. 5.14** A photograph of MTA hexbot III walking with fast gait.

## Chapter 6

# High-level analog control: attitude control and Motor Maps

Even if animals deprived of commands from high-level neural centres are still able to locomote, in many cases posture control is lost. This is clear evidence of the fact that posture control is a complex form of behavior involving high-level control and feedback from the visual system, vestibular system, and so on. In the motor system the role played by high-level control centers, also called command neurons, can be very complex. In this and in the next Chapter we will give some examples of bio-inspired high-level control. In particular, this Chapter deals with the important issue of attitude control for the hexapod robot approached with two different strategies. The first one is based on distributed control through CNNs; the second, based on Motor Maps, introduces a more general framework for the control of nonlinear systems in which self-organization and unsupervised learning play a fundamental role in realizing an adaptive controller inspired by the motor cortex of the human brain.

### 6.1 CNN-based attitude control

This Section focuses on a biologically inspired analog control system to solve the task of attitude stabilization and locomotion control in a hexapod robot. Attitude control of a rigid body in three-dimensional space consists of leading the object to a desired orientation with respect to a fixed reference frame by computing a suitable control law (for a survey on the attitude control problem see [Kreutz-Delgado and Wen (1991)]). Of course this issue is fundamental in hexapod robots when walking on uneven terrain.

In [Uchida *et. al.* (2000)] attitude control in a hexapod robot is addressed by using both feedforward and feedback controllers, but it is based on a specific locomotion pattern in which the robot is always supported by

five legs. In [Barnes (1998)] bio-inspired hexapod robot control is achieved using six controlled coupled nonlinear oscillators obtaining a dynamically variable gait. However, the overall control is implemented on a digital controller. This strategy is typical of general control strategies that are developed at a very high-level, completely discarding implementation issues. This approach may be convenient for controlling systems with a relatively low number of controlled joints, but it loses its efficiency when this number grows. The strategy introduced here focuses on the attitude control problem from a space distributed point of view looking directly at systems with several joints distributed in space. Moreover, the control problem is formulated considering above all the hardware realization, through CNN structures. Therefore, CNNs are considered here as an emerging paradigm both for the generation and the control of motion in space-distributed nonlinear systems. Attitude control is implemented by using distributed control via CNNs.

The need for posture control led to an upgraded structure of the robot. The robot, called Rexbot II, has 3 DOF for each leg and has a structure similar to that of MTA hexbot II and III (see Chap. 5 and Fig. 6.1). A further CNN structure, able to control the robot attitude in real time during any type of locomotion, was designed.

Considering an added degree of freedom for each leg is a mandatory step for the attitude control problem and offers further advantages. In fact, the structure strictly mimics the biological case and therefore a number of different locomotion patterns shown by insects, crustaceans and other sophisticated moving creatures [Orlovsky *et. al.* (1999); Waldron and Song (1989)] can be implemented. The added degree of freedom for each leg enables us to consider a more realistic and efficient trajectory in the stance phase. On the other hand, the weight of the whole system grows, mainly due to the presence of six further servomotors actuating the third degree of freedom for each leg.

Rexbot II has a carrying structure in aluminium and servomotors as actuators. The structure of each leg has three rotational joints orthogonally displaced as shown in Fig. 6.1(a). Figure 6.1(b) sketches the structure of the whole robot, together with the reference system, rigidly connected to the body, in which the *Euler angles* [Siciliano and Sciavicco (1989)] used for posture control are highlighted. The mechanical structure, similar to that used for MTA hexbot, was designed to be totally autonomous.

For the attitude control task a 2-axis acceleration sensor, ADXL202,



was used to measure the inclination of the robot. This low-cost sensor integrates two analog measurements on one chip. The two signals revealing the orientation of the robot with respect to the Earth plane are used as error signals, detecting the error on the roll ( $\theta$ ) and pitch ( $\varphi$ ) angles (Fig. 6.1(b)). The sensor is located on the centre of the underside of the aluminium carrying structure.

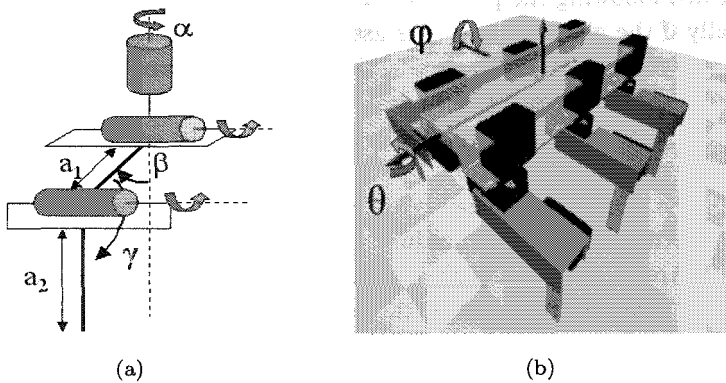


Fig. 6.1 (a) Leg model (three actuated joints). (b) Structure of the robot Rexbot II.

The bio-inspired control system of Rexbot II consists of a CNN-based CPG and a distributed network for posture stabilization. These two structures are described in the following Sections.

### 6.1.1 The CNN for gait control

In this Section it is shown how the reformulation of the problem in terms of a legged structure with 3 DOF for leg requires a new design of the CPG for the locomotion generation task. Unlike in the robots shown in Chap. 5 the third joint is in fact actuated by an independent signal that has to be generated by the leg controller, that now consists of two motor-neurons (2.5). The design of this cell is addressed by following the guidelines discussed in Chap. 2.

The configuration assumed for the leg is that of the well-known anthropomorphic manipulator [Siciliano and Sciavicco (1989)]. This allows us to find the trajectories of the joint variables for a reference trajectory of the leg tip in the operating space by using closed formulae for the inversion of the kinematics matrix for this manipulator.

The leg controller is a CNN generating the joint variables and then driving the servomotors using a Pulse Width Modulation scheme. This CNN-based CPG is designed to drive the leg joints so as to follow a given leg tip trajectory in the operating space.

In other words, the actual problem was, given a leg trajectory suitable for correct movement of the whole hexapod, to design a network of CNN neurons able to generate the signals for the given trajectory. This is accomplished following the principles of Chap. 2. Clearly, this is a hard task, especially if the given trajectory is assumed to be arbitrary.

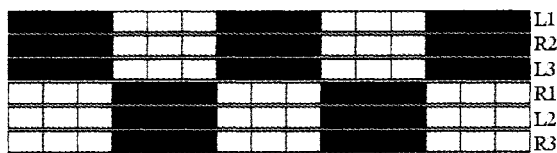
The strategy adopted was to consider only a few constraints on the leg trajectory and to design a CNN-based leg controller whose outputs can generate a trajectory satisfying these constraints. In our case the constraints refer mainly to the stance phase, while some arbitrary assumptions can be made on the swing phase, in which the dynamic constraints are less strong. A trial and error procedure led to a leg controller with two CNN neurons [Arena *et. al.* (2002b)]. Each CNN neuron is described by Eq. (2.5). The parameter values of the first CNN neuron are those given in Table 2.1, while those of the second are:

$$\mu = 0.5; s = 1; i_1 = i_2 = 0.3 \quad (6.1)$$

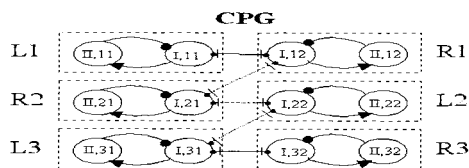
Obviously, both sets of parameters are such that system (2.5) is characterized by a globally stable limit cycle with a slow-fast dynamics, usually encountered in biological neurons [Shepherd (1997)]. The first neuron excites and is inhibited by the second neuron.  $y_{1,I}$  and  $y_{2,I}$  being the outputs of the first neuron and  $y_{1,II}$  and  $y_{2,II}$  those of the second, the inhibitory synapse is realized by adding the output  $y_{1,II}$  to the second equation of (2.5) for the first neuron, while the excitatory one is accomplished by adding the output  $y_{2,I}$  to the first equation of the second neuron. Both synaptic weights are  $\varepsilon = 1$ . The leg controller is indicated in Fig. 6.2 with a dashed box. Each leg controller generates four outputs, from which the signals to drive the joints are drawn using the following relations:

$$\begin{cases} q_1 = y_{2,I}/2 \\ q_2 = y_{1,II} + 0.5 \\ q_3 = -y_{1,I}/5 - 0.9 \end{cases} \quad (6.2)$$

where  $q_1$ ,  $q_2$  and  $q_3$  are the driving signals for the  $\alpha$ -joint,  $\beta$ -joint and  $\gamma$ -joint, respectively (Fig. 6.1(a)).



(a)



(b)

Fig. 6.2 The CPG for the alternating tripod gait: (a) scheme of the pattern (black areas indicate the stance phase, white areas indicate the swing phase); (b) the structure of the CNN-based CPG (the arrows indicate excitation, the dots indicate inhibition).

Interaction among leg controllers is achieved by creating synaptic connections between each pair of neurons controlling each leg. These synapses guarantee the generation of a given locomotion pattern. Thus, they depend on the desired pattern. Among the possible locomotion patterns the alternating tripod gait was implemented. The neurons of the leg controllers are therefore connected into pairs of mutually inhibited neurons. As can be noticed in Fig. 6.2, these connections are established only between the “first” neurons of each leg controller cell. Each leg controller now includes a fourth-order circuit able to drive the joints of the leg and is locally connected to the neighbors in the classical CNN paradigm. This is clearly a generalization of the approach presented in Chap. 2 and therefore can be applied to implement other gaits. In Fig. 6.2 the leg controllers are indicated with dashed boxes together with the leg that they control.

### 6.1.2 The attitude control CNN

The attitude control task has to guarantee that the robot keeps its body in a horizontal position when it is walking on sloping planes. Clearly, this means a strong increase in the stability of the robot during locomotion. The solution proposed to solve this task is to use traditional controllers such as PIDs. The current position of the body is revealed by using the

accelerometer sensor. This device provides two signals indicating the roll and pitch rotations of the body. These signals are filtered and integrated and then used as measures of the robot attitude position error. A PID for each leg is then used to generate signals to correct the attitude error. These signals, merged to those from the CPG, are used to control the joint variables.

It has been found that for the purpose of stability control only a proportional-integrative action is needed. This solution, as discussed below, can be formulated as a CNN, whose inputs are the sensor signals and whose outputs, together with those from the CNN-based CPG, constitute the joint variables. This aspect makes the approach particularly appealing, since the CNN paradigm can be used to generate and control the whole locomotion phase. Moreover, although the control strategy has been designed to work in the linear range, the use of piece-wise linear nonlinearities, that come with the CNN implementation, has a number of advantages, including the avoidance of motor current overloads and the rise of emergent behavior as discussed below.

The 3 DOF leg allows us to achieve attitude control by applying very intuitive principles. Each leg will make different use of the information from the sensor to stabilize the posture of the robot. The pattern of the control actions required in the event of given stimuli can be found with the following guidelines.

If the robot has to go up a slope, maintaining a horizontal attitude, the front legs have to be lowered, while the hind legs have to be raised; the middle legs can be maintained at a constant distance from the ground. These results can be achieved by suitably controlling the  $\beta$ -joints and  $\gamma$ -joints of the front and hind legs. For the  $\beta$ -joints of the front legs the angle  $\beta$  between the femur (link  $a_1$ ) and the vertical line is increased, while the angle  $\gamma$  of the  $\gamma$ -joint between the femur and the tibia (link  $a_2$ ) is decreased by the same quantity. For the hind legs the opposite holds: the angle  $\beta$  is decreased and the angle  $\gamma$  increased. No action is performed on the middle legs. This pattern of action can be summarized in a matrix  $P_\beta$ :

$$P_\beta = \begin{bmatrix} -1 & -1 \\ 0 & 0 \\ +1 & +1 \end{bmatrix} \quad (6.3)$$

in which the signs of the action for each leg ('+' increase the height, '-' decrease the height) are indicated. The legs are displaced as in Fig. 6.2. The effect of the pitch correction is described in Fig. 6.3(a) with the help

of a simplified robot model emphasizing the angles of correction, while in Fig. 6.3(b) a photo of the robot when the control is acting is shown. It can be seen that the body of the robot is kept horizontal against the slope of the ground plane.

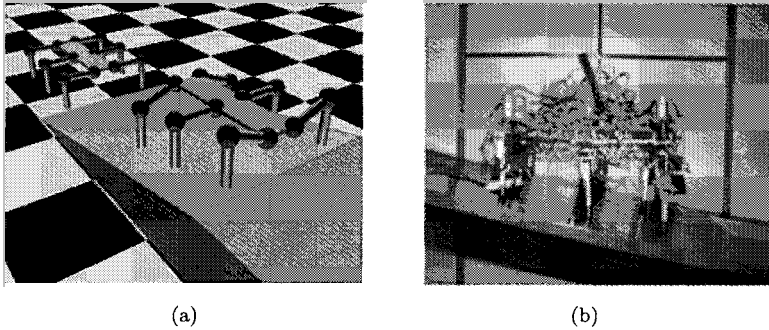


Fig. 6.3 Attitude control of Rexabot II. (a) Ideal Pitch. (b) Actual Pitch.

Equivalently, for roll control, the action has to be performed in opposite directions for the contralateral legs, thus resulting in the pattern expressed by the matrix  $R_\beta$ :

$$R_\beta = \begin{bmatrix} +1 & -1 \\ -1 & +1 \\ +1 & -1 \end{bmatrix} \quad (6.4)$$

Notice that the middle terms of  $R_\beta$  reflect the scheme shown in Fig. 6.2. In Fig. 6.4 both the robot prototype model and a photo of the robot are shown when the roll control is performed.

In Fig. 6.5 the attitude control system is illustrated. The signals  $\varphi$  and  $\theta$  indicate pitch and roll angular velocities measured by the accelerometer sensor, respectively. By applying the integrative action the position error signals  $e_{ij}$  can be obtained according to the following formula:

$$\dot{e}_{ij} = -\tau e_{ij} + (k_p p_{ij} \varphi + k_r r_{ij} \theta) \quad (6.5)$$

The error signals  $e_{ij}$  are added to the signal  $q_2$  from the CNN-based CPG and used to drive the  $\beta$ -joint, and subtracted from the signal  $q_3$  driving the  $\gamma$ -joint, due to the fact that opposite actions have to be performed on the femur-tibia ( $\gamma$ -joint) and femur-coxa joints ( $\beta$ -joint).  $\tau$  is the pole of the integrative action.

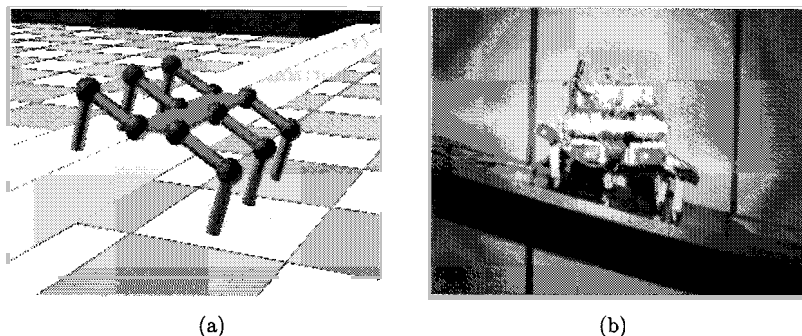


Fig. 6.4 Attitude control of Rexabot II. (a) Ideal Roll. (b) Actual Roll

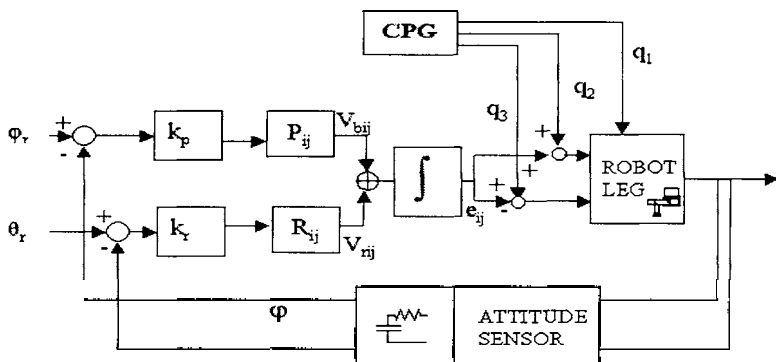


Fig. 6.5 Scheme of the attitude controller for each leg

In this way an analog distributed system for both locomotion pattern generation and attitude control is used. In particular, the approach used for hexapod attitude control makes use of a network of distributed proportional integral linear controllers, one for each leg. The control signal for each leg depends on global information on the robot attitude, with respect to a virtual horizontal plane. This is in agreement with the biological case, in which central feedback signals affect local control. Of course, in biology a fundamental role is also taken by local stimuli, which in our case are still under investigation.

The hexapod locomotion pattern was generated by a CNN-based CPG, and the analog distributed attitude control could also be formulated in terms of CNN structures. If a saturation block is inserted in cascade with the integrator in Fig. 6.7, Eq. (6.5), written for each of the six legs, can be viewed in terms of a space-invariant CNN with the following templates:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 - \tau & 0 \\ 0 & 0 & 0 \end{bmatrix}; B = \begin{bmatrix} 0 & -1/2 & 0 \\ 1 & -1 & -1 \\ 1 & -1/2 & 0 \end{bmatrix}; U = \begin{bmatrix} -k_r\theta & k_p\varphi \\ 0 & 0 \\ -k_r\theta & -k_p\varphi \end{bmatrix}$$

where  $U$  is the input pattern that depends on the roll and pitch angles.

As the terms of the feedback template  $A$  are all zero with the exception of the central term, the stability study is straightforward. In fact, when the dynamics lies in the linear region, each cell is characterized by a negative eigenvalue ( $\lambda_{ij} = -\tau$ ); when the state of the cell enters the saturation region  $\lambda_{ij} = -1$  holds. In these conditions the stability of the control law is assured. The CNN formulation is particularly suitable for practical reasons: output saturation corresponds to motor saturation, which cannot be neglected in the control loop.

Therefore, the whole motion generator and control system for the hexapod robot is provided by CNNs.

It should be remarked that the nonlinear control scheme represented by the CNN includes a saturation block in the control loop, preventing the motors from reaching the joint limits due to the mechanical configuration of the hexapod robot. This prevents the motors from entering the current saturation region. However, the control system discussed cannot prevent this from occurring in other cases, for example when unforeseen situations cause overloading in one leg. In the following Section a result which emerged in experimental tests of the prototype is discussed.

### 6.1.3 Experimental tests

The whole control system was implemented on Rexbot II by considering a simplified version of the CPG illustrated in Fig. 6.2. To build the CPG for the alternating tripod only one pair of leg controller cells was considered. Each one is connected to three legs, thus the first one is the controller of the first tripod (legs L1, R2 and L3) and the second one is the controller of the other tripod (legs R1, L2 and R3). These cells will be referred to using the subscript indexes 11 and 12, respectively. This pair of mutually

inhibited fourth-order cells, with a suitable choice of synaptic weights, is characterized by anti-phase synchronization of the activities of the two cells.

The design of the leg controller circuit follows the considerations discussed in Appendix B. The two cells controlling the two tripods are connected by two inhibitory synapses using inverting amplifiers. The weights of these synapses are  $\varepsilon = -0.6$ . The outputs of the CPG circuit are added to the signals from the attitude controller and then converted into Pulse Width Modulated signals and used to drive the servomotors of the leg joints. The circuitry for the attitude control was designed in a similar fashion. The whole control system was implemented on a discrete-component board using TL084 operational amplifiers.

The stability of the whole control approach was investigated by a huge number of experiments performed on the prototype built in our laboratory. The tests performed showed the suitability of the methodology: the attitude control takes place in real time, assuring dynamical stabilization of the whole structure when walking over rough terrain. It is also surprising to note how efficiently the structure can escape from situations in which some points, typically the  $\gamma$ -joints, undergo unforeseen conditions of saturation. Since these conditions come together with perturbations in the attitude, the feedback signal and the analog structure work efficiently to quickly recover normal conditions.

One of the experimental tests revealing the capabilities of the robot to escape from a situation in which the load capability of a motor is exceeded is presented in Fig. 6.6, which shows some frames from a video taken in our laboratory. In frame 039 the  $\gamma$ -joints of the left legs are saturated: this results in unnatural locomotion by the hexapod. This situation is characterized by a non-zero value of the attitude error signals. Thus, this situation is not stable. Self-organization makes the hexapod able to escape from such a condition. In particular, as can be seen in frame 056, the roll body orientation situation is reversed; in fact, this situation is also unstable. These oscillations of the roll orientation, together with the action of gait generation, allow the robot to avoid a static stalling situation, by continuously perturbing the whole system. In frame 118 the robot achieves its natural attitude. This is an emerging property due to both attitude control and gait generation.

Obviously, the attitude control discussed in this Chapter can be applied also on the prototypes (MTA hexbot II and III) previously shown.



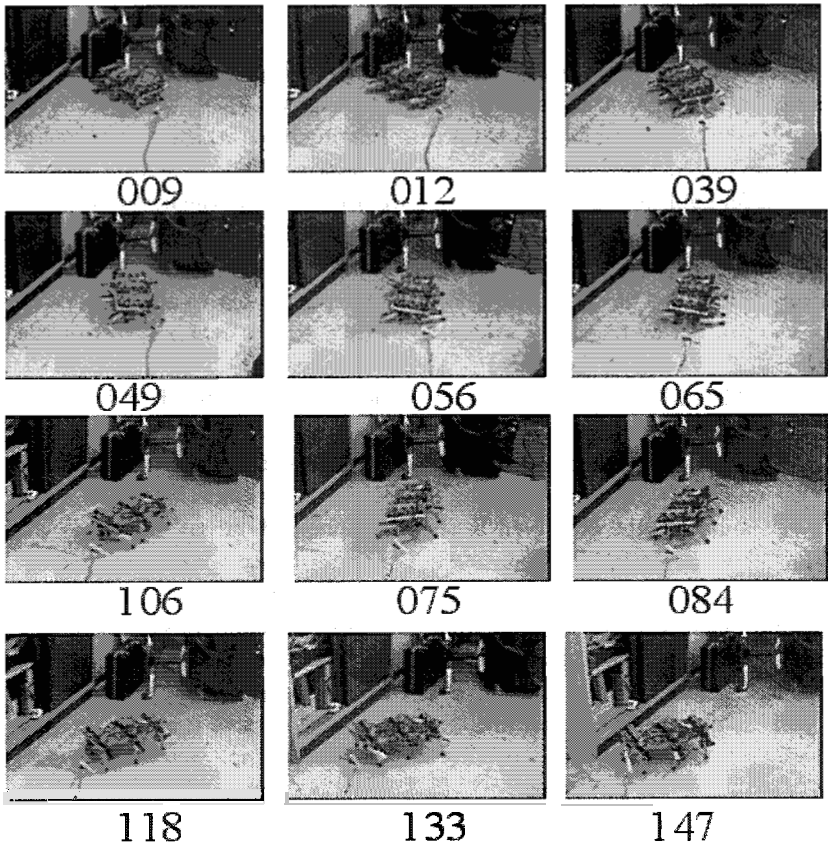


Fig. 6.6 Video frames showing robot capability for recovery from a situation in which the  $\gamma$ -joints of the left legs undergo overloading current saturation.

## 6.2 Motor Maps and attitude control

### 6.2.1 Motor Maps

The importance of topology-preserving maps in the brain relies on both the representation of sensory input signals and the ability to perform an action in response to a given stimulus. Indeed, neurons in the brain are organized in local assemblies able to perform a given task such as sending appropriate signals to muscles. These neural assemblies constitute two-dimensional layers in which the locations of the excitation are mapped into

movements.

Topology-preserving structures able to classify input signals inspired the paradigm of Kohonen's Neural Networks [Kohonen (1972)]. These artificial neural networks formalize the self-organizing process in which a topographic map is created. Neighboring neurons are thus excited by similar inputs. Successful applications of these maps have been found in the field of pattern recognition, clustering and so on [Kohonen (1989)].

An extension of these neural structures is represented by Motor Maps [Ritter *et. al.* (1992)]. These are networks able to react to localized excitation by triggering a movement (like are the *motor cortex* or the *superior colliculus* in the brain).

To this end Motor Maps, unlike Kohonen's networks, should include the storage of an output specific to each neuron site. This is achieved by considering two layers: one devoted to the storage of input weights and the other one for the output weights. The plastic characteristics of the input layer should also be preserved in the assignment of output values, so the learning phase deals with updating both the input and the output weights. This allows the map to perform tasks such as motor control. These considerations led to the idea of using Motor Maps as an adaptive self-organizing controller for nonlinear systems. The main difficulties of this approach are due to the fact that the learning phase may be long and especially in the transient phase the random initial values of the control law may be very inefficient. Moreover, the feasibility of a hardware implementation is seriously conditioned by the size of the map. It is possible to overcome these difficulties by adopting a control scheme in which the Motor Map acts at a higher level: for instance, one can consider inner feedback loops to stabilize the system and thus reduce the number of neurons needed by the Motor Maps as well as the effects of the initial learning phase.

With this idea, Motor Maps can constitute an adaptive controller: they have been successfully applied to control chaotic circuits [Arena *et. al.* (2002a)]. In this Chapter, after a brief review of this result, the scheme is applied to control the attitude of the hexapod robot.

Formally, a Motor Map can be defined as an array of neurons mapping the space  $V$  of the input patterns onto the space  $U$  of the output actions:

$$\Phi : V \rightarrow U \quad (6.6)$$

The learning algorithm is the key to obtain a spatial arrangement of both the input and output weight values of the map. This is achieved by considering an extension of the winner-take-all algorithm. At each learning

step, when a pattern is given as input, the winner neuron is identified: this is the neuron that best matches the input pattern. Then, a neighborhood of the winner neuron is considered and an updating involving both the input and output weights for neurons belonging to this neighborhood is performed. Both supervised and unsupervised learning can be applied. However, in order to investigate an autonomous self-organizing system for nonlinear control, unsupervised learning was considered. In this case there is no *a priori* information on the appropriate control action and no teacher is available. The algorithm has to find the correct control action by itself. The only source of information is provided by the so-called *Reward Function*, introduced below, which indicates how well the control is being performed.

In this case (unsupervised learning), weight updating only takes place if the corresponding control action leads to an improvement in the controlled system; otherwise, the neuron weights are not updated. In this framework a fundamental role is taken by the Reward Function. The definition of this function is perhaps the most crucial point in the whole network design. Since the target signal is not directly available, some information is required on the effect of a given control action on the system being controlled. This information has to be formalized in the form of a function of the controlled variable, the Reward Function. This has to be defined so that decreasing values lead to better control performance.

More precisely, the unsupervised learning algorithm of the Motor Map can be described in the following steps.

*Step 1.* In the first step the topology of the network is established. The number of neurons is chosen and the Reward Function is established. The number of neurons needed for a given task is chosen by a trial and error strategy, thus once numerical results indicate that the number of neurons is too low, one must return to this step and modify the dimensions of the map. At this step the weights of the map are randomly fixed.

*Step 2.* An input pattern is presented and the neuron whose input weight best matches the input pattern is established as the winner. Therefore, to establish the winner neuron, the distance between the neuron input weight and the input pattern is computed for each neuron, considering the absolute value of the difference between these two vectors.

*Step 3.* Once the winner neuron has been chosen, its output weight is used to perform the control action  $f(t)$ . This is not used directly, but a random variable is added to the value to guarantee a random search for

possible solutions, as follows:

$$f(t) = w_{winner,out} + a_s \lambda \quad (6.7)$$

where  $w_{winner,out}$  is the output weight of the winner neuron,  $a_s$  is a parameter determining the mean value of the search step for the neuron  $s$ , and  $\lambda$  is a Gaussian random variable with a zero mean. Then the increase  $\Delta R$  in the Reward Function is computed and, if this value exceeds the average increase  $b_s$  gained at the neuron  $s$ , the next step (updating of the neuron weights) is performed; otherwise this step is skipped. The mean increase in the Reward Function is updated as follows:

$$b_s^{new} = b_s^{old} + \rho(\Delta R - b_s^{old}) \quad (6.8)$$

where  $\rho$  is a positive value. Moreover,  $a_s$  is decreased as more experience is gained (this holds for the winner neuron and for the neighboring neurons), according to the following rule:

$$a_i^{new} = a_i^{old} + \eta_a \xi_a (a - a_i^{old}) \quad (6.9)$$

where  $i$  indicates the generic neuron to be updated (the winner and its neighbors),  $a$  is a threshold the search step should converge to, and  $\eta_a$  is the learning rate, while  $\xi_a$  takes into account the fact that the parameters of the neurons to be updated are varied by different amounts, defining the extent and the shape of the neighborhood.

*Step 4.* If  $\Delta R > b_s$ , the weights of the winner neuron and those of its neighbors are updated following the rule:

$$\begin{aligned} w_{i,in}(t+1) &= w_{i,in}(t) + \eta \xi (v(t) - w_{i,in}(t)) \\ w_{i,out}(t+1) &= w_{i,out}(t) + \eta \xi (f(t) - w_{i,out}(t)) \end{aligned} \quad (6.10)$$

where  $\eta$  is the learning rate,  $\xi$ ,  $v$ ,  $w_{in}$  and  $w_{out}$  are the neighborhood function, the input pattern, the input weights and the output weights, respectively, and the index  $i$  takes into account the neighborhood of the winner neuron. In *supervised learning*  $f(t)$  is the target, while in *unsupervised learning* it is varied, as discussed above.

*Step 5.* Steps 2-4 are repeated. If one wishes to preserve a residual plasticity for a later readaptation, by choosing  $a \neq 0$  in step 3, the learning is always active and so steps 2-4 are always repeated. Otherwise, by setting  $a = 0$ , the learning phase stops when the weights converge.

### 6.2.2 Motor Maps for Chaos Control

A brief example of the suitability of Motor Maps to control nonlinear systems is given below, referring to the nonlinear control of a chaotic system.

The advantage of using Motor Maps for chaotic system control lies in the unsupervised learning of a nonlinear time-varying feedback gain, leading to a control architecture suitable for hardware implementation in an analog circuit. In this Section the scheme of the Motor Map Controller (MMC) is described with application to the control of chaotic systems. In particular, the example given deals with the well-known Chua's circuit.

For chaos control, a feedback entrainment scheme is adopted. The purpose of this control is to entrain a nonlinear system to a given trajectory. The system to be controlled is chaotic and the reference system is the same as the controlled one, but with a different parameter set, in order for example to generate a periodic orbit, i.e. a cycle limit. However, other reference trajectories (even not properly entrainment trajectories) can be considered. The whole control scheme is illustrated in Fig. 6.7. The MMC acts on the system to be controlled by setting the value of the feedback gain. This time-varying adaptive gain, represented by the value of the output weight of the winner neuron of the Motor Map, modulates the error signal between the reference trajectory and the actual one and is fed back into the system to be controlled.

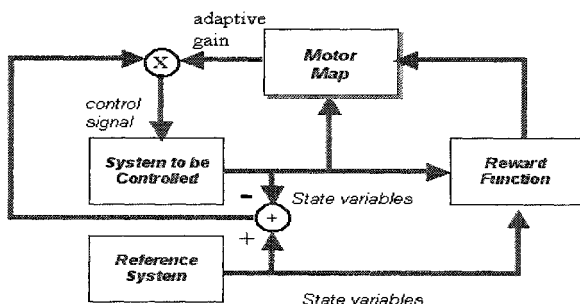


Fig. 6.7 Scheme of the Motor Map based controller. The Motor Map controls the time-varying adaptive gain multiplying the error signal exclusively on the basis of the information of the Reward Function that is established taking into account the difference between reference system and the controlled system.

Instead of controlling the whole state space, only two variables have

been considered in the feedback. So they constitute the inputs of the MMC. Of course, the possibility of achieving good performance with this reduced state space control is not a general conclusion, but it holds for this specific chaotic attractor. However the Motor Map based control scheme is a general one and can include feedback from the whole state space.

The equations of the Chua's circuit [Madan (1993)] are the following:

$$\begin{aligned}
 \dot{x} &= \alpha(y - h(x)) \\
 \dot{y} &= x - y + z \\
 \dot{z} &= -\beta y \\
 h(x) &= m_1 x + 0.5(m_0 - m_1)(|x + 1| - |x - 1|)
 \end{aligned} \tag{6.11}$$

With the feedback the model becomes:

$$\begin{aligned}
 \dot{x} &= \alpha(y - h(x)) + k_x(x_{ref} - x) \\
 \dot{y} &= x - y + z + k_y(y_{ref} - y) \\
 \dot{z} &= -\beta y + k_z(z_{ref} - z) \\
 h(x) &= m_1 x + 0.5(m_0 - m_1)(|x + 1| - |x - 1|)
 \end{aligned} \tag{6.12}$$

where  $x_{ref}$ ,  $y_{ref}$  and  $z_{ref}$  are the state variables of the reference system. As previously outlined, only two variables have been fed back, i.e. in our simulations  $k_z$  was set to zero.

The Reward Function was chosen as the square of the Euclidean distance between the two controlled state variables and the corresponding reference ones as follows:

$$R = -((x - x_{ref})^2 + (y - y_{ref})^2) \tag{6.13}$$

Two schemes were used to perform the control. In the first one a generic case is assumed: a Motor Map is built for each variable to be controlled. In the second scheme a single Motor Map acts on the system controlling the variable having the larger error. In the former scheme, the two Motor Maps act independently on the two state variables  $x$  and  $y$ , storing (in the input layer) the values of the corresponding input pattern and the values of the corresponding adaptive feedback gain  $k_x$  for the first map or  $k_y$  for the second map (in the output layer). In the second scheme a single Motor Map is devoted to controlling both variables, at each step controlling the one having the largest error.

In both cases the winner-take-all law establishes which neuron has the most suitable gain value. Moreover, only the weights of the winner neuron are updated. This assumption leads to a simpler hardware implementation than is the case when a neighborhood of the winner neuron is also updated.

Moreover, this assumption is in some sense justified by the low number of neurons needed for the task.

In the Motor Map implemented the *learning rate* was kept constant. In fact, while in supervised learning schemes the learning rates,  $\eta$  in Eqs. (6.10), can be gradually decreased to stop the learning phase, in unsupervised learning schemes the learning rate is usually kept constant. Keeping the learning rate constant enables continuous learning for the Motor Map and makes it able to deal with changes in the reference trajectory. On the other hand, this can lead to instabilities in the value of the feedback gain. In fact, simulations carried out reveal ever-increasing values for the feedback gain  $k$ . To eliminate this drawback an error performance goal is introduced in the Motor Map: that is, when the error is below a given bound, the learning is stopped. This avoids dependence on the time of the learning rate  $\eta(t)$  and also provides a mechanism to re-start the learning when the error exceeds the given bounds. Moreover, the achievement of this goal constitutes a performance parameter for the Motor Map.

To reduce hardware requirements, by exploiting the noise-like behavior and the broad-band spectrum of a chaotic attractor, it can be assumed that the *stochastic term*  $\lambda$  is implemented by a chaotic signal from an analog circuit. In particular, a zero-mean variable can be chosen, for example variable  $y$  of another Chua's circuit. In order for the hypothesis of broad-band spectrum to make sense, this *chaos generator* Chua's circuit should have time constants smaller than those of the one being controlled. Simulation results emphasize the role played by the variance of this signal. The settling time depends on this variance: small increases in the variance, by multiplying this signal by a gain factor, lead to smaller settling time values. This consideration can be useful to decrease the settling time, considering parameter  $a_s$  in Eq. (6.7) as a function of the actual error, and providing a high value when the error is large and a low value when the error is small. Thus  $a_s$  is not updated as in (6.9), but is a function of the actual error.

As an example, the results of control of a Chua's circuit using the scheme with two Motor Maps, here called the 2 Motor Maps Controller (2MMC), are illustrated. The controlled system is the double scroll Chua attractor in Eqs. (6.12) with the following parameter values:

$$\alpha = 9; \beta = 14.286; m_0 = -\frac{1}{7}; m_1 = \frac{2}{7} \quad (6.14)$$

As discussed above, an Error Bound (EB) on the steady state error between the reference and controlled systems for each of the variables being controlled was considered as a performance specification to establish the end

of the learning phase.

A limit cycle was chosen as the entrainment trajectory for system (6.12). This limit cycle is generated by choosing the parameters of system (6.11) as follows:

$$\alpha = 7; \beta = 12; m_0 = -\frac{1}{7}; m_1 = \frac{2}{7} \quad (6.15)$$

Two maps, each having four neurons, were trained for 390 epochs to achieve the performance specifications  $EB_x = 0.08$  (error bound for the variable  $x$ ) and  $EB_y = 0.04$  (error bound for the variable  $y$ ). Figure 6.8(a) shows the trends for the variable  $x$  of the controlled system and the variable  $x_{ref}$  of the reference system, while the phase planes  $x - y$  for both systems are shown in Fig. 6.8(b). Figure 6.8(c) shows the error and the EB for the variable  $x$ .

From these results it is clear that the Motor Map efficiently learns the control action to be performed on the system being controlled.

Many other examples emphasizing the capabilities of the Motor Map to learn by itself the control action by itself even when the reference attractor is switched to another one are given in [Arena *et. al.* (2002a)].

The performance of the scheme based on two Motor Maps (each acting on a state variable) and that of the scheme based on a single Motor Map (acting on the variable having the largest error) are comparable. In the second case the neurons cluster to deal with the two state variables, but a longer learning phase is required.

The approach is suitable for hardware implementation, since the simulation results indicate that a low number of neurons is needed to perform chaos control [Arena *et. al.* (2002a)]. Moreover, the experimental results agree with simulations, confirming that the application of Motor Maps to chaotic system control give very efficient results. The Motor Map learns the adaptive control law in a unsupervised manner without any information on the structure of the controlled system.

### 6.2.3 Motor Maps for attitude control

Motor Maps provide suitable bio-inspired control for attitude stabilization. They can be applied to stabilize the posture of the hexapod robot posture according to the principles underlying attitude control introduced in Sec. 6.1.

The scheme adopted for Motor Map-based attitude control is shown in Fig. 6.9, which refers to a single joint. The inner feedback loop is regulated



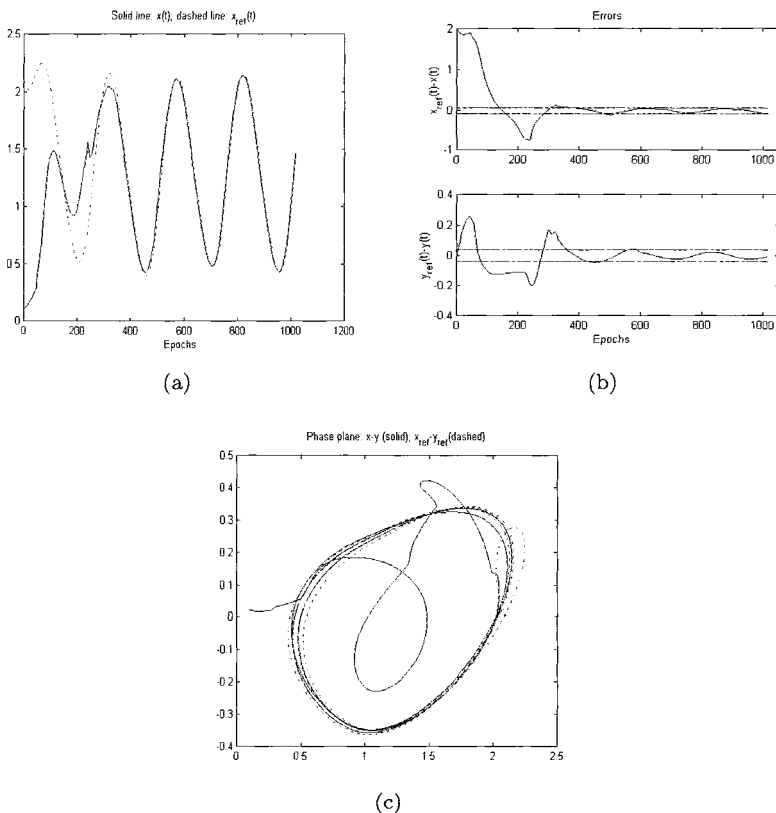


Fig. 6.8 Tracking of a limit cycle with 2 MMCs (each MMC acts independently on a variable of the controlled system): (a) trend for the variable  $x$  of the controlled system and the variable  $x_{ref}$  of the reference system; (b) limit cycle of the controlled system (solid line) and the reference system (dashed line) ; (c) errors and error bounds for the variables  $x$  and  $y$ . The errors are within the admissible error bounds after a short transient phase in which the MMCs learn the control actions.

by an integral controller, while the Motor Map provides the adaptive time-varying gain for this controller. Since the aim of the control is to keep  $\phi_d = 0$  and  $\theta_d = 0$ , the MMC learns the correct value of the gain parameter on the basis of the following Reward Function:

$$R = -(\phi^2 + \theta^2) \tag{6.16}$$

Several examples of attitude control are given below. In these examples

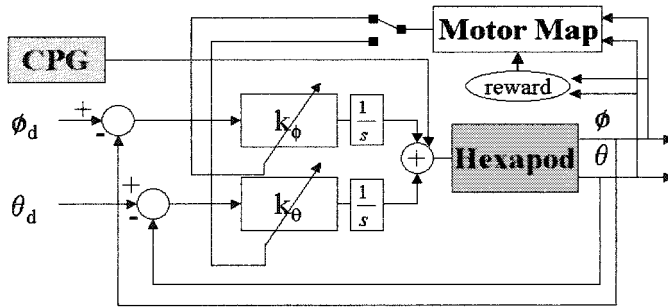


Fig. 6.9 Motor Map for attitude control in the hexapod robot.

the MMC has been applied to the dynamic hexapod model HexaDyn and acts parallel to the CPG-based locomotion control. The MMC consists of either two Motor Maps for independent, parallel control of roll and pitch, or a single Motor Map controlling the variable with the largest error. The results for the two cases are similar, so below we will only refer to the single MMC scheme, given the advantages related to the use of only one Motor Map. In all the example the MMC consists of 16 neurons.

In the first example, the HexaDyn robot walked on a rough terrain until it reached a position characterized by large roll and pitch errors. At this point HexaDyn stops walking and begins controlling its attitude. Therefore, in this first example the CPG control is not active, while the attitude control is turned on. This example allows us to evaluate the performance of the MMC in two distinct cases. In the first case only an integrative action is taken, while in the second case an integrative-proportional action is performed. This latter case makes use of two distinct Motor Maps, one giving the value of the proportional action, the other that of the integral action. Figure 6.10 shows the pitch and roll errors for both cases. As can be noticed, when the proportional action is present, oscillations are damped out.

In the second example the MMC automatically activates when the error exceeds a fixed threshold: here again the robot stops walking and then controls its posture. Figure 6.11 deals with a simulation in which HexaDyn walks on a flat terrain until it reaches an upward slant. The pitch error due to the slope can be appreciated in Fig. 6.11(a), where the attitude control is off. Fig. 6.11(b) shows how, when HexaDyn approaches the upward slope,

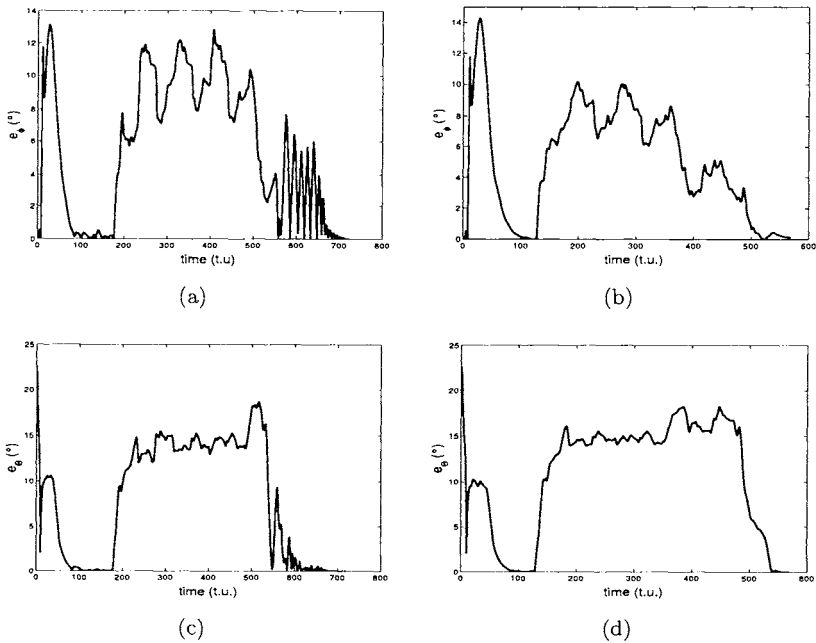


Fig. 6.10 Attitude control of HexaDyn through Motor Maps: (a) pitch error  $e_\phi(^{\circ})$  (integrative action); (b) pitch error  $e_\phi$  (proportional-integrative action); (c) roll error  $e_\theta(^{\circ})$  (integrative action); (d) roll error  $e_\theta$  (proportional-integrative action).

a large (over the threshold) pitch error appears; the posture is rapidly controlled and then HexaDyn continues walking.

In the third case the robot controls its attitude while walking. Figure 6.11(c) refers to this case. As can be noticed, the pitch error never reaches the values shown in Fig. 6.11(a).

The behavior of HexaDyn on rough terrain is further investigated in the most interesting case, in which the MMC is always active. Figure 6.12 shows the pitch and roll error and the winner neurons. As can be noticed this is maintained at low values by the MMC. At each simulation step the MMC controls either the roll or the pitch of the robot model, choosing to control the variable with the largest error. To distinguish between neurons activated for roll control and pitch control, a variable has been stored; this allows us to plot the winner neuron for roll and pitch control in separate graphs, as in Fig. 6.12(c)-(d). The corresponding waveforms of the driving signal for the  $\beta$ -joint of legs R1 and L1 are shown in Fig. 6.13.

A further example deals with an environment in which flat terrain is

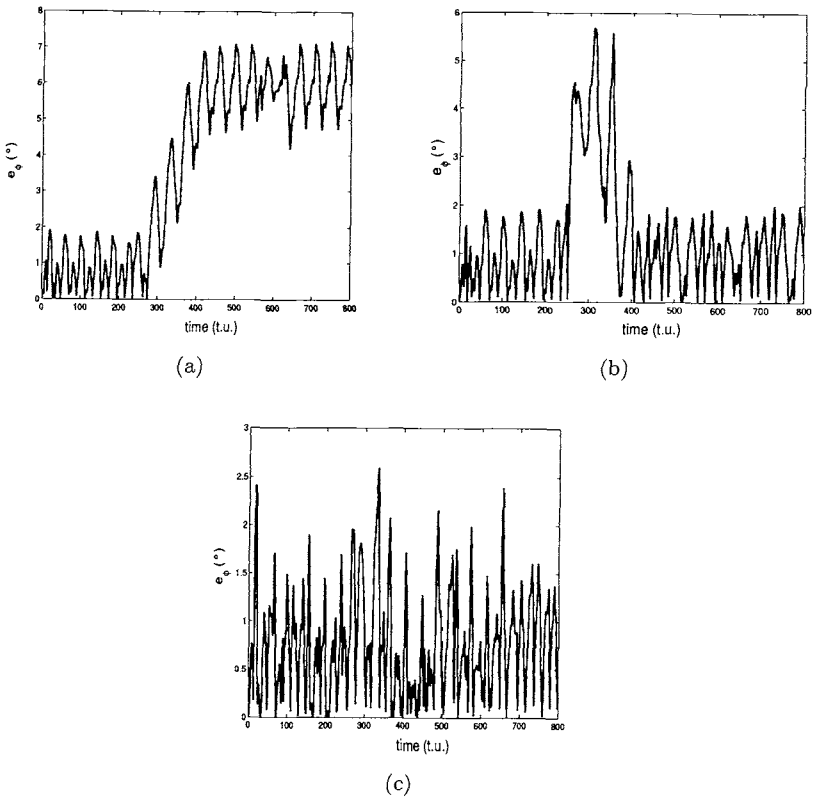


Fig. 6.11 Pitch error  $e_\phi(^{\circ})$  of HexaDyn walking on an upward slant (the sloped plane is approached at  $t \simeq 300(t.u.)$ ): (a) no attitude control; (b) MMC with threshold; (c) MMC always active.

followed by an upward slant, then by flat terrain and a downward slope. In the absence of attitude control walking on this terrain causes the pitch errors shown in Fig. 6.14(a). Figure 6.14(b) shows the effect of the attitude control. The corresponding waveforms of the driving signal for the  $\beta$ -joint of legs  $R1$  and  $L1$  are illustrated in Fig. 6.15.

#### 6.2.4 Motor Map-based attitude control in a simplified biped model

The results presented in Sec. 6.2.3 refer to the use of the MMC as an adaptive controller establishing the gain of the integrative action of the

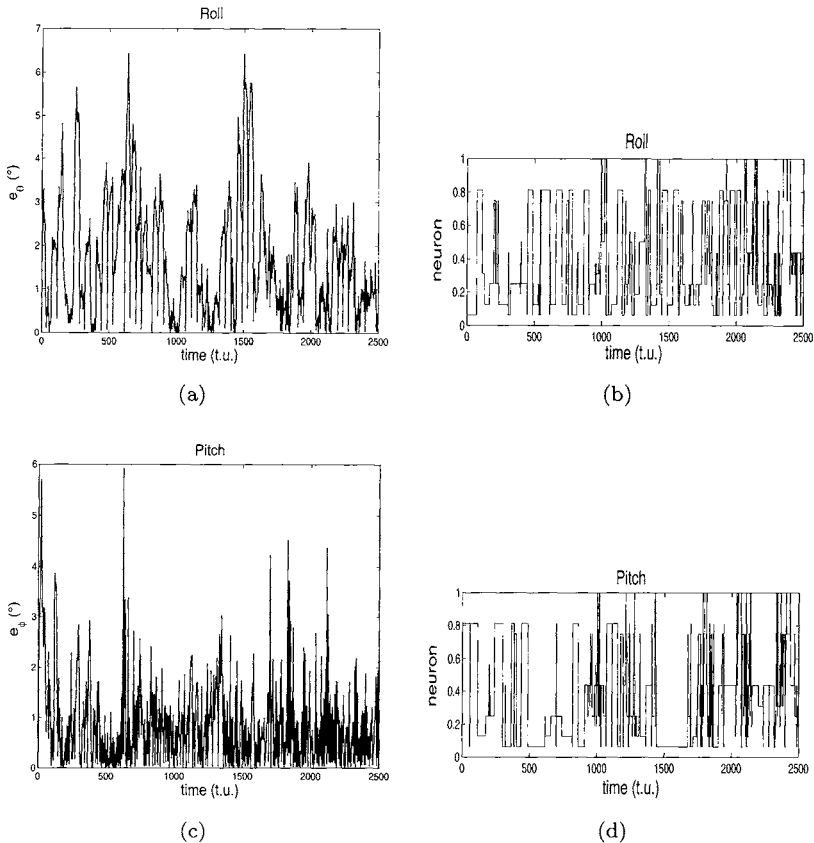


Fig. 6.12 HexaDyn walking on rough terrain: (a) roll error  $e_\theta$ ; (b) pitch error  $e_\phi$ ; (c) roll winner neuron; (d) pitch winner neuron.

inner feedback loop: in this case the objective of the control is fixed ( $\phi_d = 0$  and  $\theta_d = 0$ ). In this Section a case study is presented in which it is the MMC that gives the reference signals for a stabilizing inner feedback loop, while the parameters of the PID of the inner loop are kept constant. The case study is represented by a simplified model of a biped robot. The biped is in the single-leg support phase, therefore one leg is on the ground, while the other is a disturbance for the system. The support leg has two links and can only move on the sagittal plane. The knee and the hip of the support leg are actuated, while the ankle joint acts as a source of disturbance for the system.

According to the Lagrange formulation [Siciliano and Sciavicco (1989)],

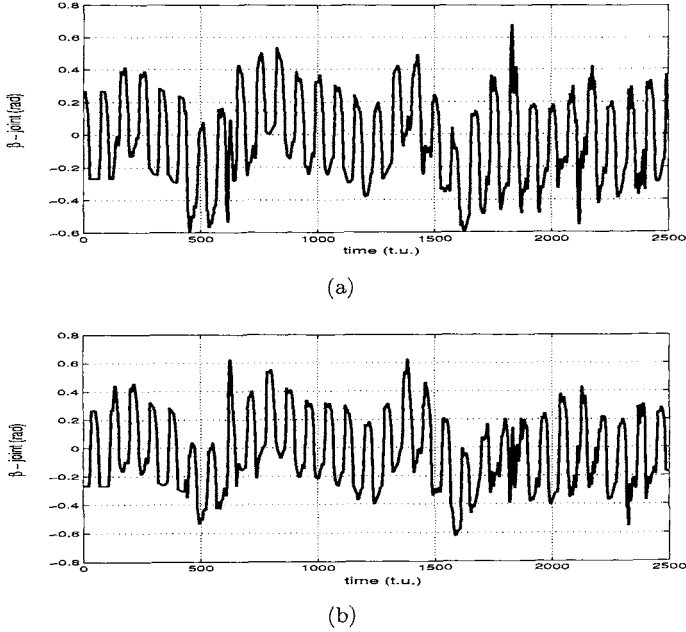


Fig. 6.13 HexaDyn walking on rough terrain: driving signals of the  $\beta$ -joint of legs R1 (a) and L1 (b).

the model of the biped, shown in Fig. 6.16, can be described by the following equations:

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{F}_v\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} \quad (6.17)$$

where  $\mathbf{q} = \begin{bmatrix} \theta_{hip} \\ \theta_{knee} \\ \theta_{ankle} \end{bmatrix}$ ,  $\boldsymbol{\tau} = \begin{bmatrix} \tau_{hip} \\ \tau_{knee} \\ \tau_{ankle} \end{bmatrix}$ ,  $\mathbf{B}(\mathbf{q})$ ,  $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ ,  $\mathbf{F}_v$  and  $\mathbf{g}(\mathbf{q})$  are,

respectively, the 3x1 vector of joint positions, the 3x1 vector of torques of joint actuators, the 3x3 positive definite joint space inertia matrix, the 3x1 vector of centrifugal and Coriolis effects, the 3x3 diagonal matrix of viscous friction and the 3x1 vector accounting for the effects due to gravity.

The joint torques are given by the inner feedback loop (consisting of a PD controller) and are therefore:

$$\boldsymbol{\tau} = \mathbf{K}_p(\mathbf{q}_d - \mathbf{q}) - \mathbf{K}_d\dot{\mathbf{q}} \quad (6.18)$$

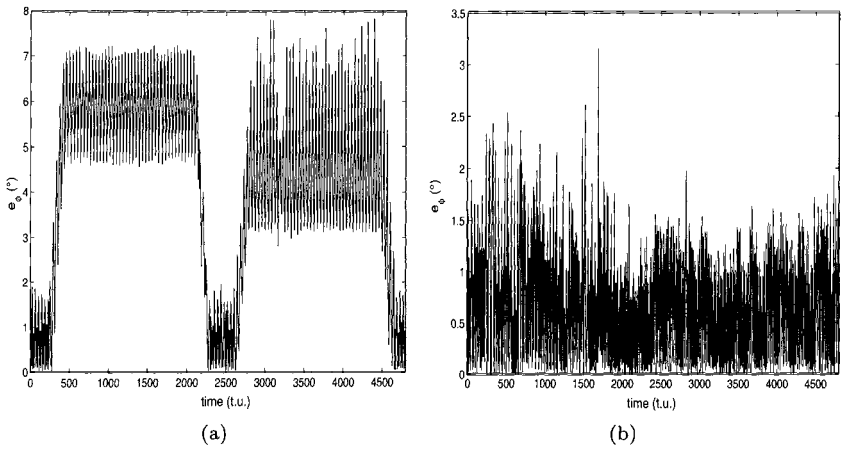


Fig. 6.14 Attitude control of HexaDyn walking in an environment with an upward slope, followed by a flat stretch and a downward slope. Pitch error  $e_\phi$ : (a) attitude control off; (b) attitude control on.

where  $\mathbf{K}_p$  and  $\mathbf{K}_d$  are  $3 \times 3$  diagonal matrices of proportional or derivative gains.

Given a desired joint position  $\mathbf{q}_d$ , the inner feedback loop makes the biped able to assume this configuration. Of course, this kind of control does not assure the stability of the biped. For this purpose the MMC has been introduced. Of course, the two feedback loops illustrated in Fig. 6.17 have different bandwidths.

The stability of the biped can be tracked by the Zero Moment Point (ZMP)[Park and Chung (1999)]. When the projection of the ZMP is on the foot the biped is stable. However, for sake of simplicity the projection of the center of gravity ( $x_{COG}$ ) has been taken into account here. This parameter defines the Reward Function. Since the objective of Motor Map-based control is, given a disturbance, to maintain the COG projection inside the foot and in particular at  $x = 0$  (see Fig. 6.16), the Reward Function has been defined as follows:

$$R = -x_{COG}^2 \quad (6.19)$$

The ankle joint position is a disturbance that can lead to a  $x_{COG}$  outside the supporting polygon: the MMC allows the supporting leg to reconfigure

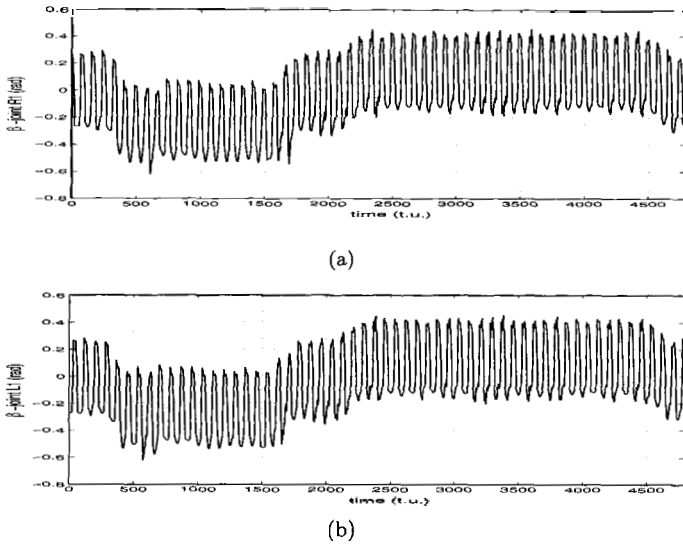


Fig. 6.15 Attitude control of HexaDyn walking in an environment with an upward slope, followed by a flat stretch and a downward slope: driving signals of the  $\beta$ -joint of legs R1 (a) and L1 (b).

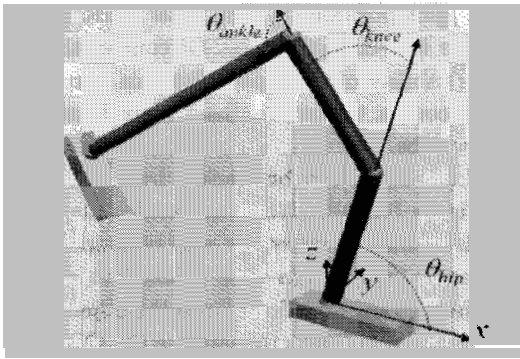


Fig. 6.16 The biped model.

itself to bring  $x_{COG}$  back to stable values.

Figures 6.18 and 6.19 illustrate a simulation example. The value of the disturbance leg angle was fixed at  $\theta_{ankle,d} = \frac{7}{6}\pi$ , while  $\theta_{hip,d}$  and  $\theta_{knee,d}$  are given by the MMC to minimize the Reward Function (6.19). Figure 6.18



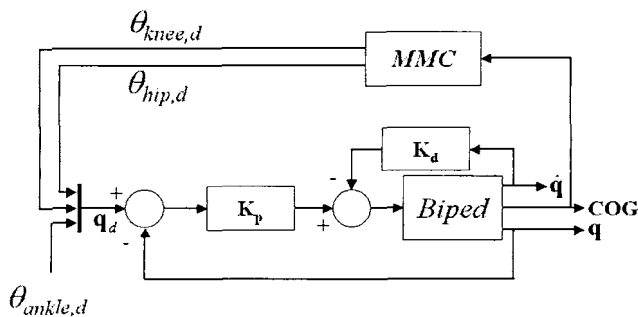


Fig. 6.17 The biped control scheme.

shows the waveforms of the joint variables (desired and actual): these are almost indistinguishable except for the transient phase when the reference changes according to the MMC output. The different bandwidths of the two feedback loops are evident in the panels in Fig. 6.18. As can be noticed in Fig. 6.19,  $x_{COG}$  converges to zero as the Motor Map learns the appropriate control law.

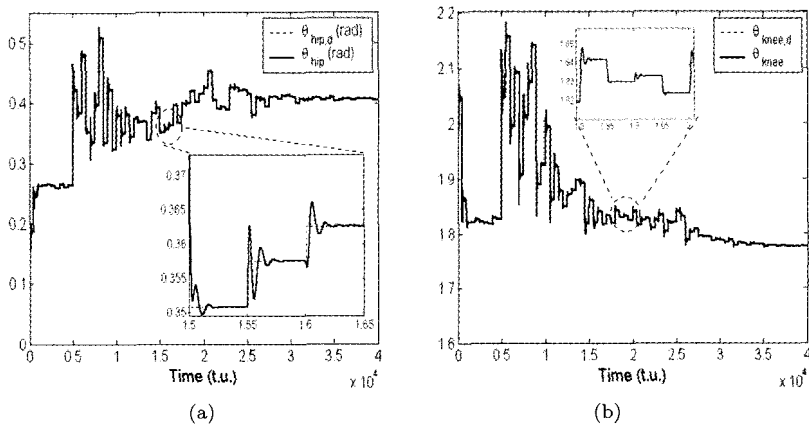


Fig. 6.18 Attitude control of the biped: trends of the desired (dashed line) and actual (continuous line) joint variables. (a)  $\theta_{hip}$ . (b)  $\theta_{knee}$ .

The example discussed here may resemble the inverted pendulum control reported in [Ritter *et al.* (1992)]. However, in [Ritter *et al.* (1992)]

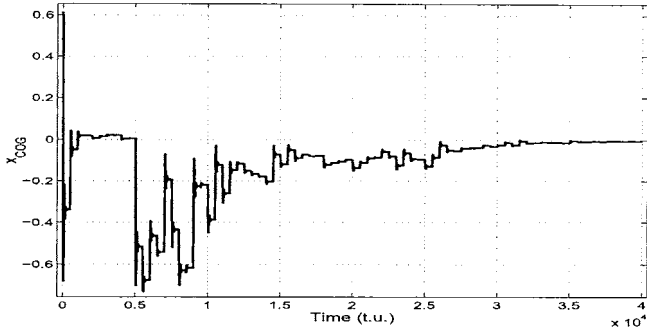


Fig. 6.19 Trend of  $x_{COG}$ .

the Motor Map directly controls the force to be applied to the inverted pendulum, thus leading to a huge number of neurons and a long learning phase. The scheme presented here, making use of an inner feedback loop, allows the number of neurons needed for the task to be reduced. This is an important condition for the feasibility of the hardware implementation.

### 6.3 Learning with Motor Maps

There is no intelligence without learning. All the examples reported here highlight the importance of learning in bio-inspired control. On the other hand, learning is a challenge for mobile robots for two reasons: firstly, the learning times should be acceptable, and secondly learning has to avoid situations which are dangerous for the robot. Despite the great interest in these topics, there are few works implementing real-time learning for robots. For example Kirchner [Kirchner (1997)] proposes a hierarchical algorithm for learning complex behavior based on reinforcement.

Motor Maps provide a suitable way to implement reinforcement-based learning. This is an important concept that opens up the way to new applications. For instance as regards attitude control, it should be noted that posture control in insects is a very complex form of behavior. Experimental data, illustrated in [Watson *et al.* (2002)], show that the cockroach adopts different types of behavior to overcome obstacles along its path according to the height of the barrier. If the barrier is small it does not change its posture, while if the barrier is high, it heightens the middle leg to overcome the obstacle. Of course, this complex behavior involves vision and high-

level centers to determine the height of the obstacle although the actual mechanism is still unclear. However, this is very interesting behavior in which the MMC can be useful to build up an efficient bio-inspired robot able to learn how to overcome obstacles with different strategies.

An example of learning with Motor Maps is discussed in the following. In Chap. 2 the value of  $\varepsilon$  for a given gait was found by using numerical simulations. Motor Maps can be used to build an adaptive system able to find the appropriate value of  $\varepsilon$  for a given locomotion gait without any prior information. This problem is clearly a problem of learning, the Motor Map makes the system able to self-organize itself passing from an unorganized to an organized behavior.

The block scheme of the control loop is shown in Fig. 6.20. The CNN-based CPG generates the locomotion gait for the robot: this gait is now adaptive, and its characterizing parameter ( $\varepsilon$ ) is varied by the MMC. Let us for instance focus on fast gait and let us suppose that the parameter  $\varepsilon = -\varepsilon_f$  is unknown. The Motor Map output weight directly maps the value of this parameter.

In order to find the appropriate value of the parameter, the Reward Function has to be defined. It takes into account the error between the reference speed  $v_{ref}$  and the actual speed  $v$  of the robot as follows:

$$R = -(v_{ref} - v)^2 \quad (6.20)$$

Of course, the objective of the control is to maximize the Reward Function. The definition of speed for a walking machine has to be clarified: in fact while it is unambiguous for a wheeled robot, the same does not hold for a legged robot. Walking robots do not move at a constant speed, but are subjected to accelerations due to the nature of step locomotion. An average of the speed over three complete cycle times was therefore taken into account.

Some simulation results using HexaDyn are shown in Fig. 6.21(a), where the trend of the parameter  $\varepsilon$  is reported. It should be pointed out that, as learning is always active, small fluctuations around the average value are present. Moreover, the average value is similar to the value  $\varepsilon_f = -0.6$  given in Table 2.2. The robot starts from unorganized leg movements and learns how to synchronize the legs in order to achieve a pattern of coordinated movements.

Figure 6.21(b) illustrates the winning neuron. In this case two neurons have been trained for the fast gait. This is not a general result, since in

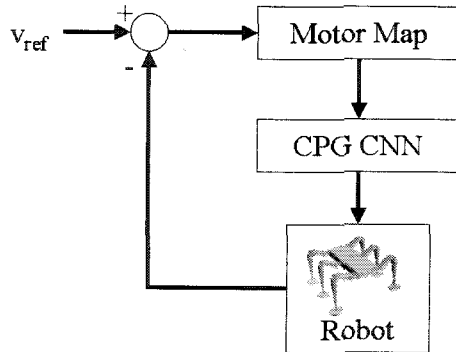
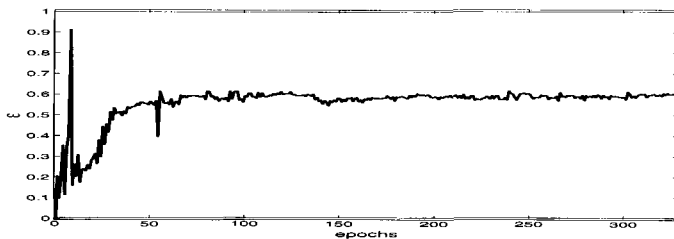
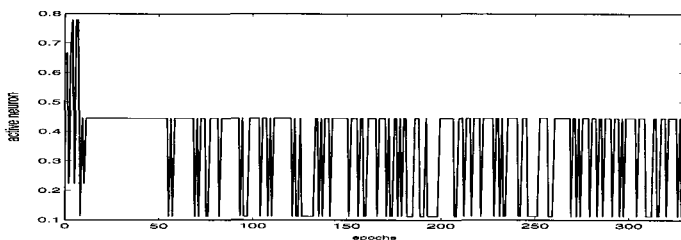


Fig. 6.20 Block scheme of hierarchical control of the robot. The locomotion is generated at the CPG level, the MMC controls the parameters of the CPG on the basis of speed feedback.

most cases only one neuron is trained. However, this redundancy is due to the fact that only one reference input is presented to the system, so further learning steps can lead to a higher specialization. The approach based on Motor Map can be extended to control the generalized continuous gait.



(a)



(b)

Fig. 6.21 Learning a given locomotion gait (fast gait): (a) trend of the parameter  $\epsilon$ ; (b) winning neuron.

## Chapter 7

# High-level analog control: Turing patterns and autowaves

This Chapter deals with navigation control from the perspective of analog computation. Two approaches are introduced: the first is based on Turing patterns and the second on autowaves. The two paradigms, Turing Patterns and autowaves, are solutions of a Reaction-Diffusion (RD) equation and are both implemented on a CNN. Therefore the first Section of the Chapter is devoted to a brief introduction to RD-CNNs.

The common idea underlying the two approaches is the use of a RD medium as an analog processor for robot navigation. The analog processor plays the role of a high-level control center or a pool of command neurons, providing the robot with high-level capabilities for obstacle avoidance and trajectory planning.

The first example concerns the implementation of reactive behavior in an array of locally coupled command neurons through Turing patterns. The behavior of the robot is regulated by the status (the pattern) of the whole neuron ensemble rather than that of single neurons. In the second case, wave computing, an emerging paradigm of nonlinear science, is applied to the problem of robot navigation in a complex environment with obstacles.

As regards practical issues, the two approaches can be used to control the locomotion of the robots introduced in Chap. 5. However, for the sake of simplicity the experimental examples given in this Chapter deal with simple roving robots implemented using LEGO MindStorms<sup>TM</sup> which are gaining increasing interest in robotics as low-cost, easy-to-build and re-configurable mobile robot kits for educational and research objectives [Greenwald and Kopena (2003); Klassner and Anderson (2003)].

## 7.1 Reaction-Diffusion CNN

The two control strategies that will be discussed in this Chapter are both based on RD equations in a two dimensional field. The general RD equation is here recalled:

$$\frac{d\mathbf{u}}{dt} = f(\mathbf{u}) + \nabla^2 \mathbf{u} \quad (7.1)$$

Like the discretized Laplacian introduced in Sec. 2.4.1 the following discretization of the diffusive term in a 2D array can be assumed:

$$\nabla^2 \mathbf{u} = u_{i+1,j} + u_{i,j+1} - 4u_{i,j} + u_{i-1,j} + u_{i,j-1}$$

Taking into account this assumption and a reactive part similar to Eq. (2.5) the following equations for a single generic cell  $C_{ij}$  of the RD-CNN can be written:

$$\begin{cases} \dot{x}_{1;ij} = -x_{1;ij} + (1 + \mu + \varepsilon)y_{1;ij} - s_1 y_{2;ij} + D_1(y_{1;i+1,j} + \\ \quad + y_{1;i-1,j} + y_{1;i,j+1} + y_{1;i,j-1} - 4y_{1;ij}) \\ \dot{x}_{2;ij} = -x_{2;ij} + (1 + \mu - \varepsilon)y_{2;ij} + s_2 y_{1;ij} + D_2(y_{2;i+1,j} + \\ \quad + y_{2;i-1,j} + y_{2;i,j+1} + y_{2;i,j-1} - 4y_{2;ij}) \end{cases} \quad (7.2)$$

where  $i = 1..M$ ,  $j = 1..N$  and  $y_{h;ij} = 0.5(|x_{h;ij} + 1| - |x_{h;ij} - 1|)$  with  $h = \{1, 2\}$ .

According to how the parameters are chosen, the RD-CNN (7.2) can show either Turing patterns or autowaves.

Turing patterns are steady-state solution of the evolution of differentiated dynamic behavior arising from local interactions between identical elements. The emergence of these patterns has been observed in a large variety of phenomena (biological cells, particles, chemical reactions) and was demonstrated in arrays of locally coupled circuits in [Goraş *et. al.* (1995a); Goraş and Chua (1995); Goraş *et. al.* (1995b); Arena *et. al.* (1998)].

The shape of the pattern depends on the initial conditions, boundary conditions and the dimensions of the spatial domain. For instance, if the parameters of the RD-CNN are chosen according to Table 7.1(a), a non-homogeneous steady-state emerges. Figure 7.1(a) shows an example of a Turing pattern arising from a 25x25 RD-CNN (7.2) with zero flux boundary conditions and the following initial conditions:

$$\begin{aligned} x_{1,ij}(0) &= \begin{cases} 1 & \text{if } 6 \leq i, j \leq 11 \\ -1 & \text{otherwise} \end{cases} \\ x_{2,ij}(0) &= \begin{cases} -1 & \text{if } 6 \leq i, j \leq 11 \\ 1 & \text{otherwise} \end{cases} \end{aligned} \quad (7.3)$$

Adopting the same convention of Fig. 7.1, these initial conditions are a black square near the top left corner of the image representing the first layer of the RD-CNN.

The same RD-CNN is also able to generate autowaves. If the parameters are chosen according to Table 7.1(b), the RD-CNN reproduces the behavior of a nonlinear medium in which autowaves propagate. It is worth remarking that in this case the reactive part is practically the same one implementing the CNN neuron dynamics. Figure 7.1(b) shows an example of RD-CNN generating autowaves. Zero flux boundary conditions have been chosen. The following initial conditions have been assumed:

$$\begin{aligned} x_{1,ij}(0) &= \begin{cases} 1 & \text{if } j = 3 \\ 0.4980 & \text{if } j = 4 \\ -0.5059 & \text{if } j = 4 \\ -1 & \text{otherwise} \end{cases} \\ x_{2,ij}(0) &= \begin{cases} 0.4980 & \text{if } j = 4 \\ -0.5059 & \text{if } j = 5 \\ -1 & \text{if } 6 \leq j < 7 \\ 1 & \text{otherwise} \end{cases} \end{aligned} \quad (7.4)$$

These initial conditions represent a vertical black line giving rise to autowave propagation.

Figure 7.1 also shows an important difference of the single cell behavior in the two cases. In the case of autowaves the single cell dynamics is a stable limit cycle, while in the case of Turing patterns the trajectory of each cell state variable converges into an equilibrium point.

Table 7.1 Parameter values of the RD-CNN.

	$\mu$	$\varepsilon$	$s_1$	$s_2$	$i_1$	$i_2$	$D_1$	$D_2$
CASE (a) (Turing patterns)	-0.6	1.82	2	2.5	0	0	0.01	1
CASE (b) (Autowaves)	0.7	0	1	1	-0.3	0.3	0.1	0.1

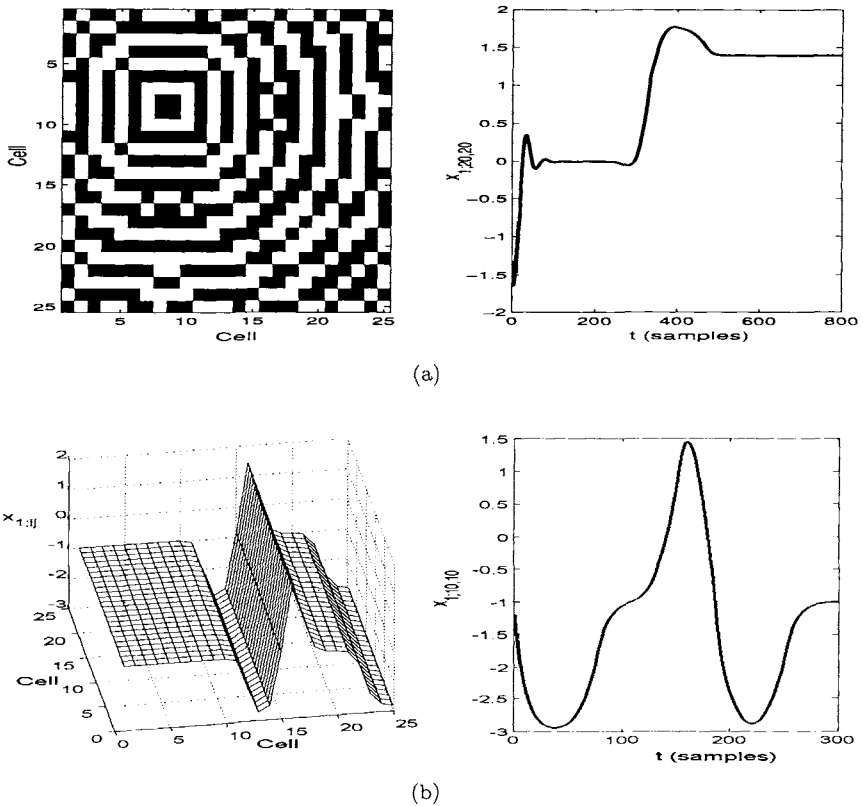


Fig. 7.1 (a) Example of RD-CNN generating Turing patterns (black pixels +1, white pixels -1). (b) Example of RD-CNN generating autowaves.

## 7.2 Navigation control based on Turing patterns

In order to design a robot that has to move in a complex environment, it seems mandatory to deal with terms, concepts and ideas from various disciplines: ethology, neurology and psychology. The concept of *behavior* by itself is rich in different meanings and is strictly related to the animal world. A broad picture of the robot control system spectrum is given in [Arkin (1998)] and adapted in Fig. 7.2. Deliberative reasoning and purely reactive behavior constitute the extremes of this diagram. Most advantages in terms of speed of response are achieved by adopting a purely reactive control strategy, while predictive capabilities grow if deliberative systems are used. Most of the literature describing robot behavior, however, is



based on traditional software architectures that suffer when the number of sensors is high.

This Section focuses on an analog implementation of reactive behavior, which is located as shown in Fig. 7.2. The proposed implementation is based on a system of analog dynamical nonlinear units, locally coupled. These constitute a pool of neurons implementing fixed-action patterns for a robot. Fixed-action patterns are time-extended response patterns triggered by a sensor stimulus persisting for longer than the stimulus itself [Arkin (1998)]: unlike reflexes, they can be motivated and therefore can be stimulated by the internal state of the animal. The robot wanders in the environment and the status of its sensors is processed by the network of neurons. As a result of the evolution of these neurons a pattern emerges, and drives the robot through a series of actions.

The behavior of the robot is, therefore, determined by the sensor data. Obviously, this is only an easy task if the number of sensors is low. When the number of sources grows, it requires more and more computational resources. This is not the case of the information processing that takes place in the nervous system of animals and humans, where the high processing capabilities arise from the complexity of the whole system and not from a powerful computational unit. Taking inspiration from this principle the approach proposed here is fully analog, based on simple neural cells, able to perform parallel data processing. The importance of an analog approach lies in the fact that it is independent of the number and kind of sensors: data fusion is performed in an analog, fully parallel way. Moreover, the topological information coming from the displacement of the sensors can be preserved in the spatially extended neural system.

The network of neurons is implemented by a RD-CNN generating Turing patterns. This system of locally coupled units starts from an initial condition reflecting the sensor status and the previous pattern and evolves towards a new pattern that will establish the action of the robot.

This Section introduces the CNN controlling the robot and presents the experiments carried out on a small roving robot.

### 7.2.1 Turing patterns and CNNs

The key mechanism of pattern formation thoroughly discussed in [Goraş *et. al.* (1995a)] is based on two considerations: each isolated cell should have a unique stable equilibrium point; moreover, this equilibrium should also be an equilibrium point for the whole interconnected array, but un-

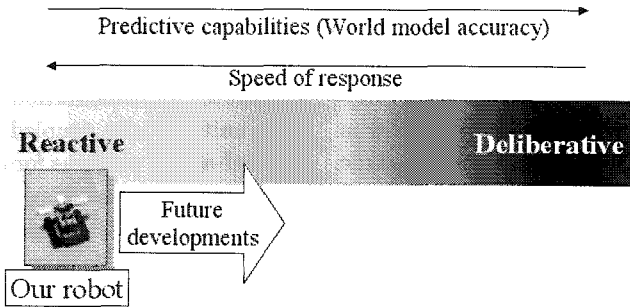


Fig. 7.2 Spectrum of robot behavioral control.

stable in order to give rise to the evolution of a pattern corresponding to other equilibrium points of the array. From these conditions an analytical approach can be used to derive the parameters of the two-layer RD-CNN [Goraş and Chua (1995)].

The technique developed in [Goraş and Chua (1995)] allows us to derive analytical formulas to satisfy the conditions for pattern formation and is based on decoupling the linearized state equations of the whole array by looking for a particular kind of solution in the form of a sum of spatial eigenfunctions. The temporal eigenvalues associated to these spatial eigenfunctions cannot all have a negative real part, since in this case the homogeneous equilibrium point will be stable and no pattern can evolve. At least one eigenvalue should have positive real part. The spatial eigenfunction associated to this unstable eigenvalue and the effect of the nonlinearity of the system will lead to a stable evolved spatial pattern.

In order to satisfy this condition the so-called *dispersion curve* is examined. This curve represents the relation between temporal eigenvalues and spatial eigenvalues and allows us to obtain the band of unstable modes of the system. The parameters of the RD-CNN should be chosen in order to have at least one spatial mode in this band.

However, this linear theory can only predict how the pattern begins to evolve and does not exclude the possibility of other patterns arising. In other words nonlinearity plays a fundamental role in these systems to determine the final pattern. In fact, when more than one unstable eigenfunction is present, nonlinearity is fundamental to establish which of the unstable modes will win the competition between the possible patterns.

### 7.2.2 From CNN patterns to action patterns

The idea underlying the approach presented here is that the pool of neurons, represented by the CNN generating Turing patterns, evolves towards a pattern that is a function of the sensor status revealed by the roving robot: a pattern is generated each time the sensor status changes. This pattern constitutes the motion instructions for the robot to respond to the perceived situation. To this end it is necessary to take into account the sensor status in the CNN evolution: this has been done by acting on the initial conditions of the network. Only the initial conditions of a few cells were related to the sensor status (one for each sensor). Moreover, to further simplify the algorithm in view of the hardware implementation it was decided to set the initial conditions of these cells not analogically but only to two different values indicating two distinct states of the sensor. In this Section the choice of the CNN parameters (cell parameters, dimension and boundary conditions) is illustrated.

The key point in designing the CNN is to guarantee that given a certain pattern at a discrete time  $k$  and given the sensor status at time  $k$ , the CNN will evolve towards a new pattern which on the basis of the sensor status indicates to the robot the direction to be pursued. Therefore, the CNN initial conditions are fixed by taking the previous pattern as the initial condition for all the cells except those connected to the sensors.

The example given in this Chapter deals with a small roving robot equipped with only three sensors (see Fig. 7.3). The aim of the control system is to generate fixed-action patterns to enable the robot to avoid obstacles. The small number of sensors makes the fusion of data from these sensors an easy task if pursued with traditional methods. However, this simple example demonstrates the validity of the proposed approach, the advantages of which with a high number of data sources are evident.

The parameters of the RD-CNN (7.2) have to be chosen so as to satisfy the pattern formation conditions. They were chosen according to Table 7.1(a).

The dimensions of the array and the boundary conditions are chosen so as to satisfy the following considerations. First of all, the network should be able to generate enough stable patterns to provide the robot with a sufficient number of strategies. Each pattern, in fact, constitutes a fixed-action pattern for the robot. Therefore the number of unstable modes is a key factor in choosing the CNN. The unstable modes also depend on the boundary conditions. Moreover, the actual pattern (at time  $k$ ) evolves from

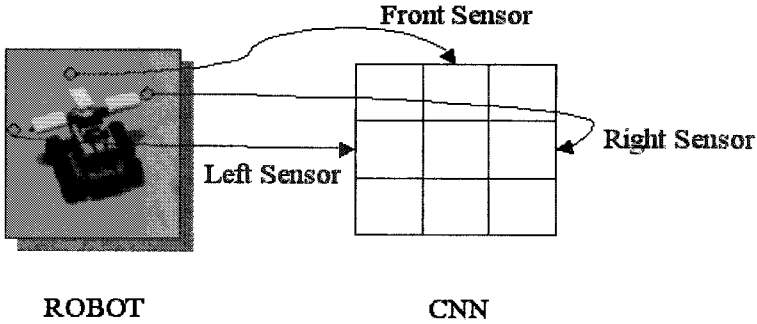


Fig. 7.3 Correspondence between sensors on the robot and CNN cells.

initial conditions obtained by merging the pattern at time  $k - 1$  and the sensor status and has to be effective to implement the obstacle avoidance behavior.

The low number of sensors implies the possibility of choosing a small CNN. A 3x3 CNN with cyclic boundary conditions proved to be satisfactory to obtain the desired performance.

The eigenfunctions satisfying the cyclic boundary conditions are the following [Goraş and Chua (1995)]:

$$\begin{aligned}\phi_{MN}(m, n, i, j) &= e^{j \frac{2\pi}{M} mi} e^{j \frac{2\pi}{N} ni} \\ \phi_{MN}(m, n, i, h) &= e^{j \frac{2\pi}{M} mi} e^{j \frac{2\pi}{N} nh}\end{aligned}\quad (7.5)$$

while the corresponding eigenvalues are given by:

$$K_{mn}^2 = 4 \sin^2 \frac{m\pi}{M} + 4 \sin^2 \frac{n\pi}{N} \quad (7.6)$$

For a 3x3 CNN the matrix of unstable modes is the following:

$$K = \begin{bmatrix} 0 & 3 & 3 \\ 3 & 6 & 6 \\ 3 & 6 & 6 \end{bmatrix} \quad (7.7)$$

The dispersion curve is shown in Fig. 7.4. Since  $k_1^2 = 1.83$  and  $k_2^2 = 115.77$ , eight modes are in the unstable band.

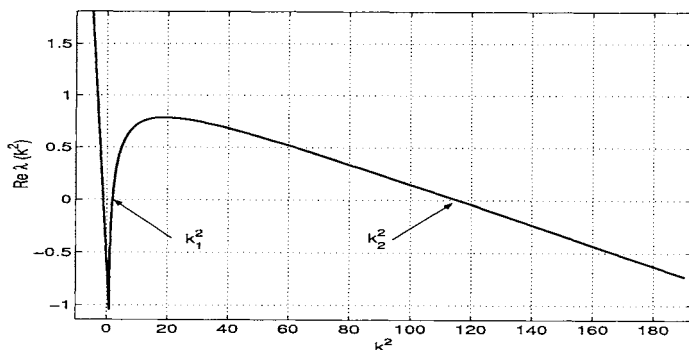


Fig. 7.4 Dispersion curve for cell (7.2).

The CNN was simulated with different initial conditions on the first layer (variable  $x_{1,ij}$ ), while they were kept constant for the second layer. The number of patterns obtained in simulation is greater than that predicted by the linear theory. However, by restricting the analysis to the different initial conditions that can be obtained by considering only the cells related to sensors (the correspondence was fixed in order to reproduce the displacement of sensors as shown in Fig. 7.3) eight patterns were obtained. These are shown in the middle column of Fig. 7.5. Each pattern corresponds to the fixed-action pattern shown in the right column of Fig. 7.5, and is the result of the evolution of the previous pattern merged with the initial conditions from the obstacle position depicted in the left column of the Figure.

### 7.2.3 Experimental Setup

As is known, CNNs are suitable for VLSI realization, so the whole approach is suitable for implementation on an autonomous robot. However, in this work the experimental setup is provided by simplified hardware. A roving robot was built using LEGO MINDSTORMS™ [MindStorms webpage], while the CNN was implemented on a PC. They communicate through an infrared device, included in the LEGO toolbox. A routine in the CNN software allows the action pattern for the roving robot to be downloaded as a program, written using nqc language [Baum (2002)]. A scheme of the experimental setup is shown in Fig. 7.6: the roving robot, equipped with three light sensors, is able to detect black obstacles. This experiment shows

















Obstacle Position	CNN Pattern	Fixed-action pattern
		Rotate by 180°
		Rotate by -90° (clockwise)
		Rotate by +90° (counterclockwise)
		Rotate by -135° (clockwise)
		Rotate by +135° (counterclockwise)
		Go straight on
		Rotate by 180°
		Go straight on

Fig. 7.5 Obstacle position, CNN patterns and fixed-action patterns. CNN patterns refer to the first layer (black indicates low saturation levels, white high values).

the suitability of the technique which, of course, can be applied to the other bio-inspired robots introduced previously.

#### 7.2.4 To probe further

The approach described here consists of a CNN devoted to processing information from an arbitrary high number of sensors to control the reactive behavior of a roving robot. The analog, fully parallel processing capabilities of the CNN are exploited to obtain a stable Turing pattern representing the fixed-action pattern of the robot in order to obtain obstacle avoidance

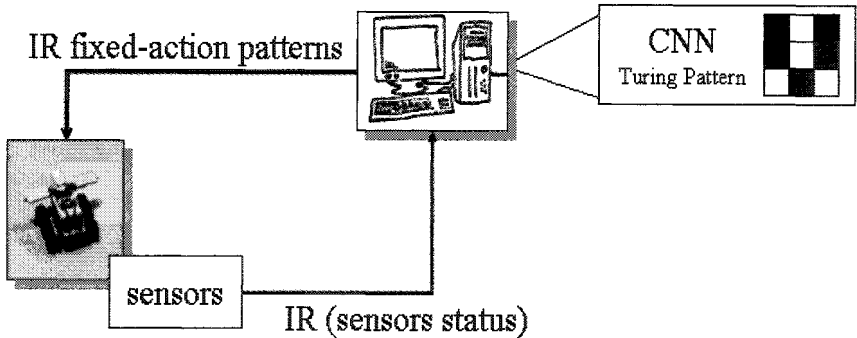


Fig. 7.6 The experimental setup for the navigation control based on Turing patterns.

behavior. The analog solution to the problem of data fusion allows us to obtain fast responses, independently of the number of sensors, at the same time maintaining the topological information from the displacement of the data sources.

The decision to use CNN to implement the network of neurons controlling the behavior of the robot is motivated by the growing number of integrated chips presented in literature and opens up the way to implementation of the whole control system on an autonomous robot. A possible direction for future developments is indicated in Fig. 7.2: the possibility of including learning and considering high-level internal state motivations (or objectives) is particularly appealing. Another important point to be addressed is how Turing patterns are associated with a given action. This process should be the result of the dynamic evolution of the controller, thus learning should be included in the control scheme. A possible way to address this issue is the use of Motor Maps: on the basis of the Reward Function they can provide the system with adaptability and flexibility, enabling it to learn the correspondence between Turing patterns and robot actions.

Another strategy for navigation control based on wave computation in analog processors will be described in the following Section. The interesting point is that the basic cell can be kept the same. Thus CNNs provide a general framework for navigation control based on analog techniques.

### 7.3 Navigation control based on autowaves

In robotics a very common approach to the problem of navigation control is the use of artificial potential fields [Arkin (1998)], where the whole experimental arena is mapped onto the computational architecture of the robot navigator: obstacles generate repulsive fields and targets generate attractive fields. However the computational resources required by this approach are quite large, and therefore real-time adaptation to moving obstacles is difficult to achieve.

On the other hand there exist a large amount of literature on chemical controllers applied to calculate the optimal path for a robot moving in an obstacle-filled environment. These controllers are analog processors mostly based on the excitation wave dynamics in the Belousov-Zhabotinsky (BZ) medium [Agladze *et. al.* (1997)]. Their intrinsic parallel capabilities allow the limits of the potential field approach to be overcome, but their main drawback is the limited speed of diffusion of the wavefronts in chemical reactions. For instance when chemical excitable media are used to control phototaxis-based navigation of a mobile robot as in [Adamatzky *et. al.* (1998)], the idea is to excite the medium according to a light source, let the waves propagate in the medium and let the robot follow the gradient to reach some target or to avoid obstacles.

Starting from these considerations, we propose a scheme using an analog processors based on CNN applied to generate autowaves for the navigation control of a robot, thus overcoming the limits of chemical processors by keeping the peculiarities of the paradigm. The basic idea is to use RD equations that can be implemented in CNNs and reflect the nature of the BZ medium.

In our approach the RD medium is devoted to control the trajectory of a robot in an environment with obstacles, where a target point for the robot to reach can also be fixed. Both obstacles and target are mapped onto the nonlinear medium as autowave sources, stimulating different regions or cells. A key characteristic of autowaves is that they annihilate when they collide [Krinsky (1984)]; in such a way they intrinsically determine the path of points equidistant from the obstacles.

Therefore autowaves are very important for the navigation control, since the trajectory computation can be easily computed through their generation, propagation and interaction. From this perspective nonlinear media represent a class of parallel computers with unique features: parallel input of data, parallel information processing and parallel outputs [Adamatzky



*et. al.* (2004)].

More specifically, the robot arena is mapped on a CNN: its role is to select obstacles (or the target) and identify them as sources of repulsive (or attractive) autowaves. Subsequently a RD-CNN generates autowave fronts. The robot is allowed to move freely within the arena. A simple AND operation reveals the wavefront collisions with the robot, and a steering command is sent to the robot for the generation of highly adaptive trajectories. In Fig. 7.7 a schematic view of this approach is shown. This methodology allows the optimal robot trajectory in dynamically changing environment to be selected in an extremely short time: the algorithm, which solves a complex task, is in several cases even faster than the robot typical reaction times.

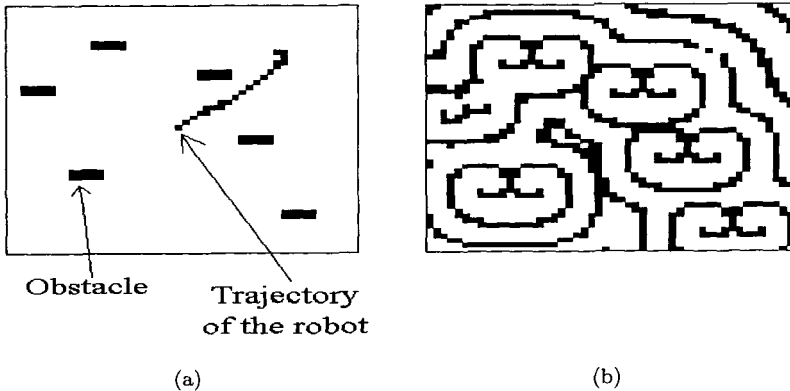


Fig. 7.7 Autowaves to control the navigation of a robot. (a) Trajectory of the robot and position of obstacles (the target, not shown, is on the upper right hand-side corner). (b) A frame of the autowaves propagating in the CNN medium (only the autowaves generated by the obstacles are shown).

### 7.3.1 The CNN algorithm

The algorithm for navigation control consists of two parts: target/obstacle recognition and wave computation based on RD Eqs. (7.2). Thanks to the universal computing properties of a CNN, a huge number of algorithms can be used, making them able to perform additional complex image filtering routines, which are needed to have a complete and really working algorithm for robot trajectory planning. In fact, a first, essential step that has to be

performed a priori, is target/obstacle recognition. In our case, the input is provided by a video camera taking a motion picture of the environment where the robot moves. By taking into account the peculiar characteristics of the obstacles (i.e. their shape), several algorithms, already implemented and tested on VLSI CNN chips, can be adapted to the features of the environment and applied to provide an image containing only the obstacles surrounding the robot, whose position is assumed to be known at any given time. This image is processed by the RD-CNN (the analog processor) in which autowaves propagate.

The CNN cells corresponding to the position of obstacles are stimulated by setting initial conditions for the first layer of the CNN to  $x_{ij} = +1$ . Waves, generated at the obstacle positions, propagate until they collide with the robot. Depending on the kind of source generating the wave (obstacle or target) and on the particular compass direction the wavefront comes from, the new direction for the robot can be directly determined. Thus the trajectory planning is not performed before the robot starts moving, but step by step while the robot is moving, soon after each wave-robot collision. After the completion of a robot movement, one can decide whether to process another frame or not, i.e. one can decide either to reset the RD-CNN and start with a new configuration of obstacles and target, or to continue with the previously acquired configuration. This also allows moving obstacles to be taken into account or makes it possible to speed up the algorithm.

Two complementary RD-CNNs are used: one is the medium for propagation of waves generated by obstacles, while the second deals with the target. The flow diagram of the algorithm as regards the case of autowaves generated by obstacles is shown in Fig. 7.8. The robot is viewed as a four active pixel object with a given orientation according to Fig. 7.8. After a threshold operation an AND operation is performed between the snapshot of wave propagation, represented by the state of the first layer of the RD-CNN, and the image containing the robot. Depending on which of the robot pixels is first reached by the wavefront, a particular motion instruction is given to the robot. The parameters of the RD-CNN (7.2) are chosen according to Table 7.1(b).

Simulation results have been obtained by using a dedicated framework, written in C++, for the simulation of the robot and of the CNN algorithm. A  $50 \times 50$  CNN was simulated. Figure 7.9 shows several examples of simulation results. In this case, instead of a target to reach, the robot focuses on proceedings in a pre-specified direction (in this case north). Similar re-

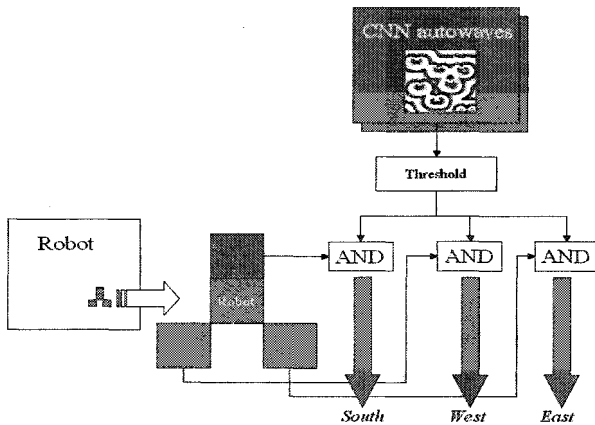


Fig. 7.8 Flow diagram of the CNN algorithm (the case of autowaves generated by obstacles is shown).

sults have been obtained in the case in which the target is also taken into account.

As can be seen from the simulation results, the algorithm presented is really effective. However, there may be cases in which the robot gets trapped in a local minimum, for example when target and obstacles are in particular symmetrical points with respect to the robot. In such cases the robot moves alternatively between the same two positions. To overcome such a condition, a random perturbation on the robot trajectory is added. Moreover, in view of a real implementation, it is not necessary for the video feedback to refer to the whole robot arena, but it can be assumed that the processing takes place only on a certain region surrounding the robot. Due to the adaptive characteristics of the methodology, the environment focused by the camera can vary as the robot moves. So for example, at the beginning, when the target could be out of the captured area, the robot is made to move only in the direction of the target. Its first task could therefore be to avoid obstacles while moving in a particular direction. As the robot position approaches the target, the final focus could be taken into consideration. Moreover, taking limited, next-to-robot frames improves the computational efficiency, since distant obstacles do not generate wavefronts; so the local trajectory is influenced only by obstacles in the immediate vicinity of the robot. These aspects are extremely important, in view of a real implementation on the RD-CNN chip.

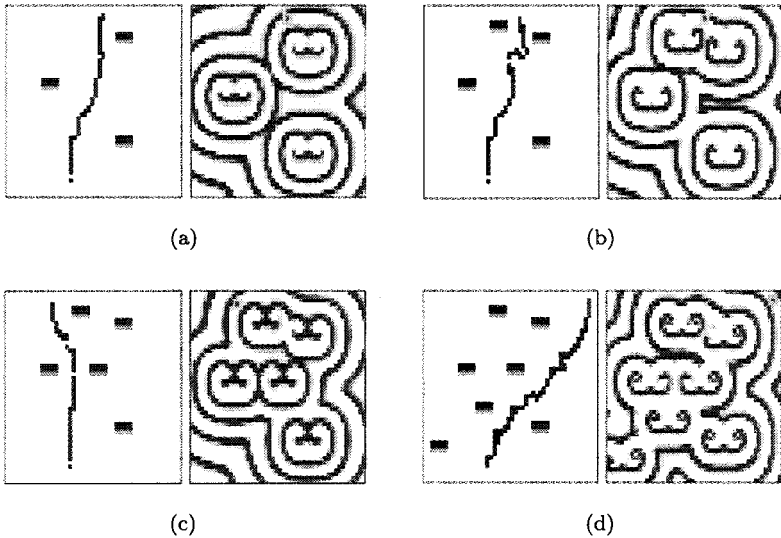


Fig. 7.9 Simulation results of wave-based navigation control: case with only obstacles. The target for the robot is to proceed in the northbound direction.

### 7.3.2 Implementation on a roving robot and experimental results

The algorithm discussed previously is quite general and can be implemented with two different installations of the camera: on the ceiling of the laboratory (as was assumed in the simulations) or on board, taking a picture of a small planar space in front of the robot. Thus, a *world-centered perception*, or a *robot-centered perception* can be implemented by using the general paradigm of RD navigation control. The *robot-centered* case is particularly appealing for the implementation of high-level analog control of autonomous robots.

The experimental set-up is provided by a real environment, where a roving robot is required to move in a pre-specified direction, in this case southwards: so no particular target positions are given. The camera is connected to a PC simulating the RD-CNN algorithm.

In the first experiment the *world-centered perception* case is reproduced. At each step the robot camera takes a picture of the environment: obstacles are used as initial conditions for the RD-CNN. When the wavefronts reach the robot, depending on the direction the wavefront comes from, the robot executes the proper command. The robot succeeds in going in the pre-

specified direction avoiding two obstacles, as can be appreciated in Fig. 7.10. Here the positions successively occupied by the robot are reported through red points. From the analysis of the Figure it emerges that the robot is indeed able to avoid both the obstacles, but it passes much closer to the one than the other. This potential problem can be explained as follows. Since autowave fronts annihilate while colliding, if the robot moves in a direction where a wavefront has just been annihilated, it will “sense” the subsequent front when the robot will be next to its source, i.e. the obstacle. A suitable synchronization between the robot motion and the wave generation avoids this problem.

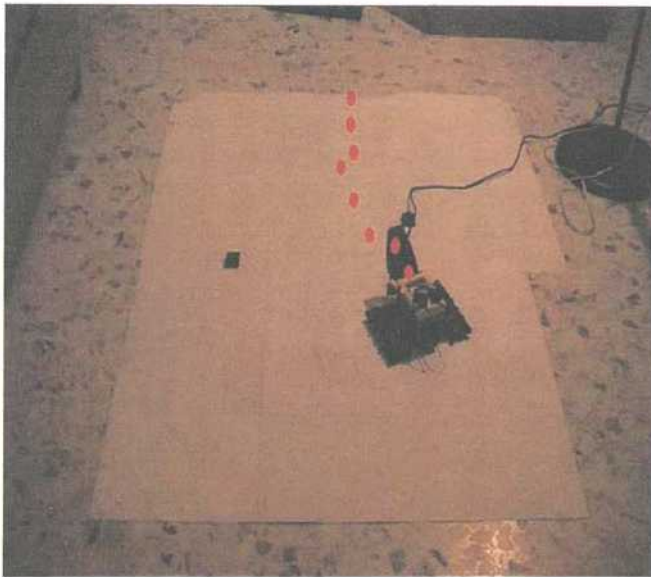


Fig. 7.10 Example of a typical trajectory of the robot while avoiding an obstacle. The camera is placed on the ceiling of the laboratory.

The second experiment refers to the *robot-centered perception* strategy: the camera is placed on the robot via a vertical pole whose height is about 1m. The camera is positioned in such a way that the focal plane is almost parallel to the ground and the robot is situated in the central bottom position within the frames. In this condition the frames refer to the environment in the forward motion direction of the robot. The whole experiment

is illustrated in the snapshots in Fig. 7.11. As can be noticed, the robot is able to avoid both the obstacles, while proceeding in the pre-specified direction.

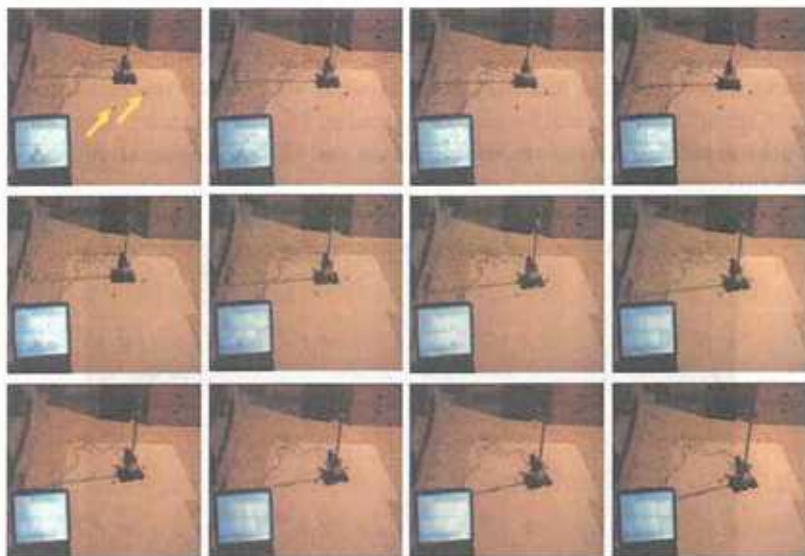


Fig. 7.11 Example of robot-centered navigation control based on RD-CNN. Several snapshots of a video are shown. The obstacle position is indicated in the first frame by two arrows.

This example shows the advantages of the analog approach. With respect to traditional algorithms for the path planning, for instance the potential field approach, where the core is calculation, for each environment state, of the potential function for each point in the robot arena, the analog approach based on wave computing relies on massively parallel computing. Each element of the excitable medium acts as a very primitive computing device. The medium's sites take on ideally continuous values and the site interacts locally; therefore the path planning problem is solved through an analog and massively parallel computation, also allowing for a real-time control of the robot's trajectory. In the case of moving obstacles or dynamically changing situations, the advantages of the approach based on an analog processor are apparent: in our case each obstacle/target is a wave source independently on its position. Another advantage of the approach is

the adaptability of the parallel computation to a vast class of optimization problems.

The interested reader is referred to [Adamatzky *et. al.* (2004)], where experimental results carried out using a VLSI CNN Chip implementing wave-based navigation control are detailed.

A final remark regards the consideration that as the same model for the RD-CNN (7.2) was also successfully used as the basic neuron model for bio-inspired locomotion control in Chap. 2, it derives that the class of CNN discussed in this Chapter represents a unifying paradigm for the solution of locomotion control, from the low level of the motion pattern generation, to the very high level of the trajectory planning.

This last Chapter close the loop: CNN-based control systems allow to establish a new paradigm for bio-inspired robots from low level to supervised control. The reported studies, even if establish new well-posed control concepts, open a new way towards innovation robot control strategies.

This page intentionally left blank



## Chapter 8

# Conclusions

Walking robots offer a large spectrum of potential advantages over wheeled structures, due to their superior adaptability capabilities on rough terrain and in environments with obstacles, but their control is a challenge because they have a large number of interconnected degrees of freedom. On the other hand, in nature there exist numberless examples of walking creatures which move on unstructured terrain in an efficient and elegant way. Therefore, many robot designers look to biological inspiration to build efficient walking robots. Two key points emerge from the study of the neural control of locomotion in many living creatures:

- the motor system is hierarchically organized;
- a network of neurons, called CPG, is able to generate the rhythm of locomotion independently of sensory feedback and signals from higher centres.

The problem of locomotion control has already been dealt with using bio-inspired approaches in many works. The approaches and the degrees of inspiration in these works vary. In some cases [Arkin (1998); Brooks (1991)], the biological inspiration is not a key point and the animal is often considered as a term of comparison to define what Artificial Intelligence means and to build software architectures able to show complex (or intelligent) behavior.

A very impressive synergy between biology and engineering is, instead, represented by the work of Quinn (for a review see [Webb and Consi (2001)]), where biological data are used for a very accurate design of a bio-mimetic robot with cockroach kinematics. However, the control of the architecture is performed using a traditional approach, while greater effort is directed towards the study of an actuation system that can achieve

performance similar to that of the cockroach.

Our work focuses mainly on locomotion control. This has been treated in several papers [Golubtisky *et. al.* (1998); Waldron and Song (1989); Lewis *et. al.* (1998); Saranli *et. al.* (2000)]. A number of engineering projects for walking machines consider a hierarchical organization of the control systems [Klaassen *et. al.* (2002); Brooks (1991); Zielinska and Heng (2002)] and some exploit the CPG paradigm to generate a locomotion control algorithm.

CPG approaches are discussed in [Chiel *et. al.* (1998); Beer *et. al.* (1992a); Beer *et. al.* (1992b); Kodjabachian and Meyer (1998)]. However, these works stop at the software level, whereas bio-inspired robotics [Webb (2001)] highlights the importance of the implementation on the robot because of the role of the environment and mechanics in the control. The importance of the mechanical system in bio-inspired control is almost universally recognized [Raibert and Hodgins (1993); Chiel and Beer (1997); Kubow and Full (1999)]. For instance in [Kubow and Full (1999)] the role of the mechanical characteristics of the animal's (or robot's) body is investigated and shown to be useful to stabilize locomotion. However, this is a quite open field and still requires further investigation.

Another interesting example of synergy between engineering and biology is given in [Ayers *et. al.* (1998)]. The lobster-robot makes use of nitinol-based actuators and biomimetic sensors. Behavioral modules observed in real lobsters are implemented in the robot. However, the CPG is a finite-state machine and implemented on a small board computer.

Among CPG-based controllers, [Lewis *et. al.* (1998)] focuses on a system suitable for hardware implementation and, at the same time, presents a strong degree of biological inspiration. It relies on coupled integrate-and-fire neurons and is able to control gait generation for a pair of legs. This CPG for biped locomotion is not easily extendable to control other robotic systems.

On the contrary, in [Golubtisky *et. al.* (1998)] and in many other related works [Collins and Stewart (1993a); Collins and Stewart (1994)] the focus is on providing a general framework for modelling CPGs, but implementation issues are not dealt with. These studies show how dynamical systems constitute a "natural" way to model the CPG. Our work on CPG is related to these works sharing with them the use of dynamic nonlinear units to model CPGs, but the focus here is more on implementation issues and therefore important topics such as the inclusion of sensory feedback and reflex implementation have been treated. These are fundamental to achieve

good performance on rough terrain, when legged machines are more effective than wheeled robots. Moreover, the so-called group-theoretic approach introduced in these papers can lead to some drawbacks when hardware constraints are considered. For instance the switching among locomotion patterns is there addressed by exploiting a bifurcation which involves the parameters of the CPG cells. This presents some undesirable aspects. First, if (as in our case) it is assumed that the dynamics of the cell directly controls the movements of a leg, one would avoid to change it when the locomotion pattern changes (indeed switching among locomotion patterns more often affects the re-organization only in the duration of the stance phase). Moreover, the hardware realization of this scheme requires the possibility to change these parameters, whereas a mechanism involving a change of the connections among cells with fixed parameters seems of more immediate feasibility.

These ideas and objectives also directed our work on decentralized locomotion. This approach to locomotion control relies on very different hypotheses. The suitability of this scheme as originally formulated by Cruse is shown in [Pfeiffer *et. al.* (2000)]. The locomotion control is totally based on reflexes. These regulates both the alternating of swing and stance and the coordination of all the legs. Of course here sensory feedback plays a fundamental role. The decentralized locomotion control introduced in this book makes use of dynamical units to build the control system and on an optimization strategy to find the parameters of the whole structure.

Evolutionary based approaches have also been considered in [Gallagher *et. al.* (1996); Chiel *et. al.* (1998); Beer *et. al.* (1992a); Kodjabachian and Meyer (1998)]. For instance in [Beer *et. al.* (1992a)] a control system based on a Hopfield neural network is evolved to find suitable parameters to make a hexapod agent able to walk. There is an important difference between artificial neural networks and CNNs. CNNs are locally coupled, therefore tailor-made for VLSI implementation, and some chips consisting of large arrays of CNN cells (64x64 or 128x128) have already been devised (for a deeper discussion the reader is referred to the literature on CNNs [Chua and Roska (1993); Manganaro *et. al.* (1999)]). However, these controllers are CPG-based schemes. In any case the use of evolved networks is not massive in this work: the structure of the network controlling the leg movements is kept constant and only the parameters of the reflex-based connections are chosen according to a genetic algorithm strategy.

The approach takes as a starting point the biological principles of local influences as in [Cruse (1990)], whose parameters have to be re-adapted

to the leg controller dynamics. Two distinct realizations have been discussed. The first shows that CNNs are suitable for both implementing central (CPG-based) control and decentralized locomotion control. The second, based on integrate-and-fire neurons, extends the application of networks of spiking neurons [Lewis *et. al.* (1998)] to hexapod walkers.

However, the main focus of this work is on a CPG scheme based on nonlinear coupled circuits (CNNs) to control the locomotion of bio-inspired robots. The project starts from the design of the CPG network. This is a not trivial task, since few results are available regarding analysis of the behavior of coupled nonlinear systems and few of them deal with design issues. The design of the CNN-based CPG was accomplished by further exploiting the analogy with the biological case: excitatory and inhibitory synapses connect the motor-neurons (dynamical second-order nonlinear systems). Thus the low level of control of the locomotion system is performed by position control of the joints of the articulations, while the CPG level is performed by the whole network of interconnected second-order circuits. At this level self-organization plays a fundamental role: the dynamical units of the CPG properly synchronize to control the locomotion of the robot. Thus the result of self-organization is that all the neurons oscillate at the same frequency despite parameter variations (a very important point for the hardware implementation) and the phase lags between them are those required to achieve interlimb coordination for proper walking. Moreover, the approach based on CPG implemented by CNN is modular and general as the control of the bio-inspired swimming robot demonstrated.

Even if the CPG is able to generate the rhythmic locomotion pattern by itself, sensory feedback and higher level control are fundamental for efficient locomotion control. Sensory feedback has been thoroughly investigated and several strategies taken into account, leading to suitable low-level direction control, feedback from ground contact sensors, and the implementation of reflexes and speed control.

An important advantage of the CNN-based approach to CPG implementation is its immediate hardware transposition: a VLSI implementation of the CNN-based CPG has been introduced. The experimental results on the first chip prototype confirm the suitability of the approach and open up the way to low-cost implementation of autonomous walking machines.

Moreover, the dynamical approach also allows the implementation of a locomotion control scheme based only on reflexes. This scheme, in contrast with the CPG, has been proposed by Cruse [Cruse (1990)] and is particularly appealing for robotic applications.

As concerns the high-level of locomotion control, the important issue of posture control has been investigated. A scheme based on distributed control has been introduced. Moreover, the idea of learning the right posture in real time has also been considered by means of the paradigm of Motor Maps, self-organizing maps with reinforcement learning.

Starting from this application, Motor Maps have been shown to be a suitable paradigm for implementing learning capabilities in the robot control system. An experiment in which the CPG starts from an unorganized condition and learns how to synchronize the legs in order to achieve a pattern of coordinated is an example of the emergence of an adaptive behavior (the locomotion gait) from local rules and reinforcement learning.

Further aspects of high-level control related to navigation issues have also been dealt with. The unifying strategy was to use a network of locally connected dynamical systems (namely a CNN) for this task as well. This is at the same time biologically inspired and simple to implement since a number of VLSI chips consisting of many cells already exist. In particular, two well-known phenomena arising in Reaction-Diffusion systems (Turing patterns and autowaves) have been used to implement analog approaches for navigation control.

The distributed control introduced in this work allows the solution of a hard task like the locomotion control by exploiting the self-organizing properties of the network of nonlinear circuits. The idea of distributed control is investigated in many other works. Among them the work of Quinn and colleagues [Quinn *et. al.* (1998); Espenschied *et. al.* (1996); Beer *et. al.* (1992b)] shows the effectiveness of the approach. In our work distributed control is in effect achieved by an array of analog nonlinear circuits, in which complex behavior (either at the low level of gait generation or the high level of posture control or obstacle avoidance planning) arises from the interaction between these simple nonlinear elements. This allows an efficient hardware implementation that intrinsically includes the possibility of obtaining adaptive behavior as an emerging property of the nonlinear system. The approach presents the further advantage of robustness to faults in parts of the control system. As concerns the gait generation for instance, if a neuron in the network is not working (as shown in an example reported in Appendix A), the robot is still able to move though not in a well coordinated way as in the undamaged case.

Moreover, although the main objective of this work is to provide methods to build up efficient control systems for the locomotion of bio-inspired robots, the analog approach allows us to gain further insights into the bi-

ological model. These considerations only hold if a degree of accuracy is guaranteed by the hardware implementation of the proposed strategies. This idea has driven all the work discussed in this book in which several robot prototypes have been presented.

These prototypes have been built up bearing in mind the intent of gradually presenting the new ideas regarding locomotion control and introducing them as modules (or bricks) for the design of the complete, fully autonomous robot. So direction control, posture stabilization, VLSI implementation of the CNN-based CPG, analog navigation control can all be embedded in an autonomous walking robot with adaptive capabilities.

## Appendix A

# HexaDyn and CNNTLab: two tools for bio-inspired locomotion control

Even though the final goal of bio-inspired control is to perform experiments on robots and to build neuromorphic chips, software tools and general frameworks to validate the approach and the control scheme have been developed in order to conceive a development system as a new tool to design bio-inspired robots. In this appendix a brief description of a dynamic simulator of the hexapod robot, called *HexaDyn*, and a general framework, called *CNNTLab*, in which the bio-inspired control system is implemented via software on a PC driving the real robot through the parallel port, is given.

### A.1 HexaDyn, a simulator for the hexapod robot

HexaDyn was developed starting from the libraries of DynaMechs, a software package in C++ including efficient dynamic simulation algorithms for multiple chain robotic systems, available as an open source [McMillan (1994); McMillan *et. al.* (1996)]. Real-time simulation and a realistic 3-D graphical display are the great advantages of this software package. On the other hand the package, being distributed as an open source, is not user-friendly. The main classes of DynaMechs were used to build an efficient framework in which to simulate the bio-inspired hexapod robot. The classes of DynaMechs were been used to design and simulate the mechanical part of the system, while new classes have been written for the bio-inspired control. The latter include classes for the simulation of the CNN-based CPG, the Motor Maps and many of the other control schemes discussed in this book.

HexaDyn allows the possibility of controlling a number of parameters. Before executing the main program running the simulation, all the param-

eters of the simulation are defined in a file. This file allows us to define the environment, the robot structure and the terrain characteristics. The environment can be specified by choosing the gravity field, fluid density, ground friction coefficient (static and kinetic), the spring and damping constant of the foot-ground contact and so on. The terrain is defined by a triangular grid with a given grid-step and by a matrix whose entries are the height values at the grid points. A library of terrains is included in HexaDyn.

The structure of the hexapod robot, simulated in HexaDyn, is hierarchically defined starting from the body and with the articulation consisting of three links. The articulations and links are defined by the Modified Denavit-Hartenberg parameters [Siciliano and Sciavicco (1989)]. As regards the joint control, a PD controller is implemented to model the servomechanisms used in the robot. Different hexapod configurations can be investigated allowing us to compare the advantages and drawbacks of the various models, as well as the differences in the control required by the different structures. Figure A.1 shows three examples: a stick-insect like configuration (the one used for the robot), a spider-like configuration and a cockroach-like configuration.

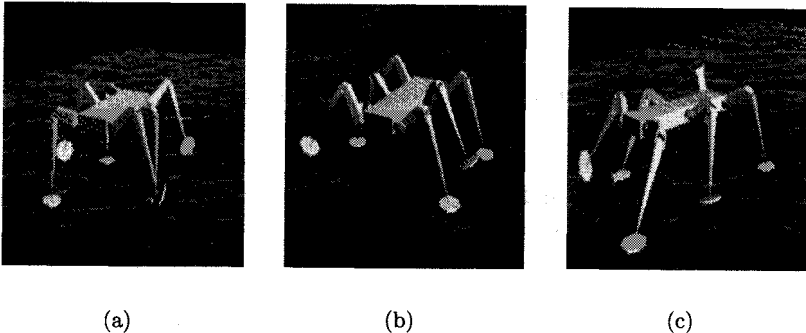


Fig. A.1 Three possible configurations of the HexaDyn hexapod: (a) stick insect-like; (b) spider-like; (c) cockroach-like.

All the variables of the system (both mechanical and electrical) can be tracked and used in the feedback loop (as, for example, in the case of attitude control through Motor Maps).

An example of a simulation using HexaDyn is shown in order to illustrate one of the advantages of the distributed approach. The distributed control is robust to faults of parts in the control system. As regards gait



generation, for instance, this robustness is illustrated in Fig. A.2, where two simulation results for the dynamic robot model HexaDyn walking on rough terrain are compared. The first refers to an “intact” CPG, while in the second case one neuron (R1) is not working (its state variables are kept constant at their initial values). Leg R1 does not move properly, but HexaDyn is still able to locomote. Of course, compared to the case in which the fast gait CPG is working correctly, the locomotion is poor and the robot deviates quite far from the direction of the motion (axis  $y$ ).

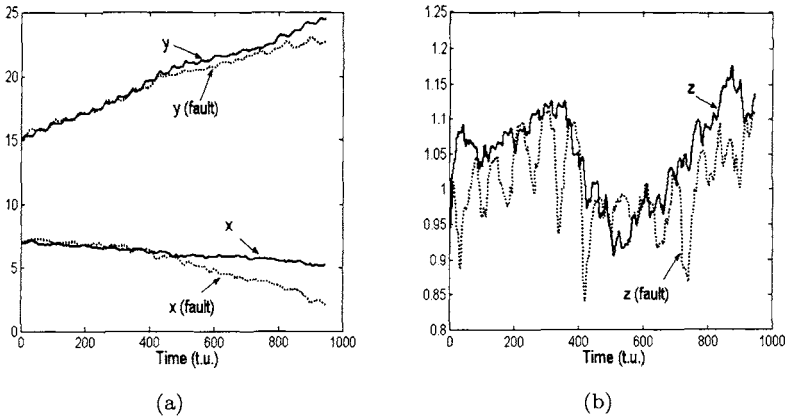


Fig. A.2 Comparison between the case in which one neuron (R1) of the CPG for fast gait is not working and that in which the CPG is working normally. Trends of the position of the body center of mass: (a) variables  $x$  and  $y$  ( $y$  is the direction of motion); (b) variable  $z$ . In the motion direction (axis  $y$ ) the discontinuous trajectory is due to the stepping movements of HexaDyn.

## A.2 CNNTLab

The purposes of CNNTLab are to provide a general framework to test bio-inspired control on the hexapod robot. The hexapod robot, whose structure is that shown in Sec. 5.3, is equipped with sensors, servomotors and a board connected to the LPT of a PC; communication between the robot and the PC takes place through an I<sup>2</sup>C bus. Moreover, the board carried on the hexapod robot includes a Pulse Width Modulator (PWM) block, able to generate the PWM signals driving up to 20 servomotors (this is achieved by a pre-programmed PIC, called SD20 [roboelectronics webpage]) and

analog to digital (A/D) converters. This allows us to consider any kind of sensor and motor, provided small changes are made in the hardware of the robot. A block scheme of the framework is given in Fig. A.3.

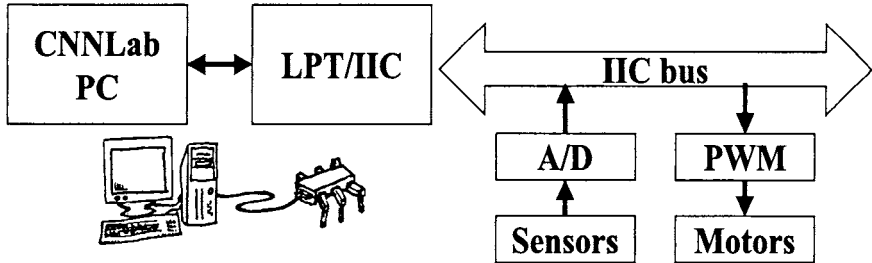


Fig. A.3 Scheme of the CNNLab framework.

The bio-inspired control is implemented via software on the PC. CNNLab allows control and visualization of all the parameters of the system in a user-friendly interface. For instance, the integration step size of the control algorithm, the time step of time-multiplexing for generating the motor outputs, all the scaling factors between CNN outputs and motor signals, the frequency of the PWM signals and so on can be fixed. Typically these are chosen to set the stepping frequency to suitable values.

CNNLab provides great flexibility in the control algorithm by using an intuitively simple script language. The CNN-based CPG is therefore defined by this script. Moreover, CNNLab already implements the attitude control described in Sec. 6.1. This is an example of the feedback loops between the hexapod robot and the PC that can be implemented in CNNLab. Moreover, the sensor outputs can be stored and visualized in real time. Hence, implementing the decentralized locomotion control outlined in Chap. 4 is also possible. To this end and to visualize a gait diagram of the robot while walking, six opto-mechanical sensors detecting the ground contact of each leg are already carried on the robot.

An example of the graphic user interface is shown in Fig. A.4, while Fig. A.5 is a photograph of the robot controlled by CNNLab.

Another possibility is to tele-operate the robot instead of using a direct connection. In this case the whole control system is devoted to the tele-operated control of the hexapod robot and is composed of two different parts: a PC hosts the CNN-based CPG, the control feedback laws and allows the storage of all the measures, while the circuitry carried on the

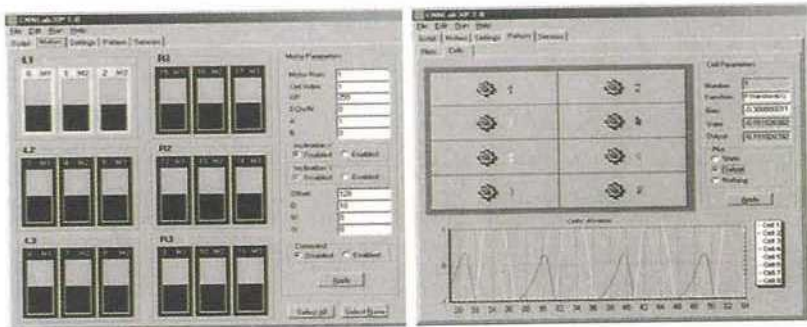


Fig. A.4 An example of the graphic user interface of CNLab.



Fig. A.5 Photographs of the robot controlled by CNLab.

robot deals only with the joint motor control and the measurement system. The communication between PC and robot circuitry occurs through two AUREL XTR 434 modules, each one driven by a PIC and with a transmission protocol *ad hoc* written. Figure A.6 shows a block scheme of the whole system.

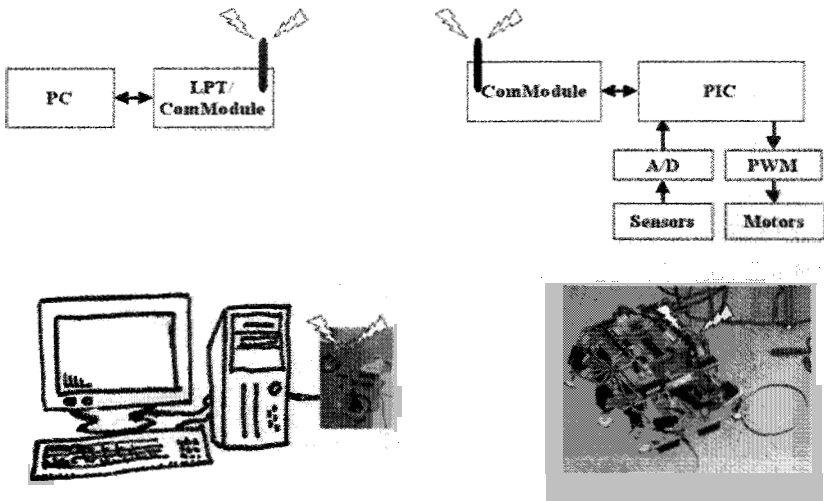


Fig. A.6 Scheme of the CNLab framework when used for tele-operated control.

## Appendix B

# Design of the CNN circuit

The design of the CNN circuit implementing the CNN motor-neuron (2.5) is now introduced. The circuit is based on operational amplifiers and RC components. Starting from this circuit and applying standard techniques for switched-capacitor design [Gregorian and Temes (1986); Johns and Martin (1997)] the switched-capacitor circuit discussed in Sec. 3.5 can easily be derived.

Equations (2.5) are dimensionless. We first consider a model suitable for the circuit implementation. The equations of the CPG cell including the time-scaling factor are the following:

$$\begin{cases} \frac{dx_1}{dt} = \frac{1}{\tau}(-x_1 + (1 + \mu)y_1 - sy_2 + i_1) \\ \frac{dx_2}{dt} = \frac{1}{\tau}(-x_2 + sy_1 + (1 + \mu)y_2 + i_2) \end{cases} \quad (\text{B.1})$$

with

$$y_i = \frac{1}{2}(|x_i + 1| - |x_i - 1|) \quad i = \{1, 2\} \quad (\text{B.2})$$

and  $\tau = RC$ . In this model the state variables are represented by voltages.

The core of our circuit is the Miller integrator/adder block [Smith and Sedra (1998)] shown in Fig. B.1. The equation it implements is the following:

$$\frac{dx}{dt} = \frac{1}{R_1 C_1} \left( -x - \frac{R_1}{R_2} y_i - \frac{R_1}{R_3} y_j \right) \quad (\text{B.3})$$

where  $x$  is the voltage across the capacitor  $C_1$ . Equation (B.3) perfectly matches that of a CNN first-order cell.  $y_i$  and  $y_j$  represent the outputs of two generic cells. The piece-wise linear output (B.2) is obtained by considering an inverting block and exploiting the saturation of the operational

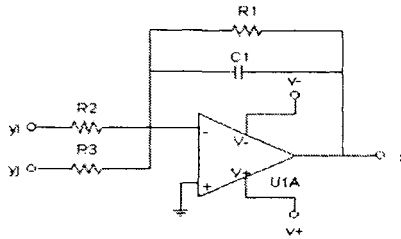


Fig. B.1 The Miller integrator is the core of the CNN circuit.

amplifier (i.e. choosing the gain to saturate the output). This approach leads to saturation points depending on the voltage supply values. These saturations are then scaled to the standard values  $\pm 1$ . Therefore the values of the resistances of the CNN circuit depend on the voltage supply. The whole CNN implementing the CNN neuron is shown in Fig. B.2.

In terms of circuit components the neuron parameters are expressed by the following relations:

$$\begin{aligned}
 \tau &= R_1 C_1 = R_2 C_2 \\
 \mu &= \frac{R_1 R_{14}}{R_8 R_{13}} - 1 = \frac{R_2 R_{16}}{R_{12} R_{15}} - 1 \\
 s &= \frac{R_1 R_{16}}{R_9 R_{15}} = \frac{R_2 R_{14}}{R_{11} R_{13}} \\
 i_1 &= \frac{R_4 R_1}{(R_3 + R_4) R_7} V_{cc} \\
 i_2 &= \frac{R_6 R_2}{(R_5 + R_6) R_{10}} (-V_{cc})
 \end{aligned} \tag{B.4}$$

The values of the CNN circuit components with  $V_{cc} = \pm 2.5V$  and the parameters in Table 2.1 are shown in Table B.1.

Table B.1 Values of the components of the CNN circuit.

$R_1, R_2$	$1M\Omega$
$R_3, R_5$	$1M\Omega$
$R_4, R_6$	$0.14M\Omega$
$R_7, R_{10}$	$1M\Omega$
$R_8, R_{12}$	$1.67M\Omega$
$R_9, R_{11}$	$2.5M\Omega$
$R_{13}, R_{15}$	$1M\Omega$
$R_{14}, R_{16}$	$2.5M\Omega$
$R_{17}, R_{18}$	$1M\Omega$

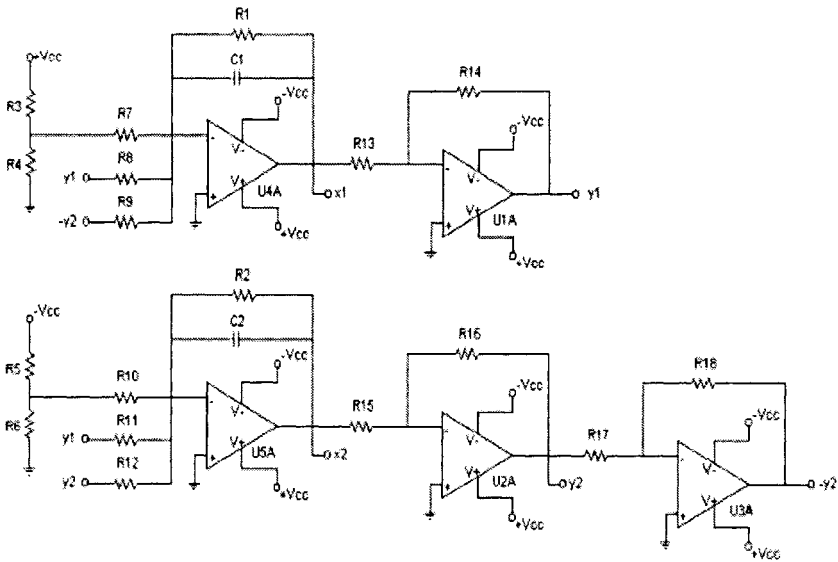


Fig. B.2 The circuit implementing the CNN motor-neuron (B.4).

In order to design the circuit implementing the whole CNN-based CPG, the circuit implementing the CNN chemical synapse of Eq. (2.13) has to be discussed. This design can be addressed in a very simple way. In Fig. B.3 the electrical scheme of an excitatory synapse from the CNN neuron a to the CNN neuron b is reported. This has been implemented by an inverting operational amplifier whose input is the signal  $y_{1,a}$  from the CNN neuron a. The synaptic input of the CNN neuron circuit is realized by considering a further input in the block of Fig. B.1. The value of the synaptic weight  $\varepsilon$  can be set according to the following equation:

$$\varepsilon = \frac{R_{21}}{R_{20}}. \tag{B.5}$$

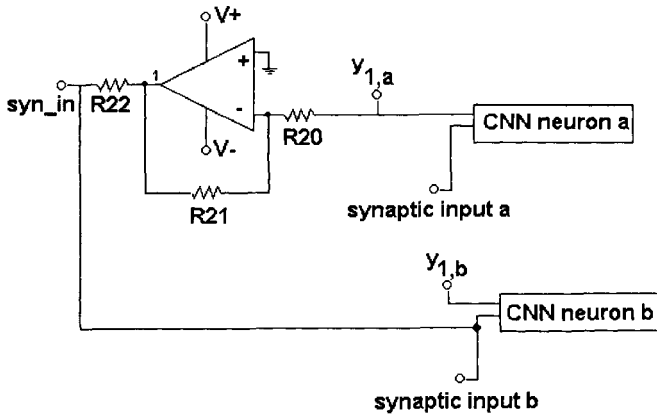


Fig. B.3 The circuit implementing the CNN chemical synapse. Each block labelled as CNN neuron represents a circuit of Fig. B.1.



## Appendix C

# A chaos-based sensor for bio-inspired robots

Autonomous robots often need a large number of sensors to span the space around them. A very common approach is the use of sonar sensors. However, complex algorithms are often required to handle sonar sensors in multipath fading and in a multi-user scenario. A bio-inspired approach to this problem is, on the other hand, appealing because of the fascinating examples (bats) present in nature. The solution proposed here exploits the principles underlying Chaotic Pulse Position Modulation (CPPM) in order to improve the efficiency of a sonar system in multipath fading situations and in a multi-user scenario.

### C.1 Continuous CPPM

Collision avoidance is one of the main issues involved when dealing with the design and construction of autonomous robots. Truly autonomous robot control often implies the ability of free-roaming platforms to travel in non-structured environments. The intelligent control of robots involves strategies to avoid obstacles and, if necessary, choose alternative routes to accomplish a task. This activity is performed on the basis of the information collected by several sensors. Among the most common sensors used for collision avoidance, those based on the measurement of the *Time of Flight* (TOF) are perhaps the most common. In particular, *sonars*, based on *Ultrasonic TOF ranging* allow reliability and precision of measurements to be conjugated with low costs and ease of interface, making these sensors widely used tools for autonomous robot navigation [Everett (1995)]. Autonomous robots are often equipped with a large number of ultrasound sensors to span the space around them and sense the presence of obstacles in their operational space. Moreover, more and more robotics applications

require more than one robot to roam in the same environment. Therefore, two main issues arise in this context: firstly, the *multi-user* scenario makes it quite hard for each sensor to distinguish its own echo; secondly, the *multipath fading* phenomenon, arising when the same sound wave comes back to the sensor along several paths, deteriorates sensor performance.

The principles of *noncoherent chaotic communications* are here used to build an effective sonar system able to perform ranging tasks in an environment affected by multipath fading and in a multi-user scenario. Chaotic communications exhibit a low probability of detection and interception, as the high sensitivity of the chaotic generator to initial conditions and to parameter values allows us to establish a great variety of *keys* in order to generate unpredictable sequences. Moreover, the advantages of chaotic communication systems lie in their high levels of performance in multipath propagation conditions and multi-user scenarios, as cited above [Kolumbán *et. al.* (1997); Kolumbán *et. al.* (1998)].

In the framework of ultra-wide bandwidth (UWB) communication systems, impulse-based schemes open up new perspectives, due to the reduced complexity involved, low power consumption, low probability of detection, insensitivity to multipath propagation, and multi-user capabilities. In particular, *Chaotic Pulse Position Modulation* (CPPM) [Maggio *et. al.* (1999)] constitutes a promising approach in the field. The main idea underlying CPPM is to generate a sequence of pulses in which the duration of the time interval between one pulse and the next one is provided by a chaotic law. As the information is in the temporal distance between pulses, additive noise on the channel does not affect the integrity of the information. Moreover, pulses with a small duty cycle are used, thus involving low power consumption.

The principles underlying CPPM are thus applied in order to design and implement a reliable sonar system, able to detect distances from targets in multipath fading situations and multi-user scenarios. The approach is based on the generation of a train of sonar pulses according to the CPPM technique. The estimate of the distance is performed by correlation-based techniques, rather than on the basis of measurement of the TOF.

In our CPPM system the chaotic sequence is generated on the basis of a continuous chaotic attractor, rather than a chaotic discrete map. This modification allows us to generate an effective chaotic sequence in an entirely analog fashion, without requiring any digital processor with a word of bits large enough to generate numbers with the required precision.

Continuous CPPM is essentially achieved via a voltage-to-time conver-

sion. The block scheme of the continuous modulator is shown in Fig. C.1. The chaotic sequence of time intervals  $\tau_k$  is generated in the following way. A sample-and-hold circuit performs a S/H operation on a single state variable of the chaotic circuit. Then, a ramp is generated and stopped when its value equals that of the sampled signal. When this occurs, a pulse is emitted from the modulator, the ramp is reset and a further S/H operation is performed. It is clear that, as the ramp signal is reset when the pulse is emitted, it reaches the value of the chaotic sample after a time which is proportional to the value itself. More details of the CPPM modulator are given in [Fortuna *et. al.* (2002)].

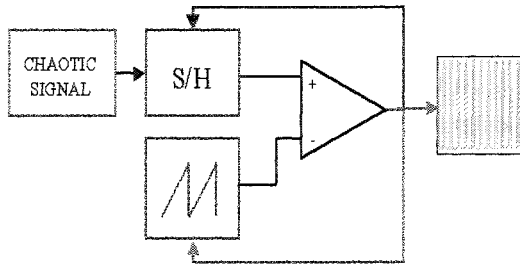


Fig. C.1 Block scheme of the continuous chaotic pulse position modulator.

The continuous circuit generating chaos is the well-known *Chua's Circuit* [Madan (1993)]. It is clear that the modulator only works with positive continuous chaotic signal values. As the state variables assume both positive and negative values in a lot of attractors coming from Chua's circuit, the introduction of a suitable offset may be required to guarantee the generation of only positive values. The continuous CPPM approach allows us to preserve the chaotic behavior as shown in [Fortuna *et. al.* (2001); Fortuna *et. al.* (2003)], in which the demodulated signal is compared with the original one.

## C.2 The CPPM Sonar

Distance measurements based on sonars are usually performed by measuring the *time of flight* of an ultrasound wave propagating in air.

The user can retrieve the distance  $d$  by measuring the time  $\bar{t}$  which has elapsed from the time at which the signal is sent to the time at which the

echo is received, by performing the simple operation:

$$d = \frac{c\bar{t}}{2} \quad (\text{C.1})$$

where  $c$  is the speed of propagation of sound in the air.

It is clear that this simple operation mode has strong drawbacks in the case of multi-user scenarios. It is worth remarking that even a single vehicle is usually equipped with more than one sensor in order to span the operational space over a large angle, which makes almost any case a multi-user scenario. Although one may think of modulating the pulse train at different frequencies, the multipath fading phenomena in a multi-user scenario can lead to relevant errors in the measurement, because of the distortion introduced.

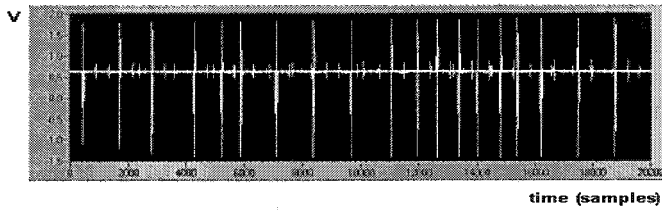
The sonar sensor (Polaroid Series 600) is driven by a continuous CPPM modulator. The output of the modulator is fed as input to the power stage of the Polaroid 6500 Ranging Module [Polaroid (1995)] in order to provide a suitable value of 400 V of amplitude to drive the sensor membrane.

Exploiting the properties of correlation of chaotic signals, the cross correlation between the transmitted and the received signal should show a relevant peak at the TOF of the sound wave. Moreover, as the cross correlation between different portions of a chaotic signals is ideally zero, multipath fading and multi-scenario problems are avoided by adopting this approach.

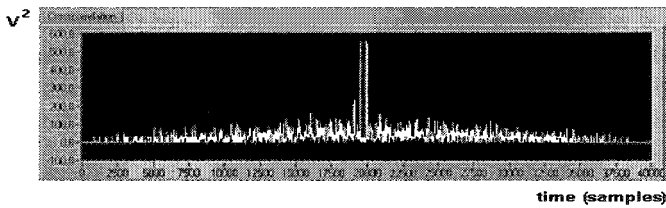
The experimental setup was built with a Polaroid series 600 sensor. The train of pulses was emitted by continuous CPPM driven by a Chua's circuit evolving according to a *double scroll Chua attractor* [Madan (1993)]. The Chua's circuit has been implemented by using CNNs [Arena *et. al.* (1995)].

Experimental results related to a single measurement experiment are given in Fig. C.2, which refers to measurement of the distance of a target placed at 75cm.

Figure C.2(a) shows the received signal in a time window lasting 200ms, which coincides with the window used for the cross correlation between the transmitted and received signals. The received signal is characterized by three components: the first is constituted by the highest peaks, corresponding to the transmitted peaks themselves, also received by the sensor, which is not inhibited during transmission. The smaller peaks are related to two sources: the first is the real reflected ultrasound wave, while the second is a reflected ultrasound wave coming from another sensor driven by CPPM relying on a further double scroll Chua attractor with different parameters. This last signal source has obviously been introduced to evaluate the per-



(a)



(b)

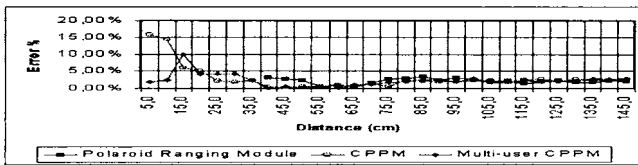
Fig. C.2 Experiment with a target at 75cm. (a) Received signal. (b) Cross correlation between the transmitted and received signals.

formance of the sensor in multipath fading conditions and in a two-user scenario. Figure C.2(b) shows the cross correlation between the transmitted and received signals. It clearly shows two large peaks. One of them is centered at sample 20000, corresponding to time origin; this refers to the fact that the transmitted signal is entirely enclosed in the received one. The other peak refers to the time delay due to the time of flight. Other smaller peaks are present, evidently due to the multipath and multi-user condition. By evaluating the distance between the two largest peaks, a value of 73.48cm is estimated.

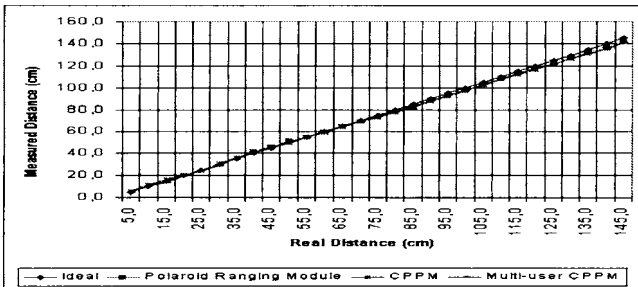
The sensor has been characterized for measurements ranging from 5cm to 145cm, in 5cm steps. Three sets of experiments were carried out. The first refers to measurements performed using the sonar driven by CPPM, in a single-user scenario. The second refers to the same experiment, in the presence of another CPPM sensor located close to the sensor to be characterized, in order to emulate the two-user scenario. The third, as a reference to evaluate the error committed, refers to measurements performed by a sensor driven by an original Polaroid 6500-Series Sonar Ranging Module. It is worth remarking that in this case, measurements of distances under 40cm require peculiar techniques to damp the echo of the transmitted sig-

nal itself which would lead to incorrect measurement. This is prevented by the constructor by introducing an interval of 238ms, when the sensor is inhibited. Figure C.3(a) and (b) gives the error committed and the distance measured, respectively, versus the real distance. Table C.1 gives the average measurement error committed in the three cases. To make a comparison, the table has been worked out considering the range 40cm–145cm. As can be noticed, the error committed in the three cases is not affected by the modification introduced in the modulation scheme. This leads us to conclude that the CPPM approach allows distance measurements to be performed with an error comparable with that obtained through traditional modulation, despite the presence of another user, thus obtaining a better overall performance. Moreover, the CPPM approach allows us to perform measurements under 40cm without adopting any particular technique to damp the echo of the transmitted signal.

	Polaroid Ranging Module	CPPM	Multi-user CPPM
Mean	2.29%	1.87%	1.84%



(a)



(b)

Fig. C.3 Characterization of the sensor from 5cm to 145cm. (a) Error committed. (b) Measurement performed.

The characterization of this type of sensor based on Chua's circuit (implemented by CNN) chaotic encryption system is a further proof of the fact that CNN techniques are used in the bio-inspired approach from low level to high level control but not only: CNNs are introduced also for sensor implementation.

This page intentionally left blank



# References

- Adamatzky A., Arena P., Basile A., Carmona-Galan R., De Lacy Costello B., Fortuna L., Frasca M., Rodriguez-Vazquez A. (2004). Reaction-diffusion navigation robot control: from chemical to VLSI analogic processors, *IEEE Trans. Circuits and Systems I*, Vol. 51, No. 5.
- Adamatzky A., Holland O., Rambidi N. G. and Winfield A. (1998). Wet artificial brains: Towards the chemical control of robot motion by reaction-diffusion and excitable media, *Lecture Notes in Computer Science*, 1674, pp. 34–313.
- Agladze K., Magome N., Aliev R., Yamaguchi T. and Yoshikawa K. (1997). Finding the optimal path with the aid of chemical wave, *Physica D*, 106 , pp. 247–254.
- Arena P., Fortuna L., Frasca M., Patané L. (2003a). Sensory Feedback in CNN-Based Central Pattern Generators, *International Journal of Neural Systems*, Vol. 13, No. 6, pp. 469–478.
- Arena P., Cruse H., Frasca M. (2003b). Cellular nonlinear network based bio-inspired decentralized control of locomotion for hexapod robots, *Adaptive Behavior*, Vol. 10, No. 2, pp. 97-111.
- Arena P., Fortuna L. and Frasca M. (2002a). Chaos control by using motor maps, *Chaos*, Vol. 12, No. 3, 559–573.
- Arena P., Fortuna L., Frasca M. (2002b). Attitude control in walking hexapod robots: an analogic spatio-temporal approach, *Int. J. Circ. Theor. Appl.*, 30, pp. 349–362.
- Arena P., Fortuna L., Frasca M. (2002c). Multi-template approach to realize central pattern generators for artificial locomotion control, *Int. J. Circ. Theor. Appl.*, 30, pp. 441–458.
- Arena P., Bonomo C., Fortuna L., Frasca M. (2002d). Electro-active polymers as CNN actuators for locomotion control, *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, Vol. 4, pp. 281–284.
- Arena P., Fortuna L. (2000). Collective behaviour in cellular neural networks to model the central pattern generator, *International Journal of Systems Science*, Vol. 31, No. 7, pp. 827–841.
- Arena P., Fortuna L., Branciforte M. (1999). Reaction-diffusion CNN algorithms to generate and control artificial locomotion, *IEEE Trans. Circuits and*

- Systems I*, Vol. 46, No. 2, pp. 253–260.
- Arena P., Branciforte M., Fortuna L. (1998). A CNN based experimental frame for patterns and autowaves, *International Journal of Circuit Theory and Applications*, 26, pp. 635–650.
- Arena P., Caponetto R., Fortuna L., Manganaro G. (1997). Cellular neural networks to explore complexity, *Soft Computing Research Journal*, 1, pp. 120–136.
- Arena P., Baglio S., Fortuna L., Manganaro G. (1995). Chua's circuit can be generated by CNN cells, *IEEE Trans. Circuit and Systems I*, Vol. 42, No. 2, pp. 123–125.
- Arkin R. C. (1998). *Behavior-Based Robotics*. MIT Press: Cambridge, MA.
- Ayers J., Zavracky P., McGruer N., Massa D., Vorus W., Mukherjee R. and Currie S. (1998). A modular behavioral-based architecture for biomimetic autonomous underwater robots, In *Proc. of the Autonomous Vehicles in Mine Countermeasures Symposium*. Naval Postgraduate School.
- Barnes D. P. (1998). Hexapodal robot locomotion over uneven terrain, In *Proc. IEEE Conf. on Control Applications*, pp. 441–445, Trieste Italy.
- Baum D. (2002). *Definitive Guide to Lego Mindstorms*, Apress.
- Beer R. D., Gallagher J. C. (1992a). Evolving dynamical neural networks for adaptive behaviour, *Adaptive Behavior*, 1, pp. 91–122.
- Beer R. D., Chiel H. J., Quinn R. D., Espenschied K. S., Larsson P. (1992b). A distributed neural network architecture for hexapod robot locomotion, *Neural Computation*, 4, pp. 356–365.
- Braitenberg V. (1984). *Vehicles: Experiments in synthetic psychology*, MIT Press.
- Brooks R. (1991). New approaches to robotics, *Science*, 253, pp. 1227–1232.
- Chiel H. J., Beer R. D., Gallagher J. C. (1998). Evolution and analysis of model cpgs for walking i. dynamical modules, *J. Computational Neuroscience*, Vol. 7, No. 2, pp. 99–118.
- Chiel H. J. and Beer R. D. (1997). The brain has a body: adaptive behavior emerges from interactions of nervous system, body and environment, *Trends Neuroscience*, 20, pp. 553–557.
- Chrachri A. and Clarac F. (1987). Induction of rhythmic activity in motoneurons of crayfish thoracic ganglia by cholinergic agonists, *Neuroscience Letters*, 77, pp. 49–54.
- Chua L. O., Roska T. (1993). The CNN paradigm, *IEEE Trans. Circuits and Systems I*, 40, pp. 147–156.
- Chua L. O., Yang L. (1988a). Cellular neural networks: theory, *IEEE Trans. Circuits and Systems I*, 35, pp. 1257–1272.
- Chua L. O., Yang L. (1988b). Cellular neural networks: applications, *IEEE Trans. Circuits and Systems I*, 35, pp. 1273–1290.
- Cohen A. H., Holmes P. J., Rand R. H. (1982). The nature of the coupling between segmental oscillators of the maprey spinal generator for locomotion: a mathematical model, *J. Math. Biol.*, 3, pp. 345–369.
- Cohen A. H., Wallen P. (1980). The neuronal correlate to locomotion in fish: Fictive swimming induced in an in-vitro preparation of the lamprey spinal cord, *Exp. Brain Res.*, 41, pp. 11–18.

- Collins J. J., Stewart I. (1994). A group-theoretic approach to rings of coupled biological oscillators, *Biol. Cybern.*, 71, pp. 95–103.
- Collins J. J., Stewart I. N. (1993a). Coupled nonlinear oscillators and the symmetries of animal gaits, *J. Nonlinear Sci.*, 3, pp. 349–392.
- Collins J. J., Stewart I. (1993b). Hexapodal gaits and coupled nonlinear oscillator models. *Biol. Cybern.*, 68, pp. 287–298.
- Cruse H., Kindermann T., Schumm M., Dean J., Schmitz J. (1998a). Walknet – a biologically inspired network to control six-legged walking, *Neural Networks*, 11, pp. 1435–1447.
- Cruse H., Dean J., Kindermann T., Schmitz J., Schumm M. (1998b). Simulation of complex movement using artificial neural networks, In *From animals to animats 6*, pp. 628–637, MIT Press.
- Cruse H., Muller-Wilm U., Dean J. (1993). Artificial neural nets for controlling a six-legged walking system, In H. Roitblat. S. Wilson J. A. Meyer, editor, *From animals to animats 2*, pp. 52–60, MIT Press.
- Cruse H. (1990). What mechanisms coordinate leg movement in walking arthropods?, *Trends in Neurosciences*, 13, pp. 15–21.
- Espenschied K. S., Quinn R. D., Beer R. D., Chiel H. J. (1996). Biologically based distributed control and local reflexes improve rough terrain locomotion in a hexapod robot, *Robotics and Autonomous Systems*, 18, pp. 59–64.
- Everett H. R. (1995) *Sensors for Mobile Robots - Theory and Application*, A K Peters Ltd., Natick, MA.
- Fortuna L., Frasca M., Rizzo A., (2003) Chaotic Pulse Position Modulation to Improve the Efficiency of Sonar Sensors, *IEEE Trans. Instrumentation and Measurement*, Vol. 52, No. 6, pp.1809–1814.
- Fortuna L., Rizzo A., Frasca M., Branciforte M., Bartolone M. (2002) A system for detecting distances using chaotic signals, *Patent No. 01830801.5-1248*.
- Fortuna L., Frasca M., Rizzo A. (2001), “Chaos preservation through continuous chaotic pulse position modulation, In *Circuits and Systems, 2001. ISCAS 2001, The 2001 IEEE International Symposium on*, Vol. 3, pp. 803–806.
- Gallagher J. C., Beer R. D., Epenschied K. S., Quinn R. D. (1996), “Application of evolved locomotion controllers to a hexapod robot, *Robotics and Autonomous Systems*, 19, pp. 95–103.
- Goldberg D. E. (1989) *Genetic Algorithm in Search Optimization and Machine Learning*, Addison Wesley.
- Golubitsky M., Stewart I., Buono P., Collins J. J. (1998). A modular network for legged locomotion, *Physica D*, 115, pp. 56–72.
- Golubitsky M., Stewart I. N. (1985). Hopf bifurcation in the presence of symmetry, *Arch. Rational Mech. Anal.*, Vol. 87, No. 2, pp. 107–165.
- Goraş L., Chua L. O., Leenaerts D. M. W. (1995). Turing pattern in CNNs - Part I: Once over lightly, *IEEE Trans. Circuits and Systems I*, Vol. 42, No. 10, pp. 602–611.
- Goraş L., Chua L. O. (1995). Turing pattern in CNNs - Part II: Equations and behaviors, *IEEE Trans. Circuits and Systems I*, Vol. 42, No.10, pp. 612–626.
- Goraş L., Chua L. O., Pivka L. (1995b). Turing pattern in CNNs - Part III: Computer simulation results, *IEEE Trans. Circuits and Systems I*, Vol. 42,

- No. 10, pp. 627–637.
- Graham D. (1985). Pattern and control of walking in insects, *Advances in Insect Physiology*, 18, pp. 31–140.
- Greenwald L. and Koppena J. (2003). Mobile Robot Labs, *IEEE Robotics and Automation Magazine*, Vol. 10, No. 2, pp. 25–32.
- Gregorian R., Temes G. C. (1986), *Analog MOS Integrated Circuit For Signal Processing*. Wiley-Interscience.
- Grillner S., Wallen P., Brodin L., Lansner A. (1991). Neuronal networks generating locomotor behavior in lamprey, *Ann. Rev. Neurosci.*, Vol. 14, pp.169–200.
- Heagy J. F., Carroll T. L., Pecora L. M. (1994). Synchronous chaos in coupled oscillator systems, *Physical Review E*, Vol. 50, No. 3, pp. 1874–1884.
- Hindmarch J. L. and Rose R. M. (1984). A model of neuronal bursting using three coupled first order differential equations, *Proc. R. Soc. Lond.*, 221, pp. 87–102.
- Horiuchi T., Bair W., Bishofberger B., Lazzaro J., Koch C. (1992). Computing motion using analog vlsi chips: an experimental comparison among different approaches, *Int. J. Comp. Vision*, 8, pp. 203–216.
- Huerta R., Varona P., Rabinovich M. I., Abarbanel H. D. I. (2001). Topology selection by chaotic neurons of a pyloric central pattern generator, *Biol. Cybern.*, 84, pp. L1–L8.
- Johns D. A., Martin K. (1997), *Analog Integrated Circuit Design*. John Wiley & Sons.
- Klaassen B., Linnemann R., Spennenberg D., Kirchner F. (2001). Biomimetic walking robot scorpion: Control and modeling, *Robotics and Autonomous Systems*, Vol. 41, No. 2-3, pp. 69–76.
- Klassner F. and Anderson S. D. (2003). LEGO Mindstorms: Not Just for K-12 Anymore, *IEEE Robotics and Automation Magazine*, Vol. 10, No. 2, pp. 12–18.
- Kirchner F. (1997). Q-learning of complex behaviours on a six-legged walking machine. *Second Euromicro Workshop on Advanced Mobile Robots (EUROBOT '97)*, October 22 - 24, 1997 Brescia, ITALY, pp. 51–58.
- Kodjabachian J., Meyer J.-A. (1998). Evolution and development of neural controllers for locomotion, gradient-following, and obstacle-avoidance in artificial insects, *IEEE Trans. Neural Networks*, Vol. 9, No. 5, pp. 796–812.
- Kohonen T. (1989) *Self-Organization and Associative Memory*, Springer-Verlag.
- Kohonen T. (1972). Self-organized formation of topologically correct feature maps, *Bio. Cybern.*, 43, pp. 59–69.
- Kolumbán G., Kennedy M. P., Chua L. O. (1998). The role of synchronization in digital communications using chaos—Part II: Chaotic modulation and chaotic synchronization, *IEEE Trans. Circuits and Systems I*, 45, pp. 1129–1140.
- Kolumbán G., Kennedy M. P., Chua L. O. (1997). The role of synchronization in digital communications using chaos—Part I: Fundamentals of digital communications, *IEEE Trans. Circuits and Systems I*, 44, pp. 927–936.
- Kreutz-Delgado K., Wen J. TY. (1991). The attitude control problem, *IEEE*

- Trans. Automatic Control*, Vol. 36, No. 10, pp. 1148–1162.
- Krinsky V. I. (editor) (1984), *Self-Organization: Autowaves and Structures Far from Equilibrium*, Springer-Verlag, Berlin.
- Kubow T. M., Full R. J. (1999). The role of the mechanical system in control: a hypothesis of self-stabilization in hexapedal runners, *Phil. Trans. R. Soc. Lond. B*, 354, pp. 849–861.
- Kuznetsov Y. (1998) *Elements of Applied Bifurcation Theory*, Springer.
- Lewis M. A., Etienne-Cummings R., Cohen A. H., Hartmann M. (1998). Toward biomorphic control using custom avlsi cpg chips, In *International Conference on Robotics and Automation*, San Francisco, April 1998.
- Madan R. N. (1993) *Chua's circuit: a paradigm for chaos*, World Scientific Series on Nonlinear Sciences, Series B, Vol. 1 (World Scientific, Singapore).
- Maggio G. M., Rulkov N., Sushchik M., Tsimring L., Volkovskii A. (1999). Chaotic pulse-position modulation for ultrawide-band communication systems, In H. Abarbanel L. Larson and K. Yao, editors, *Proc. UWB'99*, Washington DC.
- Manganaro G., Arena P., Fortuna L. (1999). *Cellular Neural Networks: Chaos, Complexity and VLSI Processing*, Springer-Verlag.
- Marder E. (2001). Moving rhythms, *Nature*, 410, pp. 755.
- McMillan S., Orin D. E., McGhee R. B. (1996). A computational framework for simulation of under water robotic vehicle systems, *Journal of Autonomous Robots, Special Issue on Autonomous Underwater Robots*, 3, pp. 253–268.
- McMillan S. (1994) *Computational Dynamics for Robotic System on Land and Under Water*, Ph. D. Thesis The Ohio State University Columbus OH.
- Murray J. D. (1993) *Mathematical biology*, Springer-Verlag.
- Nossek J. A. (1994). Design and learning with cellular neural networks, In *Proc. of IEEE Workshop on CNNs and their Apps '94*, pp. 137–146.
- Nossek J. A., Magnussen H., Nachbar P., Schuler A. J. (1993). Learning algorithms for cellular neural networks, In *Proc. of NOLTA '93*, pp. 11–16.
- Orlovsky G. N., Deliagina T. G., Grillner S. (1999) *Neural Control of Locomotion*, Oxford Press.
- Park J. H., Chung H. (1999). ZMP compensation by on-line trajectory generation for biped robots, In *Proc. of IEEE Conference on Systems, Man and Cybernetics (SMC'99)*, pages 960–965 (IV), Tokyo Japan, October 1999.
- Pearson K. G., Franklin R. (1984). Characteristics of leg movements and patterns of coordination in locusts walking on rough terrain, *International Journal Robotics Research*, Vol. 3, No. 2, pp. 101–112.
- Pearson K. G. (1976), “The control of walking, *Sci. Am.*, Vol. 235, No. 6, pp. 72–86.
- Pearson K. G., Iles J. F. (1971). Innervation of coxal depressor muscle in the cockroach, *periplaneta americana*, *Journal of Experimental Biology*, Vol. 52, No. 1, pp. 139–165.
- Pfeiffer F., Lffler K., Gienger M. (2000). Design aspects of walking machines, In *Proc. of the 3rd International Conference on Climbing and Walking Robots (CLAWAR)*, pp. 17–38.
- Pfeiffer F., Eltze J., Weidemann H.-J. (1995). Six-legged technical walking con-

- sidering biological principles, *Robotics and Autonomous Systems*, 14, pp. 223–232.
- Polaroid (1995) *6500-Series Sonar Ranging Module*. Product Specifications PID 615077, Polaroid Corporation, Cambridge, MA.
- Pikovsky A., Rosenblum M., Kurths J. (2001), *Synchronization: A universal concept in nonlinear sciences*, Cambridge Nonlinear Science Series 12.
- Quinn R. D., Ritzmann R. E. (1998). Construction of a hexapod robot with cockroach kinematics benefits both robotics and biology, *Connection Science*, Vol. 10, No. 3-4, pp. 239–254 .
- Rabaey J. M. (1996) *Digital Integrated Circuits*. Prentice-Hall.
- Rabinovich M. I., Abarbanel H.D.I., Huerta R., Elson R., Selverston A.I (1997), “Self-regularization of chaos in neural systems: Experimental and theoretical results, *IEEE Trans. Circuits and Systems I*, Vol. 44, No. 10, pp. 997–1005.
- Raibert M. H., Hodgins J. A. (1993). Legged robots, In R. Beer R. Ritzmann and T. McKenna, editors, *Biological neural networks in invertebrate neuroethology and robotics*, pp. 319–354, Boston, MA: Academic Press.
- Randall D., Burggren W., French K. (2001) *Eckert: Animal Physiology 5th ed.* W H Freeman.
- Ritter H., Martinetz T., Schulten K. (1992), *Neural Computation and Self-Organizing Maps*, Addison Wesley.
- Saranli U., Buehler M., Koditschek D. E. (2000). Design, modeling and preliminary control of a compliant hexapod robot. In *IEEE Int. Conf. Robotics and Automation*, pp. 2589–2596, San Francisco, California.
- Shepherd G. M. (1997) *Neurobiology*. Oxford Univ. Press.
- Siciliano B., Sciavicco L. (1989) *Modelling and control of robot manipulators*, Springer: Berlin.
- Smith K. C., Sedra A. S. (1998) *Microelectronic circuits*. Oxford University Press.
- Strogatz S. H. (1994) *Nonlinear Dynamics and Chaos*, Perseus Books.
- Uchida H., Shimoi N., Jiu H. Q., Komizo D., Nonami K. (2000). Force/attitude control of mine detecting six-legged locomotion, In *Proc. Intelligent Robots and Systems, 2000 (IROS 2000)*, Vol. 1, pp. 780–785.
- Waldron K. J., Song S. (1989), *Machines that walk: The adaptive Suspension Vehicle*, MIT Press: Cambridge, MA.
- Watson J. T., Ritzmann R. E., Zill S. N., Pollack A. J. (2002). Control of obstacle climbing in the cockroach, *Blaberus discoidalis* I. Kinematics, *Journal of Comparative Physiology A*, 188, pp. 39–53.
- Watson J. T., Ritzmann R. E. (1998). Leg kinematics and muscle activity during treadmill running in the cockroach, *blaberus discoidalis*: I. slow running, *Journal of Comparative Physiology A*, Vol. 182, No. 1, pp. 11–22.
- Webb B., Consi T. R. (2001) *Biorobotics*, MIT Press.
- Webb B. (2001). Are biorobots good models of biological behavior, *Behavior and Brain Sciences*, 24(6).
- Wilson D. M. (1966). Insect walking, *Annual Review of Entomology*, vol. 11, pp. 103–122.
- Wolpert S., Friesen W. O., Laffely A. J. (2000). A silicon model of the *Hirudo*

- swim oscillator, *IEEE Engineering in medicine and biology*, Jan/Feb 2000, pp. 64–75.
- Zielinska T., Heng J. (2002). Development of a walking machine: mechanical design and control problems, *Mechatronics*, 12, pp. 737–754.
- Zou F., Nosssek J. A. (1991). Stability of cellular neural networks with opposite-sign templates, *IEEE Trans. Circuits and Systems I*, 38, pp. 675–677.
- <http://mindstorms.lego.com>.
- <http://www.robotelectronics.co.uk/hm/sd20tech.htm>.
- <http://www.scg.dees.unict.it/activities/biorobotics/movie.htm>.

This page intentionally left blank



# Index

- Arnold tongue, 51
- Attitude control, 115
  - Motor Maps for, 125
- Autowaves, 156
  
- Biped, 136
- Braitenberg vehicles, 43
  
- Chaos Control, 129
- Chua's circuit, 129, 183
- CNN
  - Definition, 13
  - Introduction, 12
  - Nonlinearity, 13, 44, 78
  - Template, 12
- CNN neuron, 14
  - Circuit, 179
- CNNLab, 173
- CPG, 2
  - CPG cell, 14
  - VLSI chip, 63, 112
- CPPM, 181
- Cycle time, 28
  
- Decentralized locomotion control, 73
  - CNN-based, 73
  - IF neurons, 88
- Deliberative behavior, 148
- Direction control, 43, 110
- Dispersion curve, 150
- Distance sensor, 110, 151
- Duty factor, 28
  
- Euler angles, 116
  
- Fitness function, 83
- Fixed-action pattern, 149
  
- Gait
  - Caterpillar, 28
  - Fast, 33
  - Half-bound, 38
  - Medium, 33
  - Slow, 34
  - Tetrapod, 76
- Genetic algorithms, 83
- Ground contact, 81
  
- HexaDyn, 171
- Hexapod robot
  - MTA hexbot, 106
  - MTA hexbot II, 110
  - MTA hexbot III, 112
  - Rexbot II, 116
- Hybrid control, 63
  
- Lamprey robot, 101
- Learning, 142
  
- McKibben muscles, 105
- Motor Maps, 125
- Motor system, 2
- MTA-CNN, 28
  
- Navigation control, 156

- Perception, 160
- Phase lag, 28
- Pitch angle, 117
- Poincaré-Bendixson Theorem, 15, 44, 79
  
- RD-CNN, 22, 146
- Reaction-Diffusion Equation, 22, 146
- Reactive behavior, 49, 111, 148
- Reflex, 57
  - Elevator reflex, 57, 95
  - Searching reflex, 58
- Relaxation oscillator, 91
- Reward Function, 128, 130, 133, 139, 143
- Roll angle, 117
  
- Sensory feedback, 43
  - Direction control, 47
  - Feedback and ground contact, 50
  - Hybrid control, 63
- Servomotors, 106
- Sonar sensor, 181
- Speed
  - Control, 60
  - Measurement, 112
- Stance phase, 5
- Step pattern, 29
- Stick insect, 73
- Swing phase, 6
- Synapse
  - Circuit, 180
  - Definition, 20
- Synchronization, 51
  
- Tele-operated control, 110, 174
- Turing pattern, 148
  
- Virtual equilibrium point, 16, 47