DESIGN AND CONTROL OF AMPHIBIOUS ROBOTS WITH MULTIPLE DEGREES OF FREEDOM

THÈSE N^{O} 3786 (2007)

PRÉSENTÉE LE 8 JUIN 2007

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS
INSTITUT DES SYSTÈMES INFORMATIQUES ET MULTIMÉDIAS
PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Alessandro CRESPI

ingénieur informaticien diplômé EPF
de nationalité suisse et originaire de Bellinzone (TI)

acceptée sur proposition du jury:

Prof. D. Thalmann président du jury
Prof. A. Ijspeert, directeur de thèse
Prof. H. Kimura, rapporteur
Prof. A. Menciassi, rapporteur
Dr F. Mondada, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse, EPFL
2007

Abstract

This thesis presents the design and realization of two generations of robot elements that can be assembled together to construct amphibious mobile robots. These elements, designed to be individually waterproof and having their own battery, motor controller, and motor, have been used to actually construct a snake, a boxfish and a salamander robot. Central pattern generator (CPG) models inspired from those found in vertebrates have been used for online trajectory generation on these robots and implemented on their onboard locomotion controllers. CPGs proved to be an interesting way of controlling complex robots, providing a simple interface which hides the complexity of the robot to the end user. Online learning algorithms that can be used to dynamically adapt the locomotion parameters to the environment have been implemented. Finally, this work also shows how robotics can be a useful tool to verify biological hypotheses. For instance, the salamander robot has been used to test a model of CPG for salamander locomotion.

Keywords: biologically inspired robotics, amphibious robotics, central pattern generators, locomotion.

Résumé

Cette thèse présente la conception et la réalisation de deux générations d'éléments destinés à être assemblés pour la construction de robots mobiles amphibies. Ces éléments, conçus pour être individuellement étanches et ayant leur propre accumulateur, contrôleur et moteur, ont été utilisés pour la réalisation d'un robot serpent, d'un robot poisson et d'un robot salamandre. Des modèles de *central pattern generator* (CPG) inspirés par ceux qui se trouvent dans les animaux vertébrés ont été utilisés pour la génération en temps réel des trajectoires et programmés dans les contrôleurs à bord de ces robots. Les CPGs ont montré d'être une solution intéressante pour le contrôle de robots complexes, fournissant une interface simple qui cache la complexité du robot à l'utilisateur final. Des algorithmes d'apprentissage online pouvant être utilisés pour adapter dynamiquement les paramètres de locomotion à l'environnement ont été implémentés. Ce travail montre également que la robotique peut être un outil intéressant pour vérifier des hypothèses biologiques. En l'occurrence, le robot salamandre a été utilisé pour vérifier un modèle de CPG pour la locomotion de la salamandre.

Mots clés : robotique bioinspirée, robotique amphibie, *central pattern generators*, locomotion.

Contents

Chapter 1

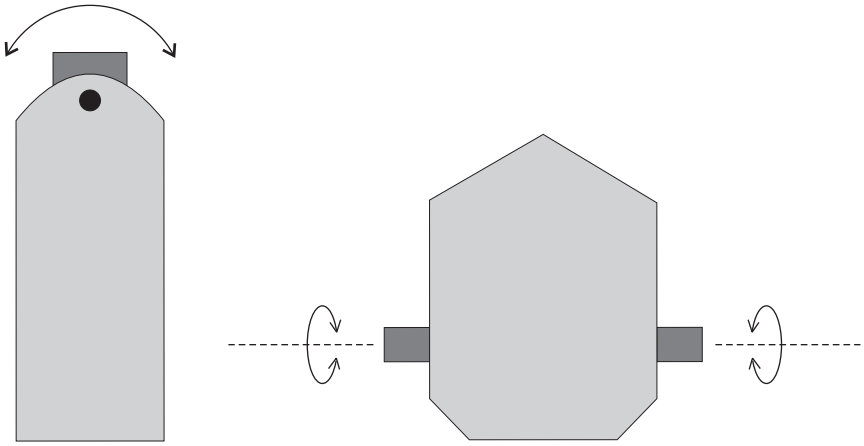
Introduction

The general goal of the project of which this thesis is part, is (1) to design robots that can be used to test neurobiological models of locomotion control and (2) to take inspiration of animals to design new types of robots and of locomotion controllers. Both goals primarily concern lower vertebrates, specifically the lamprey, the boxfish and the salamander.

The main goal of this thesis is the design of robotic tools to be used as building blocks for the construction of amphibious mobile robots. A modular approach has been taken: the idea is to have a limited number of modules (that has been fixed to two), which can be used to construct a variety of models. The most important design considerations behind these modules were the following:

- To have two types of elements: a "body" element (figure 1.1a) with a single vertical degree of freedom, and a "limb" element (figure 1.1b) with two aligned degrees of freedom capable of continuous rotation, parallel to the ground and perpendicular to the locomotion direction.
- To be completely modular, in terms of energy, actuation and motor control. This means that each element should have its own battery, motor, and motor controller.
- To be used for building amphibious robots. Each element should therefore be completely waterproof, independently from the other elements, and without the need for an external coating.
- To have a density lower than water, in order to be able to be slightly buoyant.

These assumptions, combined with the absence of off-the-shelf waterproof robot elements, implied that a custom design had to be made. For example, commercial servomotors do not meet the requirements for our design (mostly for their size and need of continuous rotation for the limb elements), therefore we needed to implement our own motor control system using a DC motor with incremental encoder, a motor controller and



(a) "Body" element (b) "Limb" element

Figure 1.1: Top view of the two types of planned elements.

a custom gearbox (see chapter 3 for all the details about the hardware implementation of the modules).

The modularity offered by such elements could be exploited to build several types of robots, simply by connecting them together in different ways: some examples are shown in figure 1.2. The robots which have been physically realized are the snake robot (described in chapter 4, see fig. 1.2a), a Boxfish-like fish robot (chapter 5, see fig. 1.2b) and the salamander robot (chapter 6, see fig. 1.2e).

A large part of this thesis thus concerns the development of these robot elements, solving the problems owing to the design considerations. All the mechanical part has been developed by André Guignard. All the electronics have been developed during this thesis, with the exception of the PID controller. The rest of the work concerns the software implementation of the robot controllers (and of all the needed external software and hardware, e.g., the transceiver connected to a PC which enables radio communication with the robot) and the experiments done with the realized robots, most of which have been published (see the different chapters for the references).

The project does not aim at mimicking snakes, lampreys, or salamanders *per se*, but to be close enough to them in order to meet the goals of the project. As stated above, the first goal of the overall project is to test neurobiological models of locomotion control: this requires the robots to have similar kinematic an dynamic properties to match the animal bodies. The robot is clearly only an approximation of the animal (for example, the number of degrees of freedom that a robot will have will be lower than the one of the corresponding animal, and DC motors have different properties from animal muscles), but it will be still useful to verify that the neural models to be tested are actually able to generate forward motion, and to provide speed and direction control. The approach

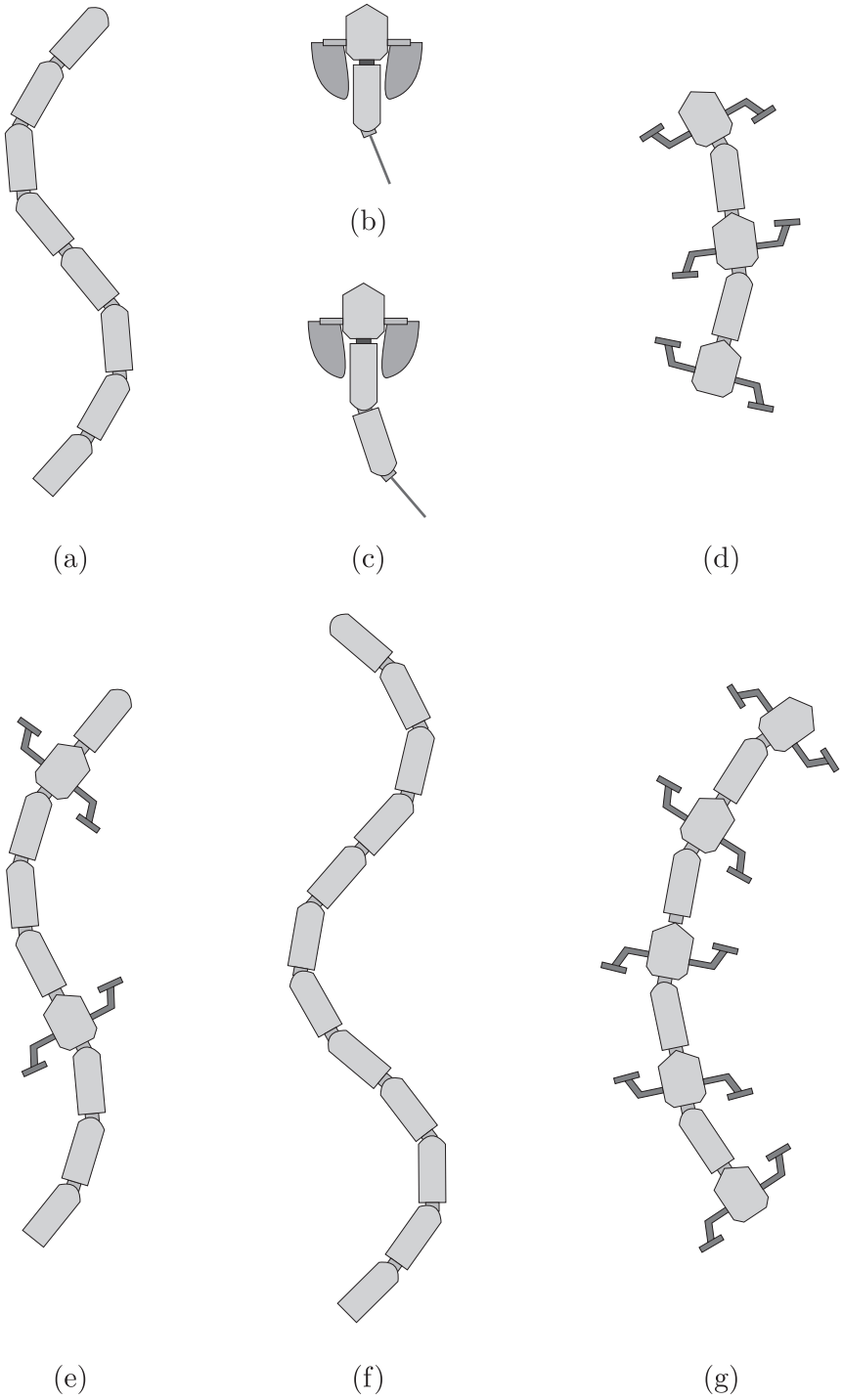


Figure 1.2: Examples of different types of robots that could be constructed by connecting together the two types of elements described. In this thesis, the robots (a), (b), and (e) will be constructed.

using a robot complements mechanical simulations, as with a real robot there is no need to model and simulate a complex environment (as it is the case with hydrodynamics, for instance).

The second goal of the project is to create novel types of agile robots that exhibit dexterous locomotion, taking inspiration from the body shape and neuronal control mechanisms found in vertebrates. Snake-like robots are indeed among the most flexible and versatile mobile robots. In particular, their long but thin body and its division in several small segments make them well-suited to a large number of applications. Such applications include, for example, exploration and inspection tasks (e.g., in areas that are inaccessible to humans, such as pipes) and the participation to *search and rescue* missions (e.g., in a collapsed building or a flooded zone). The salamander robot can be seen as an extension of the snake robot, which improves its locomotion on the ground (removing the necessity of the special contact mechanisms needed to create asymmetrical friction on ground; see chapter 4).

The concept of central pattern generator (CPG) will be extensively used in this thesis. Central pattern generators are networks of neurons that can produce coordinated oscillatory signals without oscillatory inputs (Delcomyn, 1980). In vertebrates, CPGs for locomotion are located in the spinal cord and distributed in multiple oscillatory centers. A CPG could be modeled in different ways, namely using neural networks or dynamical systems like nonlinear oscillators; the second approach will be used here.

CPGs are an interesting source of inspiration for controlling robots: (1) they implement a control scheme that can be implemented in a distributed fashion, (2) they require only simple command signals to produce complex coordinated multi-dimensional output signals, and (3) they easily incorporate sensory feedback and take mechanical perturbations into account. These properties make them suitable for the implementation of simple interfaces to facilitate the control of a complex robot by a human operator, as simple input signals are enough to generate complex locomotion. It is also possible to implement CPGs for different robots accepting the same inputs, therefore allowing the same control interface to be used for different types of robots.

Recently, the concept of CPGs is increasingly used as an alternative approach for online rhythmic trajectory generation (Fukuoka et al., 2003; Nakanishi et al., 2004; Ijspeert et al., 2005). In most cases, the CPGs are implemented as recurrent neural networks or systems of coupled nonlinear oscillators. In this thesis, CPG models will be presented for controlling the swimming, serpentine crawling, and walking for the fish, snake and salamander robots.

The main contributions of this thesis are the following:

- The construction of generic elements, meeting the constraints stated in this chapter (and more in detail in chapter 3), that can be assembled together to obtain amphibious biologically inspired robots.

- The construction of one of the first amphibious snake robots. In the literature it is possible to find a variety of terrestrial snake robots of different types, a small number of aquatic snake robots (generally named eel or lamprey robots), but to the best of our knowledge there are only two other truly amphibious snake robots (the HELIX-I, see Takayama and Hirose (2002), and its successor ACM-R5, see Yamada et al. (2005)). A review of existing snake robots is presented in the next chapter.

- The construction of the first amphibious salamander robot, and the first robot capable of swimming, serpentine crawling, and walking.

- The use of central pattern generators for online control of a mobile robot. CPGs are currently more and more used for locomotion control in robots. This thesis will show how CPG models based on amplitude-controlled phase oscillators can be constructed to control a variety of different gaits.

- The use of an online learning algorithm for the automatic determination of the optimal locomotion parameters of the snake robot in a given environment.

In the next chapters, a review of related work is presented (chapter 2), followed by a detailed technical description of the two generations of robot elements that have been developed (chapter 3). The following chapters present the snake (chapter 4), fish (chapter 5) and salamander (chapter 6) robots, and the experiments done with them. The use of online learning on the snake robot for the determination of optimal locomotion parameters is presented in chapter 7. Finally, conclusions and future work are discussed (chapter 8).

Chapter 2

Related work

In this chapter, work relating to this thesis will be presented. The first section presents existing snake robots, and the control methods used with them. A review of currently existing fish robots and of control methods is then presented, followed by a section on salamander robots. Other related work (e.g., modular robots) is also briefly presented. The last section presents a short review about the use of robots as tools for biology.

2.1 Snake robots

2.1.1 Hardware

Snake robots can be classified into two main groups:

- Robots that move using powered wheels or caterpillars (i.e., a torque is applied on the axis of the wheels, which are in contact with the ground, producing a rotation and consequently a movement).
- Robots that move by applying torques on the joints between the segments. Among these robots, some have passive wheels.

Robots using powered wheels are simpler to control: the design techniques are well known and standard algorithms for the control of mobile robots can be used; however, the resulting locomotion is completely artificial and the wheels may not be adequate in every environment. Robots of this type are often developed for inspection tasks in difficultly accessible zones (Paap et al., 1996; Klaassen and Paap, 1999) and are sometimes used, for example, for the inspection of pipes (Choi and Ryew, 2002). On the other side, robots that use powered joints instead of powered wheels are more complicated to design, and the control algorithms that can be used are partially unexplored. As we aim to design a biologically inspired snake robot that can both crawl and swim with powered joints, we are mainly interested in this second approach.

One of the first known snake robots was built by Hirose and colleagues at the end of 1972 (Umetani and Hirose, 1976). He generically named this kind of robot an *active cord mechanism* (ACM). After this first prototype he built some other snake robots (Hirose, 1993). A huge snake robot has been developed in 1992 at Caltech (Chirikjian and Burdick, 1992). The Jet Propulsion Laboratory of the NASA presented in 1994 a serpentine robot (Lee et al., 1994). Miller developed several prototypes of snake robots; among them the last one, S5 (Miller, 2002), has a very realistic lateral undulatory gait (its locomotion is probably the most similar to a biological snake, compared to other snake robots). Saito and colleagues presented in 2002 a simple snake robot used to validate some theoretical results (Saito et al., 2002). Conradt and Varshavskaya (2003) developed WormBot, a snake-like robot controlled by local CPGs. For a more detailed review of snake robots, see Dowling (1997) and Worst (1998).

Swimming snake robots (also referred to as lamprey robots or eel robots) are rarer. They are generally designed to imitate the anguilliform swimming of the eel (or the very similar one of the lamprey). Several theoretical papers have been written on this subject, but there are only a few real robotic realizations. The robots in this category that are the most interesting are the eel robot REEL II (McIsaac and Ostrowski, 1999) and the lamprey robot built at Northeastern University (Wilbur et al., 2002). In principle, these eel and lamprey robots could be adapted to terrestrial locomotion, but such experiments have not been reported. To the best of our knowledge, there are currently only a few amphibious snake-like robots, the HELIX-I (Takayama and Hirose (2001) as cited by Hirose and Fukushima (2002); Takayama and Hirose (2002)) and its successor ACM-R5 (Yamada et al., 2005), that can both swim in water and crawl on the ground (although ground locomotion is not described in the papers).

2.1.2 Control architectures

The control architectures used with snake robots can roughly be divided into three categories: sine-based, model-based, and CPG-based.

Sine-based approaches use simple sine-based functions for generating travelling waves (see for instance Miller (2002); Tsakiris et al. (2005)). The advantages of such an approach are its simplicity and the fact that important quantities such as frequency, amplitude and wavelength are explicitly defined. A disadvantage is that online modifications of the parameters of the sine function (e.g., the amplitude or the frequency) will lead to discontinuous jumps of setpoints, which will generate jerky movements with risks of damaging the motors and gearboxes. This problem can to some extent be overcome by filtering the parameters and/or the outputs but the approach then loses its simplicity. Another disadvantage is that sine-based functions do not offer simple ways of integrating sensory feedback signals.

Model-based approaches use kinematic (Ostrowski and Burdick, 1996; Matsuno and Suenaga, 2003) or dynamic (Prautsch and Mita, 1999; Date et al., 2001; Ute and Ono,

2002; McIsaac and Ostrowski, 2003) models of the robot to design control laws for gait generation. The control laws are sometimes based on sine-based functions as above (e.g., Ostrowski and Burdick (1996); McIsaac and Ostrowski (2003)), but the model-based approaches offer a way to identify fastest gaits for a given robot by using kinematic constraints or approximations of the equations of motion, for instance. Model-based approaches are therefore very useful for helping to design controllers but have two limitations. First, the resulting controllers are not always suited for interactive modulation by a human operator. Second, the performance of controllers will deteriorate when models become inaccurate, which is rapidly the case for interaction forces with a complex environment (e.g., friction with uneven ground).

CPG-based approaches use dynamical systems, e.g., systems of coupled nonlinear oscillators or recurrent neural networks, for generating the travelling waves necessary for locomotion (see for instance Lu et al. (2005); Tsakiris et al. (2006); Wilbur et al. (2002); Conradt and Varshavskaya (2003)). These approaches are implemented as differential equations integrated over time, and the goal is to produce the travelling wave as a limit cycle. If this is the case, the oscillatory patterns are robust against transient perturbations (i.e., they asymptotically return to the limit cycle). Furthermore, the limit cycle can usually be modulated by some parameters which offer the possibility to smoothly modulate the type of gaits produced. Finally, CPGs can readily integrate sensory feedback signals in the differential equations, and show interesting properties such as entrainment by the mechanical body (Taga, 1998).

However, one difficulty with CPG-based approaches is how to design the CPG to produce a particular pattern. Many CPG models do not have explicit parameters defining quantities such as frequency, amplitude, and wavelength (for instance, a van der Pol oscillator does not have explicit frequency and amplitude parameters). This does not need to be the case. The CPG model that has been used for the snake robot presented in chapter 4 is based on amplitude-controlled phase oscillators. An interesting aspect of this approach is that the limit cycle of the CPG has a closed form solution, with explicit frequency, amplitude and wavelength parameters. The approach therefore combines the elegance and robustness of the CPG approaches with the simplicity of sine-based approaches. Furthermore, our CPG model is computationally very light which makes it well suited to be programmed on a simple microcontroller on board of the robot. The implementation of the CPG is inspired from lamprey models (Cohen et al., 1982). It is close to the CPG model presented in Conradt and Varshavskaya (2003), but differs in the following aspects: (1) it is made of a double chain of oscillators, (2) it has differential equations controlling the amplitudes of each oscillator (not only the phase), (3) it has an interface function that allows easy modulation of speed and direction by a human operator, and (4) the CPG is used to control not only serpentine crawling but also swimming.

2.2 Fish robots

Although more recent than snake robots, a number of fish robots have been developed in the last years. Most of them have been designed to study the hydrodynamics of fishes and underwater vehicles.

Multiple fish robots have been designed and realized. Most robots implement anguilliform or carangiform swimming modes, which mainly use the body and the tail for propulsion (Sfakiotakis et al., 1999; Colgate and Lynch, 2004). Ostraciiform or labriform modes, which use caudal and pectoral fins and almost no body motions, have been less studied. Relatively few fish robots are fully autonomous, capable of swimming in 3D and reacting to their environment. For instance, the well-known RoboTuna from MIT, which has been designed to study speed optimization, is attached to a horizontal guide (Triantafyllou and Triantafyllou, 1995).

Several groups are very active in designing autonomous fish robots (Kato, 2000; Liu et al., 2004; Yu et al., 2004). The National Marine Research Institute (NMRI) in Japan, for instance, is working on multiple projects, including maneuvering, swimming performance and modular robotics for water; each robot is built for a particular purpose like up-down motion, high turning performance, or high speed swimming.[1] The University of Essex developed a 3D swimming robotic fish called MT1 which is fully autonomous (Liu et al., 2005). A micro robotic fish actuated by PZT bimorph actuators has recently been built by the University of California, Berkeley (Deng and Avadhanula, 2005), mimicking a boxfish.

Most of these robots are controlled using traditional control methods that combine (algorithmic) sine-based trajectory generators, and PID feedback controllers.

To the best of our knowledge, CPGs have rarely been applied to the control of a swimming robot. Arena and Ayers' groups have independently used CPG models inspired by the lamprey locomotor network for controlling tethered lamprey-like robots (Arena, 2001; Wilbur et al., 2002). The robots were capable of producing travelling waves for propulsion, but autonomous swimming was to the best of our knowledge not explored.

2.3 Salamander robots

Currently only a few prototypes of salamander robots have been object of scientific publications:

- A salamander robot with 6 segments and an on-board FPGA-based control system has been presented by Hiraoka and Kimura (2002). It is not amphibious and can only walk.
- Robo-Salamander, a salamander robot with two degrees of freedom for the spine, and two for each leg, has been presented by Breithaupt et al. (2002); no experiments

[1] Fish Robot Home Page of NMRI. URL: http://www.nmri.go.jp/eng/khirata/fish/index_e.html

seem to have been done with it, and no other publications followed. This robot was not autonomous and was powered and controlled using a cable. It is only capable of walking.

There are also some legged robots with flexible spine built by hobbyists, whose descriptions can be found on Internet, but none of them has been designed or used for scientific experiments.

None of the robots listed here is capable of swimming, and none is fully autonomous or amphibious.

2.4 Other related work

2.4.1 Aquatic and amphibious robots

Although amphibious snake robots are rare, and no amphibious salamander robot has been previously described, a number of other aquatic and amphibious robots currently exists. A short review of existing fish robots has already been presented in section 2.2. Other examples of aquatic and amphibious robots include Aqua (an amphibious RHex robot that can mount either 6 flippers or 6 legs; Dudek et al. (2007)), Madeleine (an aquatic tetrapod robot having 4 flippers; Long et al. (2006)), RoboLobster (an amphibious lobster robot with 8 legs; Ayers et al. (2000)), and Ariel (a 6-legged robot produced by iRobot, for which no publication is available).

2.4.2 Modular robotics

There is a great number of robots that have been designed for being modular and reconfigurable. These robots are made of multiple identical modules and can change structure thanks to dynamic connection mechanisms. Examples of such robots include M-TRAN (Murata et al., 2002), PolyBot (Duff et al., 2001), CONRO (Shen et al., 2002), ATRON (Jørgensen et al., 2004) and SuperBot (Shen et al., 2006). Although the modularity idea of this project is clearly related to the field of modular robotics, the reconfiguration aspect is not our primary goal (the elements have to be unmounted to be assembled together, and no attempt to automatic reconfiguration has been done), and modularity is mostly a way to simplify the construction of the robots. To the best of our knowledge, there is no waterproof modular robot, excepting the Hydron module prototypes developed for the Hydra project (Konidaris et al., 2004).

2.4.3 Robots with rotating legs

The approach used for the legs of the salamander robot presented in chapter 6 is directly inspired from the ones found in the RHex and Whegs robots.

RHex is the name of a series of hexapod robots with rotating legs (Saranli et al., 2001; Prahacs et al., 2005). The term *whegs* (derived from wheel-legs) has been introduced by another research group for the same concept (Quinn et al., 2001). This approach greatly simplifies the mechanical design and control (each leg has only one degree of freedom and thus only one motor), and still allows behaviours which are very similar to the biological ones (i.e., legs having a stance and a swing phase).

2.5 Robots as tools for biology

This thesis is part of a new trend to use robots as tools to verify biological hypotheses or as models of biological sensorimotor systems (Webb, 2000). Examples include lamprey locomotion (Wilbur et al., 2002; Stefanini et al., 2006), lobster locomotion (Ayers and Crisman, 1993), cricket phonotaxis (Webb and Reeve, 2003) and cat locomotion (Fukuoka et al., 2003). For a more detailed review, see Webb (2001, 2002).

Compared to computer simulations, the use of real robots is interesting as it provides several advantages:

- The model is completely interacting with a real environment, using real sensors and real actuators. This therefore eliminates the need to simulate the sensors and the actuators (which can be generally simulated only with approximate models). The absence of simplified models or biased results owing to the simulation is a great advantage, as some aspects could strongly depend on the interaction with the environment.

- There is no need to simulate complex environments or complicated force models. A correct simulation of some phenomena (for example friction forces, hydrodynamical forces, etc.) is extremely difficult (especially if associated with articulated moving bodies, whose shape is not constant); simulations are therefore generally limited to a simplified model which could introduce artifacts that cause the model to behave differently than in the real world.

However, it is also important to notice that the use of real robots has some drawbacks compared to simulations:

- Reproducing the mechanical properties of real animal bodies on a robot is very difficult. A robot will be an approximation of the real animal, as generally it is technically not feasible to build a robot having the same properties (especially for the number of degrees of freedom: a robot with hundreds of degrees of freedom like a real snake would be much larger than the animal). The visco-elastic properties of animal muscles are also difficult to implement in robots.

- Almost everything can be designed in a simulation, including systems using components (e.g., sensors or actuators) that are expensive, hard to use, or even not existing with the current technology. For example, some animal sensor systems (like touch) are difficult to replicate with currently existing sensing devices.
- Building a robot generally requires much more work than implementing a simulation, and robots sometimes have to be repaired or maintained. Moreover, robots only run in real time (simulations can be faster).

Chapter 3

Technical description

In this chapter, the design considerations underlying the developed robot elements are presented. A technical description of the first prototype is given, followed by the description of the current elements and of the fundamental differences between the two versions. Finally, the physical simulation of the robot is presented.

The elements described here have been designed as modules to be used to build a snake and a salamander robot. Later, they have been used to build a fish robot. They could also be used for other types of robots (e.g., an hexapod or a centipede). All the elements have been designed to be waterproof, in order to be used for the construction of amphibious robots.

All the mechanics have been developed by André Guignard. The PD controller and its electronics have been developed at the Autonomous Systems Laboratory (ASL) at EPFL. My contribution has principally been the development of the electronics (except for the already existing PD controller) and its testing and programming.

3.1 Design overview

The two types of robot elements ("body" and "limb" elements, see figure 1.1) were designed to build robots with the following characteristics:

- To be modular. We aim at having a robot that is composed of multiple elements, with only a few element types. This allows us, for instance, to quickly adjust the length of the robot by adding or removing elements, as well as to replace defective elements.

- To have distributed actuation, power and control. In order to be truly modular, each element carries its own DC motor, battery, and microcontroller.

- To be waterproof. Each individual element is made waterproof (as opposed to having a coating covering a chain of elements). This facilitates modularity and ensures that a leakage will only damage a single element.

- To be slightly buoyant. We aim at having a robot that passively returns to the surface of the water when inactive. Furthermore, we construct the elements such that the center of gravity is placed below the geometrical center, in order to obtain a vertical orientation that self-stabilizes in water.

- To have large lateral surfaces for good swimming efficiency.

- For the snake robot, to have asymmetric friction for the lateral undulatory (serpentine) locomotion on ground (lower friction coefficient in the longitudinal axis compared to the perpendicular axis).

- To be controlled by a central pattern generator (CPG) composed of coupled non-linear oscillators.

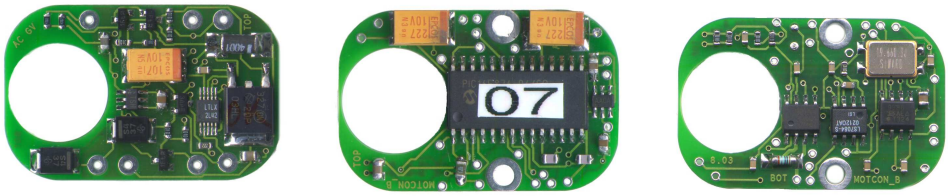
- To be remotely controlled in terms of speed and direction commands, but otherwise have an onboard locomotion controller for coordinating its multiple degrees of freedom.

3.2 First prototype

This section gives an overview of how the two types of elements of the first prototype of the robot were built, and then explains the encountered problems. The elements described in this section have been used to build a snake robot, named AmphiBot I; the robot and the experiments done with is have been published in Crespi et al. (2004, 2005a,b).

3.2.1 Body elements

Each element has a single degree of freedom, and elements are fixed such that all axes of rotation are aligned. They consist of four structural parts: a body, two covers and a connection piece (a drawing of two connected elements is visible in figure 3.2). All parts are molded using polyurethane, using molds created from positive parts in aluminium shaped with a CNC milling machine. The Li-Ion battery is directly incorporated into the bottom cover when the polyurethane is cast in the mould. To ensure the waterproofing of the robot, O-rings are placed between each cover and the body, and around the output axis (the bottom O-ring has been subsequently replaced by a silicone sealant, because the complete closing of the bottom cover was generating mechanical problems to the gearbox). An element has a length of 7 cm and a section of 5.5 by 3.3 cm.



(a) Power supply (b) Motor controller (top) (c) Motor controller (bottom)

Figure 3.1: Pictures of the printed circuit boards of the first prototype (real size).

Each element contains two printed circuits (one for the power supply/battery charger and one for the motor controller, see figure 3.1), a DC motor and a set of gears. Two different voltages are used inside an element: 3.6 V and 5 V. The first one is the typical value of a Li-Ion battery and is only used to power the motor; the second one is used to power the electronics. When the robot is battery-powered (no external power source is connected), the motor is directly powered using the battery, without any intermediary regulator or converter, and the 5 V used by the electronics are generated with a capacitive charge-pump step-up converter (LTC3200). When an external (5 V) power source is connected, the 3.6 V for the motor are generated using a low-efficiency diode to create a voltage drop, and the electronics are directly powered using the external source. When the external power source is present, the battery could also be charged if this is necessary; for this reason a small battery charger (LTC1733) is part of the power supply circuit. The charger can be enabled or disabled by the user over the I^2C bus. The battery has a capacity of 600 mAh, which is enough to power the element for an average time of approximately two hours of continuous use (but this largely depends on the movements that the robot has to do and on the external constraints applied to it). An empty battery can be charged in approximately one hour.

The motor controller is built with a PIC microcontroller (PIC16F876) and some external components. The motor has a magnetic encoder, which generates 16 impulsions for every complete rotation of the axis. This encoder is connected to a LS7084 quadrature detector that filters and decodes the signals of the magnetic coder, generating a clock signal and a direction flag; these two signals are sent to the microcontroller, allowing it to track the current position of the motor. A 10 kΩ potentiometer is fixed to the output axis (after the reduction gears) and is connected to an analog input of the PIC; this potentiometer can be used to read the absolute position of the axis (for example when the robot is switched on, or to detect possible skews between the position measured with the magnetic coder and the real one).

The motor coil is powered through a SI9986 H-bridge, which supports currents up to 1 A. The H-bridge is driven by the microcontroller using a Pulse-Width Modulation (PWM) signal, allowing the the speed of the motor to be changed.

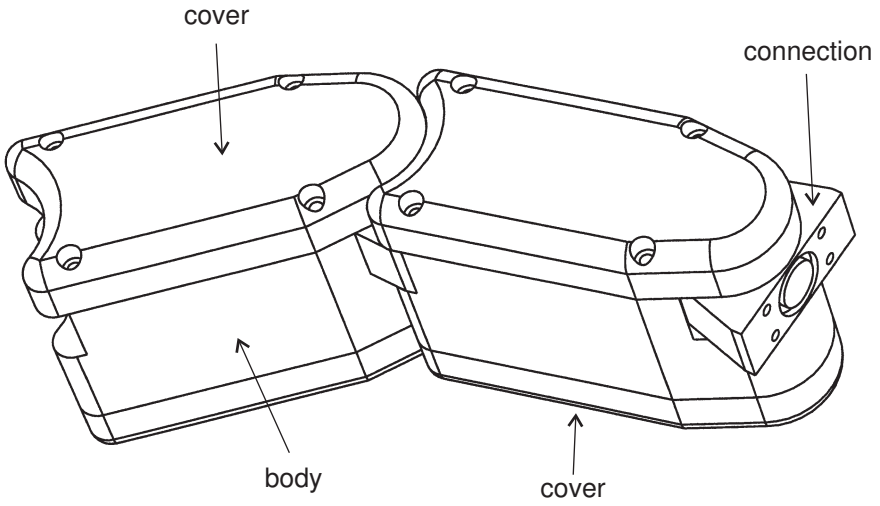


Figure 3.2: Drawing of two elements connected together.

Between the H-bridge and the motor, a 1 Ω resistor causes a voltage drop. The resistor is connected to the input of an INA146 operational amplifier, the output of which is connected to one of the analog inputs of the microcontroller, therefore allowing a measure of the current used by the motor, and then indirectly of its torque.

The 0.75 W DC motor (having a maximum torque 1.2 mN·m) drives a set of reduction gears with a reduction factor of 400, and an efficiency around 60%. The output axis of the gears is fixed to the aforementioned potentiometer and to the connection piece fixed to the next element. Considering the typical working speed of the motor and the reduction of the gears, a maximum oscillation frequency of approximately 0.3 Hz can be obtained if the full amplitude (±45°) is used.

Five wires, passing through the (internally empty) axis, are connected to the contacts that are molded into the connection piece; four of them are used to pass the I^2C bus and the external power source all along the robot.

3.2.2 Limb elements

The first limb elements, very similar in structure to the current ones (see section 3.3.2) but with the same electronics as the old body elements, were also tested as a first prototype of a salamander robot (without its tail; see figure 3.5).

Most of the body elements were damaged by water leakages (which were not immediately detected) after the tests published in Crespi et al. (2005b); only three of them were still working and have been used for the salamander prototype. This was therefore only a preliminary design for testing the conception of the limb elements (no experiments have

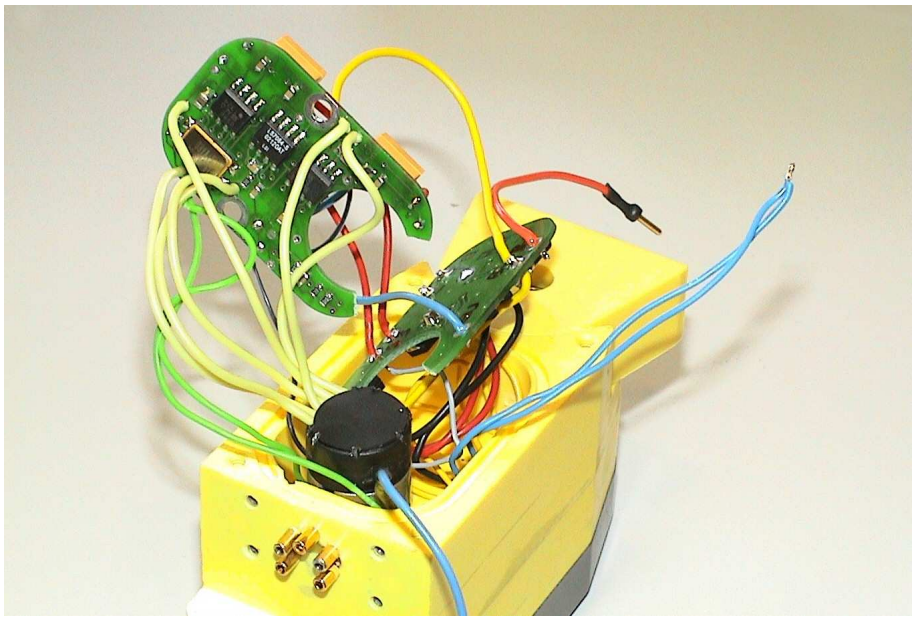


Figure 3.3: View of an element of the first prototype during mounting.

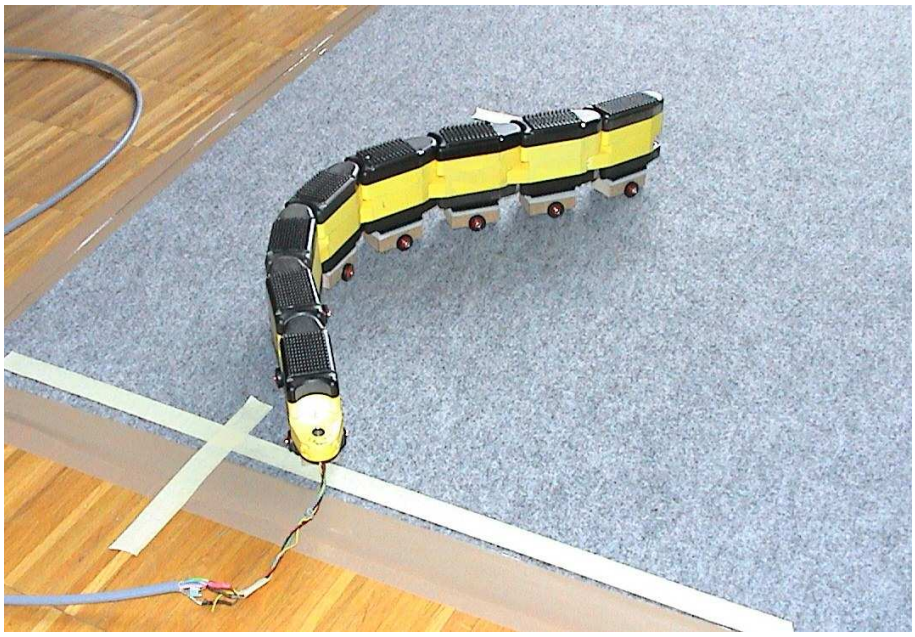


Figure 3.4: The AmphiBot I robot, built with first generation elements, during snake locomotion tests. The passive wheels are clearly visible on the bottom of the elements.

Figure 3.5: The prototype of salamander robot built with the first generation elements.

been done), and no detailed description of them will be given.

3.2.3 Design problems

This first prototype suffered of several design problems, which have been mostly corrected in the current version of the elements (see next sections):

- The direct use of 5 V for the external power supply (mostly due to the lack of internal space for step-down converters, which require big coils) rendered the usage of the robot with external power (and the battery charging) very problematic, as only a limited amount of current can pass through the internal wires (having a section of 0.127 mm^2). For instance, a current of approx. 2 A on the wires caused a voltage drop along the robot around 2.5 V, causing part of the elements to reset (disabling battery charging).

- The torque generated by the elements was insufficient to achieve full oscillations at frequencies greater than 0.3 Hz, resulting in very slow locomotion.

- The waterproofing of the elements was very problematic and required sealing them with silicone.

- There was no possibility to detect the presence of water inside elements. Any malfunctioning supposedly owing to water leakage required the robot to be completely

unmounted.

- The rigid connection between the elements combined with the small differences in the pieces caused the mounted robot to have bad contacts with the ground (i.e., it was not perfectly flat). This resulted in suboptimal serpentine crawling. Moreover, the used passive wheels were too small and badly attached to the elements.

- No battery protection mechanisms were implemented, and there was no possibility of turning off the robot, therefore it had to remain connected to the external power all the time to preserve the batteries from being completely discharged (and thus rendered unusable).

- No connectors were on the circuits, and all the connections (including those to the motor) were realized by directly soldering the wires to the PCB (see figure 3.3). This operation was difficult (hence giving high mounting times for each element) and rather unreliable.

- The absence of any onboard trajectory generation capabilities and of radio communication required the direct control of the robot through a long shielded cable connected to a PC using a RS-232–I^2C converter.

3.3 Current elements (second prototype)

These new prototype address most of the problems found with the previous one, particularly in terms of mounting simplicity, electronic reliability, and waterproofing. The elements described in this section have been used to build a snake robot (AmphiBot II, results can be found in chapter 4 and Crespi and Ijspeert (2006); Ijspeert and Crespi (2007)), a boxfish robot (BoxyBot, see chapter 5 and Lachat et al. (2006); Crespi et al. (2007)) and a salamander robot (Salamandra robotica, see chapter 6 and Ijspeert et al. (2007)).

3.3.1 Body elements

The same material of the first generation elements (polyurethane resin lighted with glass microballs) has been chosen for the external casing of the elements. They consist of two vertical symmetrical parts that are fixed together with screws. This is different from the first prototype which was having a body closed with two covers (top and bottom). The elements are connected (both mechanically and electrically) using a compliant connection piece (molded with polyurethane rubber) fixed to the output axis, which contains 6 wires. The use of compliant connection pieces corrects the bad contact with the ground that was a serious problem of the previous generation elements, and allows the robot to better

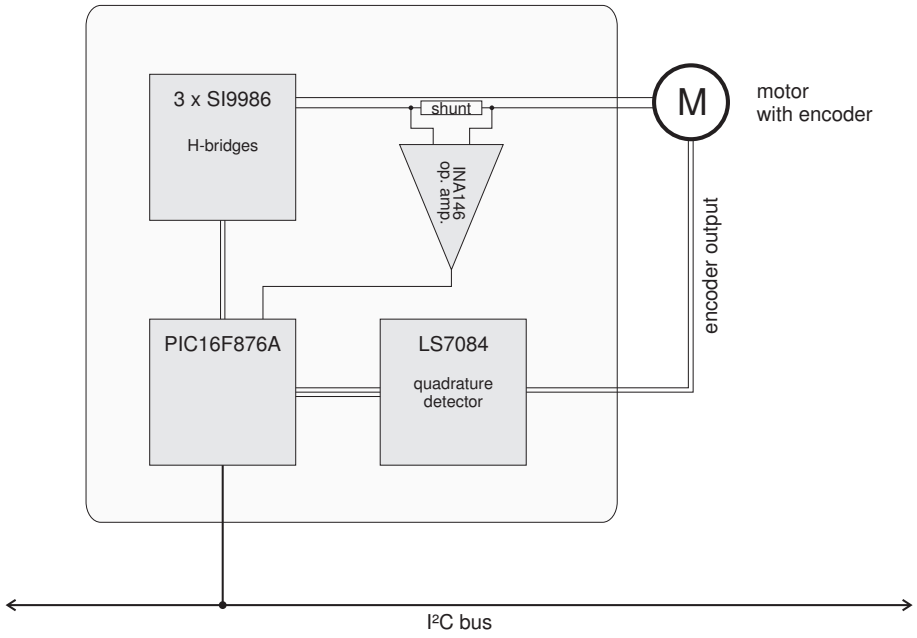


Figure 3.6: Block schema of the PD controller of the body (and legs) elements.

deal with irregularities of the ground. All the output axes of the elements are aligned, therefore producing planar locomotion. To ensure the waterproofing of each element, custom O-rings (placed between the two parts composing the body) are used.

Each element contains three printed circuits (a power board, a PD motor controller and a small internal water detector) connected with a flat cable, a DC motor with an integrated incremental encoder, a set of gears (which uses two additional printed circuits as mechanical support) and a rechargeable Li-Ion battery. A view of an open element can be seen in figure 3.8. In opposition to the old elements, where all connections were realized by soldering the wires directly on the printed circuit boards, the new circuits use MicroMatch connectors for all the interconnections (bus, battery, motor and inter-circuit connection); only the water detector (which was added later to the design; see below) uses directly soldered wires for reasons of space. The elements are completely independent from each other (both electrically and mechanically). The density of the robot elements is slightly lower than 1 kg/m^3 (the density of the old elements was slightly higher, therefore the first robot was not buoyant). The battery is placed at the bottom of the elements to have the center of mass below the vertical center, therefore ensuring the vertical stability of the robot during both swimming and crawling.

In this description, for simplicity, we will not distinguish on which of the printed circuits each component is located. The motor controller is based on a PIC16F876A microcontroller, and is basically the same of the first prototype. It is connected to the I^{2}C

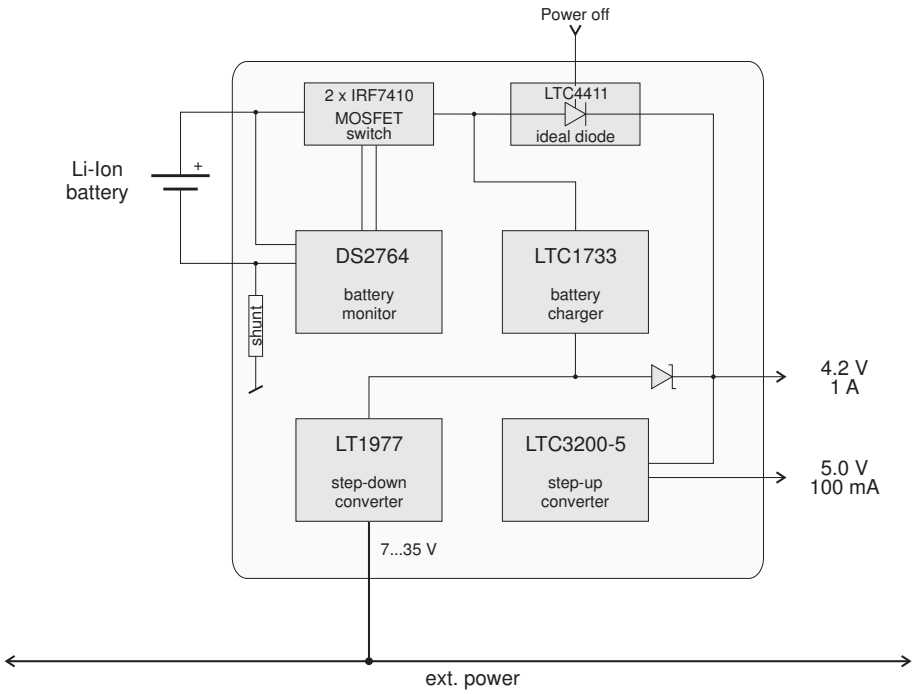


Figure 3.7: Block schema of the power circuits of the body (and legs) elements.

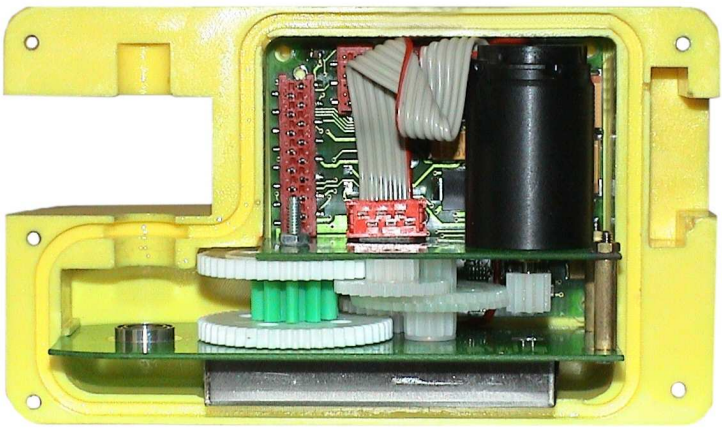
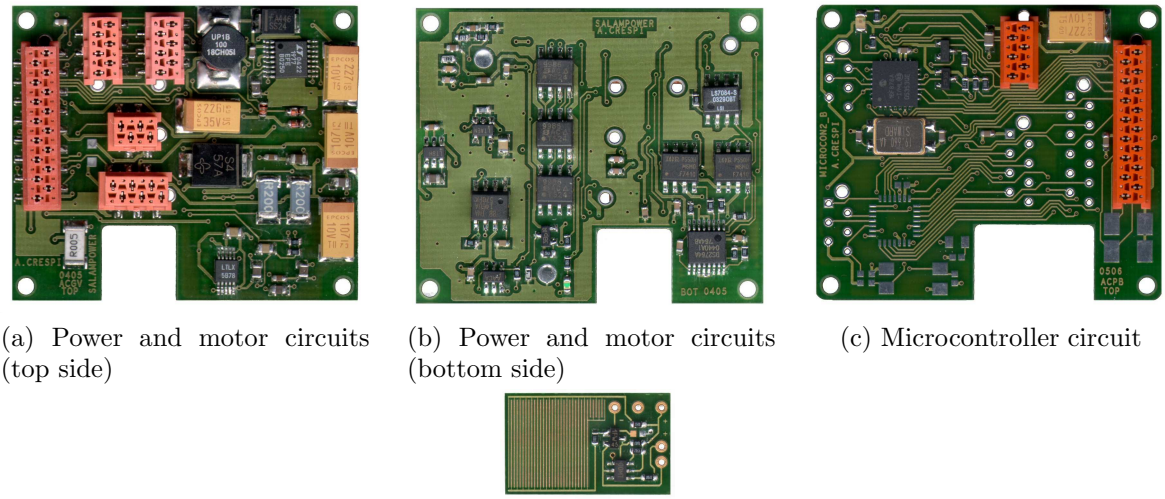


Figure 3.8: Internal view of a body element (real size). The output axis is not mounted.



(a) Power and motor circuits (top side)

(b) Power and motor circuits (bottom side)

(c) Microcontroller circuit

(d) Water sensor

Figure 3.9: Pictures of the printed circuit boards of the robot (real size).

bus of the robot through a simple bidirectional repeater (built using two BSS138 MOS transistors), which is very useful to protect the microcontroller internal drivers. The motor has an integrated magnetic incremental encoder, which generates 512 pulses for every complete rotation of the motor axis. The encoder is connected to a LS7084 quadrature detector that filters and decodes the signals coming from the encoder, generating a direction flag and a clock signal, which are connected to the microcontroller. Compared to the first generation elements, the potentiometer has been removed to simplify the mechanical structure.

The motor coil is powered through three SI9986 buffered H-bridges connected in parallel (each of which has a maximum current of 1 A; the maximal current that can be drawn by the motor is thus 3 A). These H-bridges are driven by the microcontroller with a Pulse-Width Modulation (PWM) signal, allowing the speed of the motor to be changed by modifying the duty cycle of the control signal.

To measure the current used by the motor (and then, indirectly, its torque), a couple of 0.2 Ω resistors in parallel are inserted between the output of the H-bridges and the motor. The voltage drop obtained on these resistors is amplified by a INA146 operational amplifier and sent to an analog input of the microcontroller. The negative power (-5 V) for the operational amplifier is generated using a small capacitive inverter (MAX1719).

The power supply part of the electronics has been completely redesigned compared to the first prototype. A battery monitoring and protection circuit, which was missing, has also been included. The circuit generates the voltage required by most of the electronics (5 V) using a capacitive charge-pump step-up converter (LTC3200-5). All the electronics can be either powered by the internal Li-Ion battery, or by an external power source

(connected to the last element and distributed internally to all elements). When no external power source is connected, the battery (connected to the rest of the circuit through a DS2764 battery monitoring/protection circuit that controls two IRF7410 power MOSFETs) directly powers the motor. When an external power source is connected, an inductive step-down converter (LT1977) generates a voltage of approximately 4.6 V, which can both replace the battery voltage (to power the motor and the step-up converter) and power the LTC1733 battery charger. The circuit accepts up to 35 V (to reduce as much as possible the current on the internal wires, which have a limited section). The switch between the internally generated 4.6 V and the battery is realized with a LTC4411 "ideal diode" and a SS34 Schottky diode. The used battery is the same that was used in the previous prototype and has a capacity of 600 mAh; it can power an element for approximately two hours of continuous use in normal conditions. When empty, the battery can be recharged in approximately one hour. The battery protection circuit disconnects the battery when its voltage drops below a critical threshold, thus preserving it from the often irreversible complete discharging. The circuit can also measure the instantaneous and accumulated current used by the circuit (or by the battery, during charging), and the battery voltage. This information could be read out using an I^2C bus, but these signals are currently left unconnected on the power card, to limit the total number of devices on the bus (which is global to the robot).

A signal coming from a reed contact placed in one of the elements allows the user to switch off the robot by placing a magnet on it. This solution was found to be simpler than using a big waterproof switch. This signal is connected to the enable pin of the aforementioned LTC4411 (no current is drawn, the signal can therefore be directly generated using one of the batteries).

The water detector circuit (fig. 3.9d), used internally to detect and localize any leakage, is placed at the bottom of the element. It has been introduced in the current elements to ease the detection of water leakages. It has a sensitive surface of about 1 cm^2, consisting of several parallel tracks, half of which are connected to the power source through a resistor. When water (or a big amount of moisture) is on this surface, it acts like a resistor between the power source and the base of an NPN transistor, which begins to conduce. When water is detected, the circuit blinks a LED fixed through the top of the element, therefore allowing the user to immediately detect the leakage and its position (i.e., the concerned element). The LED blinking is implemented using a PIC10F200 microcontroller, which in normal conditions accepts an incoming control signal for the LED and transparently replicates it on the output, hence permitting the LED to be used for other purposes. The introduction of this water detector dramatically simplified the handling of water leakages in the robot, which can now be localized without unmounting all the elements.

The 2.83 W DC motor (Faulhaber 1724 T 003 SR) has a maximum torque of 4.2 mN·m and drives a gearbox with a reduction factor of 125. It is approximately four times faster and stronger than the motor used in the previous generation of elements, therefore allowing higher amplitudes and oscillation frequencies to be reached. The output axis of

Figure 3.10: Internal view of a limb element (real size).

the gears is fixed to the connection piece, which is inserted into the next element. Six wires are inserted into the axis, and connected to the power boards of two adjacent elements: two are used for the external power, two for the I^2C bus, one for the power switch and the last one is reserved for future usage and currently unconnected.

3.3.2 Limb elements

The limb elements have been designed mainly as legs for the salamander robot, but can indeed be used for other purposes (for example the pectoral fins of the fish robot). Each limb element includes a pair of identical circuits (one for the left limb and one for the right one). The design is unlike a real animal limb: this element has an axis capable of continuous rotation as output (and thus only one degree of freedom), similarly to robots using whegs (i.e., wheel-legs, see Quinn et al. (2001); Saranli et al. (2001)). This gives to the element both flexibility (it can be used for other purposes than legs) and simplicity (only one motor and gearbox per limb).

These elements are based on the same electronics of the body elements, however, as the printed circuits are also used as mechanical support for the gears and the motor, the components are differently distributed between the circuits. Additionally, an infrared LED/phototransistor couple allows the detection of the absolute position of the output axis (using a hole in the last wheel of the gearbox), in order to automatically align it when powering up the robot.

3.3.3 Locomotion controller circuit

The locomotion controller circuit has been designed to meet the following criteria:

- To provide a simple but flexible locomotion controller with low energy consumption.

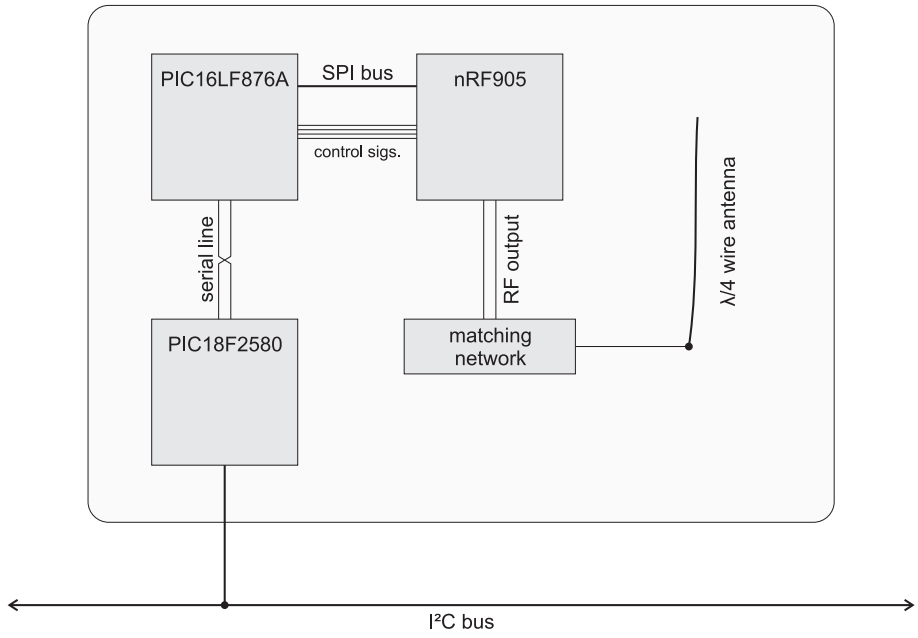


Figure 3.11: Block schema of the electronics of the locomotion controller circuit.

It should be possible to implement on it any control algorithm, following the needs of the user (in this thesis, CPG-based controllers have been implemented on it).

- To have bidirectional radio communication capabilities (both on ground and under water) for remote control and measure.

The circuit is placed inside an empty body element (i.e., without the motor and the gearbox); a variation of the same circuit without the radio communication functions has been used for controlling the BoxyBot fish robot (see chapter 5). A block schema of the controller electronics can be seen in figure 3.11. The circuit is based on a PIC18F2580 microcontroller, which is master on the I^2C bus of the robot. It can implement a locomotion controller (for example, a CPG) and sends out the setpoints to the motor controllers of each element in real time. The main microcontroller communicates, using a local serial line, with a PIC16LF876A microcontroller, which controls a nRF905 radio transceiver. The radio communication is handled by this separate microcontroller for simplicity, and because the PIC18F2580 can not handle hardware SPI and I^2C at the same time. The antenna is internal to the element and consists of a simple $\lambda/4$ wire (where λ is the wave length of the used frequency). The radio system uses the 868 MHz ISM band: preliminary experiments showed that a 10 mW signal (the power transmitted by the nRF905) on this frequency can penetrate in water up to at least 30 cm (the maximum tested depth). The more common 2.4 GHz band has not been used because it is heavily absorbed by the

water. The maximal bandwidth is approximately 50 kbps, largely enough to send control commands and parameters to the online trajectory generator.

The software running on the locomotion controller can easily be reprogrammed with an external programming connector placed on the element.

3.4 Simulation

Simulated models of robots created with the elements described in this chapter have been created with the physics based robot simulator Webots (Michel, 2004). They include a snake-like robot (see chapter 4) and a salamander robot (see chapter 6). Snapshots of the simulator are visible in figure 3.12.

These models can be controlled, for example, by one of the CPG-based controllers presented in the next chapters, and have mechanical and physical properties that are close to those of the real robots.

Having a physically realistic (although not perfect) simulation of the implemented robots is very useful, as it permits to rapidly test new control methods or parameters in a fraction of the time needed to do the same on the real robot, and without the inconveniences of doing this (e.g., battery charging, possible physical damages or water leakages, etc.). It also allows a rapid creation of new types of robots (by connecting together the elements in different ways), without any need to assemble the real elements before obtaining the wanted properties in the simulation.

The passive wheels used in the snake robot (see chapter 4) are modeled with asymmetric friction (simulated with a simplification of the Coulomb friction model):

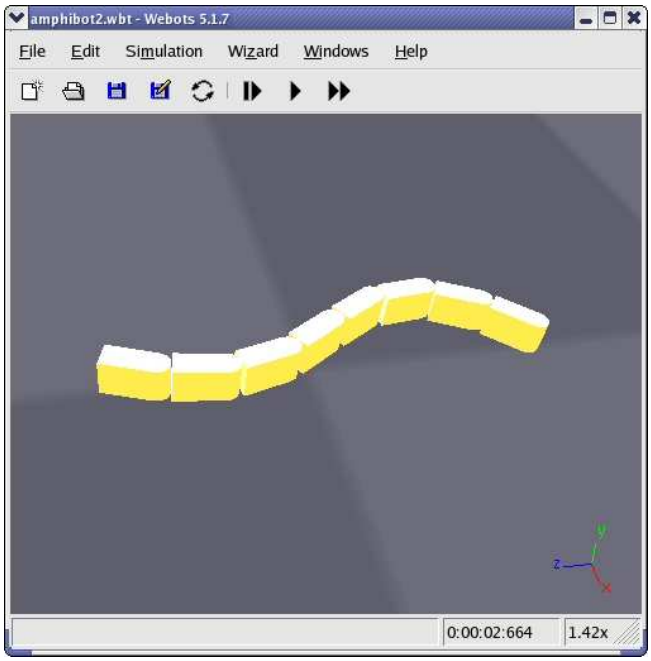
$$\begin{array}{rcl} F_{\perp} & = & -\mu_{\perp} \cdot F_N \cdot \frac{v_{\perp}}{|\vec{v}|} \\ F_{\parallel} & = & -\mu_{\parallel} \cdot F_N \cdot \frac{v_{\parallel}}{|\vec{v}|} \end{array} \tag{3.1}$$

The used friction coefficients are $\mu_{\perp} = 1.0$ and $\mu_{\parallel} = 0.05$. This friction model is only a first approximation of the real friction, and although the simulation is giving maximal velocities similar to the ones obtained in the reality, the underlying parameters are often quite different.

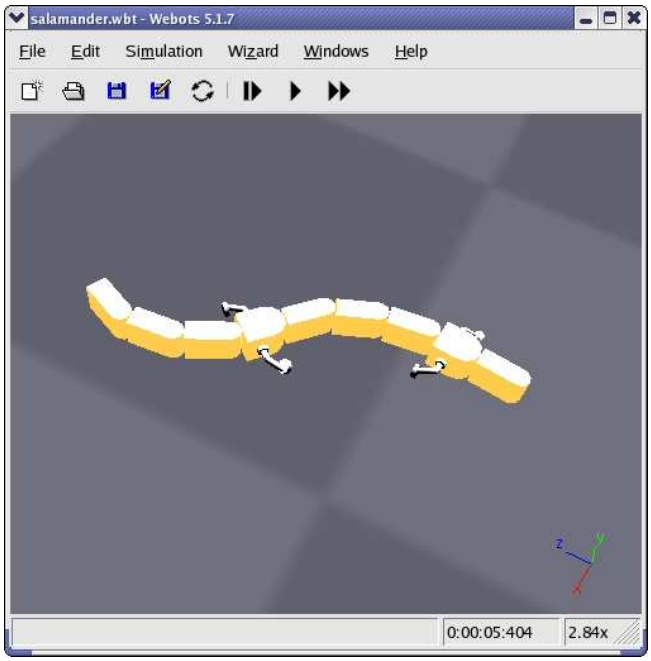
For swimming, the hydrodynamic forces applied to each element of the robot are the Archimedes' force $\overrightarrow{F_A}$ and the drag forces $\overrightarrow{F_D}$:

$$\begin{array}{rcl} \overrightarrow{F_A} & = & -V_{el} \cdot \rho_{water} \cdot \vec{g} \\ F_{D_d} & = & \frac{1}{2} \cdot \rho_{water} \cdot S_d \cdot C_d \cdot v_d^2 \ (d \in \{x, y, z\}) \end{array} \tag{3.2}$$

where V_{el} is the volume of the immersed part of the element, ρ_{water} is the density of the water, $\vec{g}$ is the gravity acceleration, S_d is the surface of the element's section perpendicular to the d axis, C_d is the drag coefficient relative to the d axis, and v_d is the speed of the element along the d axis. Although this model is relatively simple (e.g., it does not take



(a) Snake robot with 7 active elements.



(b) Salamander robot with 6 active elements and 4 legs

Figure 3.12: Snapshots of the robot simulation in Webots.

into account the turbulence generated in the water by the movement of the robot), it is enough for a qualitative analysis of the swimming motion.

3.5 Discussion

In this chapter, the mechanics and electronics of the developed robot elements (both the old and current ones) have been presented, explaining the main differences between the two generations (future work to improve the current elements is described in chapter 8). The simulation of the robot elements has also been briefly explained. Although this simulation is useful to obtain qualitative results, it still needs to be calibrated and enhanced (especially for swimming) if quantitative results are to be obtained, as the used friction and hydrodynamical models are too simple.

Chapter 4

Snake robot

This chapter describes the use of the elements presented in the previous chapter to build an amphibious snake-like robot that can both crawl on ground and swim in water. The results presented in this chapter have been published in Crespi and Ijspeert (2006); Ijspeert and Crespi (2007).

4.1 Snake locomotion

Four main different locomotion modes have been documented in snakes (Gray, 1946; Jayne, 1986; Dowling, 1997): lateral undulation (also called serpentine locomotion), concertina, sidewinding and rectilinear. Several other gaits exist, however they are used only by a restricted number of snake species in somewhat special situations (tree climbing, jumping, etc.). Sometimes, depending on the environment, snakes use more than one locomotion mode at the same time, having a locomotion mode for one part of the body and another one for the other part (Gans (1974) as cited by Jayne (1986)).

The *lateral undulatory* mode, characterized by a lateral S-shaped wave travelling from head to tail, is the most common and efficient one, and almost all snakes use it. Swimming snakes move the body practically in the same way (Jayne, 1985). This type of swimming is called *anguilliform swimming* among elongate fishes, such as eels and lampreys. In the *concertina* mode, part of the snake's body is pushed against a surface forming a small number of waves: by moving these waves, and the corresponding contact points, the snake progresses. This mode is generally used when the snake has to move along a straight path or when the friction coefficients of the floor do not allow lateral undulatory locomotion; however this is a rather inefficient mode and is seldomly used only when needed. *Sidewinding* is used by desert snakes that need to move on sand; in this mode, the snake lifts a part of the body to maintain only a few contact points with the ground, using them to move the rest of the body. The *rectilinear* mode is obtained by cyclically "fixing" parts of the skin to the ground using scales, then moving the backbone forward

with respect to the skin, and finally releasing the scales allowing the skin to move forward. This locomotion mode is generally used only by big snakes (like boas), because their weight makes the lateral undulation inefficient. As its name says, rectilinear locomotion does not produce lateral undulations like the other ones.

Our robot will use lateral undulatory locomotion. To achieve this type of locomotion, an issue is of fundamental importance: the friction coefficients between the snake and the ground have to be directional. For each segment of the body (a snake has as many segments as the number of vertebrae – between 100 and 400 depending on the species), there must be a low friction coefficient in the tangential direction (the direction in which the segment is moving) and a high friction coefficient in the perpendicular direction, in order to avoid lateral displacement of the segment. This directional friction is obtained in snakes by the particular structure of the skin. A similar mechanism is used when swimming: due to the elongate shape, propulsion is produced by the combination of a low drag coefficient in the tangential direction and a higher one in the perpendicular directions.

4.2 Central pattern generator model

The CPG model presented here has been developed together with Auke Ijspeert. It is based on a system of amplitude-controlled phase oscillators. The design of the CPG is loosely inspired from the neural circuit controlling swimming in the lamprey (Grillner et al., 1995): it spontaneously produces travelling waves with constant phase lags between neighboring segments along the body, and it is made of multiple oscillators connected as a double chain. An oscillator in the model corresponds to an oscillatory center in the lamprey, i.e., a subnetwork of several thousands of neurons located in one segment of the spinal cord that is capable of producing oscillations independently of other centers.

The CPG model is a double chain of oscillators with nearest neighbor coupling (figure 4.1). The chain is designed to generate a travelling wave, from the head to the tail of the robot. This wave is used to achieve anguilliform swimming in water and serpentine locomotion on ground. Compared to previous neural network models developed for the lamprey CPG (Ijspeert et al., 1999; Ijspeert and Kodjabachian, 1999), the model presented here is simpler (much fewer state variables) and therefore better suited for being programmed on a microcontroller on board of the robot, while keeping the essential features of lamprey travelling wave generation.

The total number of oscillators is $2N$ where $N = 7$ is the number of actuated joints in the robot. Actuated joints are numbered 1 to N from head to tail. Oscillators in the left chain of the CPG are numbered 1 to N and those on the right side are numbered $N + 1$ to $2N$ from head to tail.

The CPG is implemented as the following system of $2N$ coupled oscillators:

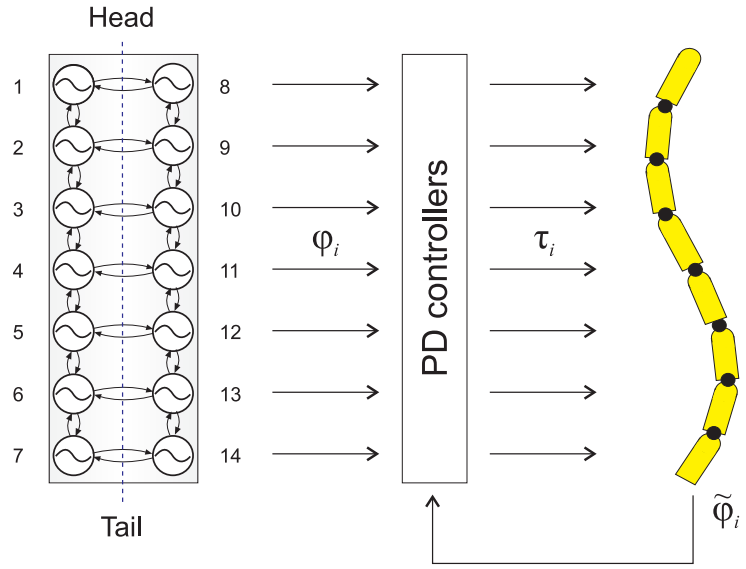


Figure 4.1: Structure of the CPG used in the robot.

$$\begin{cases} \dot{\theta}_i &= 2\pi\nu_i + \sum_j w_{ij} \sin\Big(\theta_j - \theta_i - \phi_{ij}\Big) \\ \ddot{r}_i &= a_i\Big(\frac{a_i}{4}(R_i - r_i) - \dot{r}_i\Big) \\ x_i &= r_i\Big(1 + \cos(\theta_i)\Big) \end{cases} \tag{4.1}$$

where the state variables θ_i and r_i represent, respectively, the phase and the amplitude of the i^{th} oscillator, the parameters ν_i and R_i determine the intrinsic frequency and amplitude, and a_i is a positive constant. The coupling between the oscillators is defined by the weights w_{ij} and the phase biases ϕ_{ij}. The variable x_i is the rhythmic and positive output signal extracted out of oscillator i. The first differential equation determines the time evolution of the phase θ_i. It can be shown that two (or more) coupled oscillators will synchronize (i.e., oscillate at the same frequency and with a constant phase lag) if the coupling weights w_{ij} are sufficiently large compared to the differences of intrinsic frequencies (see section 4.2.1). The phase lag between the oscillators will then depend on ϕ_{ij}, w_{ij} and ν_i. The second differential equation is a second order linear differential equation that ensures that the amplitude r_i smoothly converges to R_i in a critically dampened fashion.

The setpoints φ_i, i.e., the desired angles for the N actuated joints, are obtained by taking the difference between signals from the left and right oscillators. A standard PD motor controller is then used to compute the voltage τ_i (i.e., torque) applied to the motor

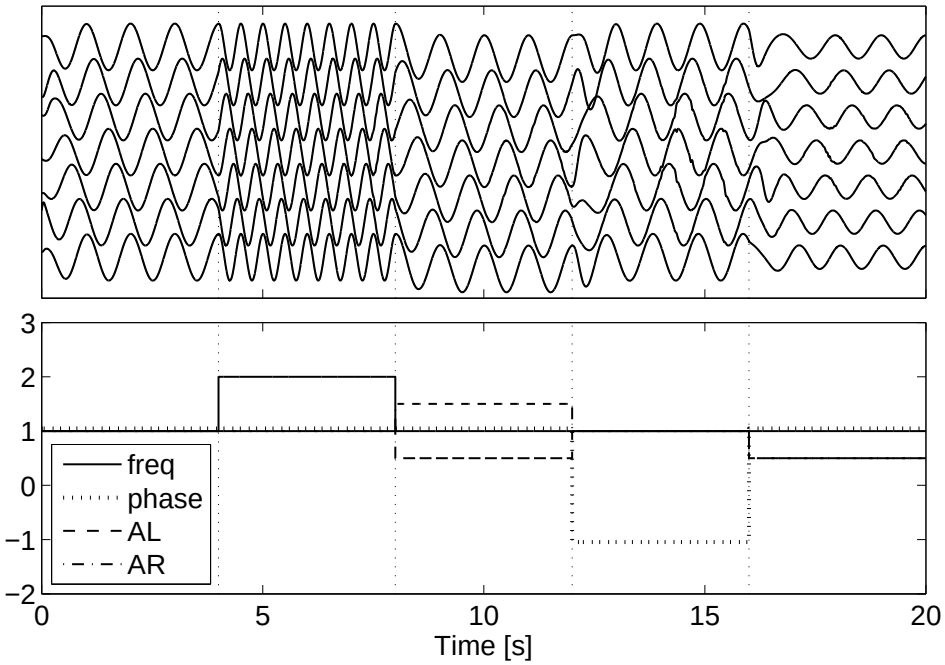


Figure 4.2: Effect of changing the parameters of the CPG. Top: setpoint signals, Bottom: control parameters. Initial parameters are $A_L = A_R = 1$, $\nu = 1$ Hz and $N \cdot \Delta\phi$=1. At $t = 4\,\mathrm{s}$, ν is temporarily changed to 2.0 Hz, at $t = 8\,\mathrm{s}$, A_L and A_R are temporarily changed to 0.5 and 1.5 respectively which leads to a negative offset of the setpoint oscillations. At $t = 12\,\mathrm{s}$, $N \cdot \Delta\phi$ is temporarily set to -1.0 which leads to a reversal of the direction of the travelling wave. At $t = 16\,\mathrm{s}$, A_L and A_R are changed to 0.5 which leads to reduced amplitude in the oscillations.

(using a PWM signal):

$$\begin{aligned} \varphi_i &= x_i - x_{N+i} \\ \tau_i &= K_p e_i + K_d \dot{e}_i \end{aligned} \tag{4.2}$$

where $e_i = \varphi_i - \tilde{\varphi}_i$ is the tracking error between the desired angles φ_i and the actual angles $\tilde{\varphi}_i$ measured by the motor incremental encoders, and K_p and K_d are the proportional and derivative gains.

Here, we chose the frequency parameters to be equal for all oscillators, i.e., $\nu_i = \nu$. We also chose the amplitude parameters on one side of the CPG to be an affine function of the maximal amplitude on that side: $R_i = \alpha_i \cdot A_L$ for the left side ($i = [1, ..., N]$) and $R_i = \alpha_{i-N} \cdot A_R$ for the right side ($i = [N+1, ..., 2N]$). The α_i parameters are linearly interpolated between the open parameter α_1 (head) and $\alpha_N = 1.0$ (tail). The phase biases ϕ_{ij} are chosen to be equal to π between left and right oscillators (i.e., these will oscillate in anti-phase). The phase biases between neighbor oscillators are set to $\Delta\phi$ for the descending connections and to $-\Delta\phi$ for the ascending connections. We used $w_{ij} = 4$ for all connections and $a_i = 100$ for all oscillators. The PD coefficients K_p and K_d are tuned manually for each element (e.g., elements in middle of the chain require larger gains than those at the extremities for good trajectory tracking).

With these settings, the CPG asymptotically converges to a limit cycle (see a skeleton of the proof in section 4.2.1) that is defined by the following closed form solution for the i^{th} actuated joint:

$$\varphi_i^{\infty}(t) = \alpha_i \cdot (A_L - A_R + (A_L + A_R) \cdot \cos(2\pi\nu \cdot t + i\Delta\phi + \phi_0)) \tag{4.3}$$

where ϕ_0 depends on the initial conditions of the system. This means that the system always stabilizes into a travelling wave which depends on the control parameters ν, $\Delta\phi$, A_L, A_R and α_i. Indeed the frequency, phase lag, amplitude and offset are directly determined by ν, $\Delta\phi$, $\alpha_i(A_L + A_R)$, and $\alpha_i(A_L - A_R)$, respectively. These parameters can be modified online by a human operator from a control PC using the wireless connection. The CPG will rapidly adapt to any parameter change and converge to the modified travelling wave after a short transient period. An example of how the CPG reacts to parameter changes can be observed in figure 4.2: when the parameters are changed, the oscillator smoothly converges to the new limit cycle, without any discontinuities in the outputs.

The differential equations are integrated by the microcontroller of the head using the Euler method, with a time step of 10 ms and using fixed point arithmetics.

4.2.1 Proof of convergence

The limit cycle of the CPG is determined by the time evolution of the amplitude and phase variables. We here show the particular case of two oscillators coupled bi-directionally with coupling weights $w_{12} = w_{21} = w$ and phase biases $\phi_{12} = -\phi_{12} = \Delta\phi$:

$$\begin{cases} \dot{\theta}_1 &= 2\pi\nu_1 + w\sin(\theta_2 - \theta_1 - \Delta\phi) \\ \ddot{r}_1 &= a(\frac{a}{4}(R_1 - r_1) - \dot{r}_1) \\ \dot{\theta}_2 &= 2\pi\nu_2 + w\sin(\theta_1 - \theta_2 + \Delta\phi) \\ \ddot{r}_2 &= a(\frac{a}{4}(R_2 - r_2) - \dot{r}_2) \end{cases} \tag{4.4}$$

It is easy to demonstrate that the state variables r_1 and r_2 asymptotically converge to R_1 and R_2, respectively, from any initial condition. Since we are interested in determining whether these two oscillators will synchronize (i.e., evolve with a constant phase difference), and, if yes, with which phase difference, it is useful to introduce the phase difference $\psi = \theta 2 - \theta 1$. The time evolution of the phase difference is determined by

$$\dot{\psi} = f(\psi) = \dot{\theta}_2 - \dot{\theta}_1 = 2\pi(\nu_2 - \nu_1) - 2w\sin(\psi - \Delta\phi) \tag{4.5}$$

If the oscillators synchronize, they will do so at the fixed points ψ_∞ (i.e., points where $f(\psi_\infty) = 0)$):

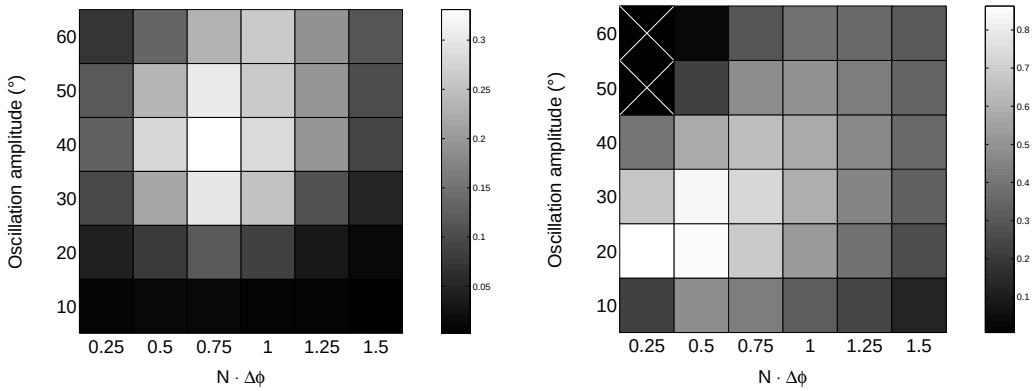
$$\psi_\infty = \arcsin(\frac{\pi(\nu_2 - \nu_1)}{w}) + \Delta\phi \tag{4.6}$$

In our case we have $\nu_1 = \nu_2 = \nu$, and this equation has a single solution $\psi_\infty = \Delta\phi$. This solution is asymptotically stable because $\partial f(\psi_\infty)/\partial\psi < 0$. The outputs of the oscillators therefore asymptotically converge to oscillations that are phase-locked with a phase difference of $\Delta\phi$: $x_1^\infty(t) = R_1(1 + \cos(2\pi\nu t + \phi_0))$ and $x_2^\infty(t) = R_2(1 + \cos(2\pi\nu t + \Delta\phi + \phi_0))$ where ϕ_0 is a constant that depends on initial conditions. Since the complete CPG is made of multiple bi-directionally coupled oscillators and that all parameters ϕ_{ij} are consistent (i.e., the sums of the parameters ϕ_{ij} are equal to a multiple of 2π on any closed path between oscillators), the same reasoning can be recursively applied to demonstrate convergence of the complete CPG.

4.3 Locomotion characterization

The results presented in this section have been published in Crespi and Ijspeert (2006); Ijspeert and Crespi (2007).

Serpentine locomotion and anguilliform swimming require a travelling wave to be propagated from the head to the tail of the robot. In this section, we analyze the results of our experiments, where we systematically tested how the three control parameters of the CPG (amplitude A, phase difference $\Delta\phi$ and frequency ν) affect the locomotion speed of the robot, both on ground and in water. The parameters have been kept into a reasonable range: the amplitude between $\pm 10°$ and $\pm 60°$ (with a step of $10°$), the frequency between 0.2 Hz and 1.0 Hz (with a step of 0.2 Hz) and the phase difference between $0.25/N$ and $1.5/N$ (with a step of $0.25/N$). For the lowest phase ($0.25/N$), the amplitude has been limited to $\pm 40°$, as a greater amplitude would lead to collisions between the elements of the robot at some points of the cycle.



(a) Crawling (b) Swimming

Figure 4.3: Speeds (m/s) for the simulated systematic tests at $\nu = 1.0$ Hz.

4.3.1 Crawling and swimming in simulation

The results of the simulated systematic tests at $\nu = 1.0$ Hz (this frequency corresponds to the observed maximal speed for both modes) are plotted in figure 4.3. The maximal speed obtained for crawling is 0.33 m/s, with $N \cdot \Delta\phi = 0.75$ and $A = \pm 40°$. For swimming, the maximal locomotion speed is 0.87 m/s, with $N \cdot \Delta\phi = 0.25$ and $A = \pm 20°$. For both modes, the optimum is relatively peaked (meaning that a small distance from the optimum causes a significant drop of the speed), without any local speed maxima. We systematically tested locomotion at other frequencies and found that the optimal amplitude and phase difference values for a given frequency slightly change compared to the tests with $\nu = 1.0$ Hz (data not shown). The frequency seems to have a direct, almost linear, effect on the locomotion speed (see figure 4.4), slightly influencing the optimal amplitude and phase parameters for crawling, and not influencing them at all for swimming.

4.3.2 Crawling with the real robot

For each parameter set, the speed has been measured as follows: the robot was placed on a wooden floor, in front of an horizontal line, then started, and was stopped manually when reaching another line, parallel to the first one, at a distance of 2.00 m. The exact time between the start and stop commands was measured by the controlling PC. If the locomotion was visibly bad (i.e., if the first line was not reached after approximately 40 s), the robot was stopped before reaching the end line and the distance manually measured.

Depending on the parameters, the locomotion speed varied between 0 and 0.40 m/s (0.52 body lengths/s). In the tested range, the speed clearly increased with the increase of the frequency, and the optimum was always obtained with an amplitude of $\pm 30°$. The optimal phase difference clearly depends on the frequency, moving from $N \cdot \Delta\phi = 0.5$ ("C" shape undulation) for $\nu = 0.2$ Hz to $N \cdot \Delta\phi = 1.0$ ("S" shape undulation) for $\nu = 1.0$ Hz.

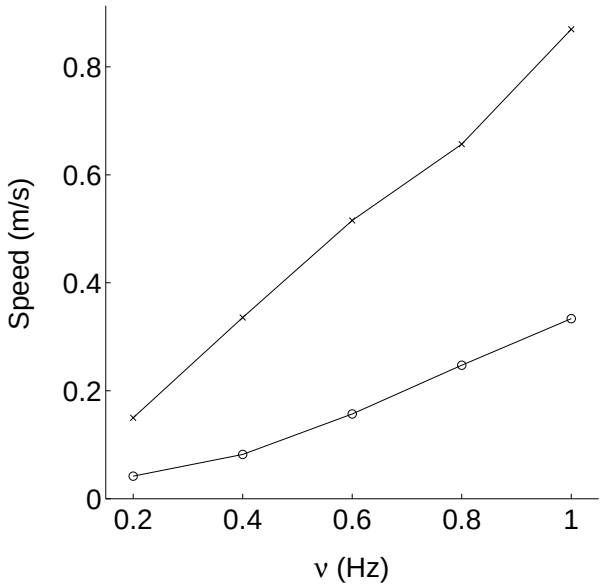


Figure 4.4: Effect of the frequency on the simulated speeds (∘: crawling at $A = \pm 40°$ and $N \cdot \Delta\phi = 0.75$; ×: swimming at $A = \pm 20°$ and $N \cdot \Delta\phi = 0.25$).

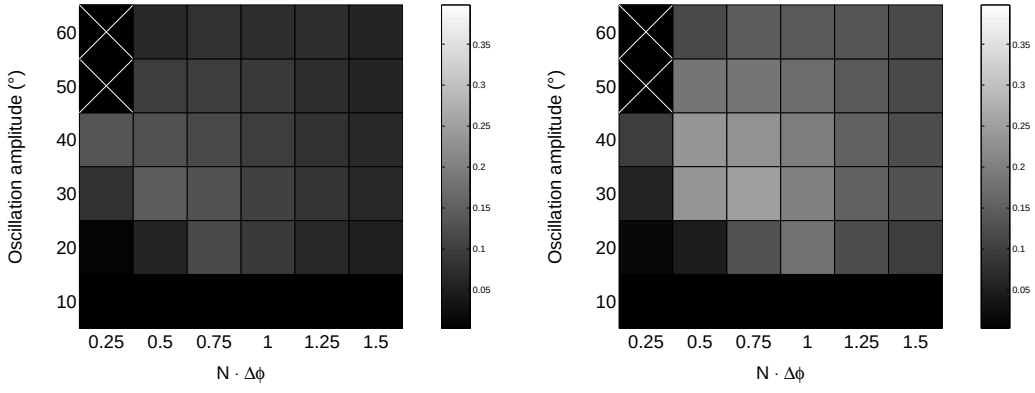
This effect was also visible in the simulation, but less important. Snapshots from the video of the locomotion producing the maximal crawling speed are visible in figure 4.6.

The complete results are presented in figure 4.5. Running the robot with $A = \pm 10°$ produces practically no locomotion, as the passive wheels tend to bend around their axis and slip on the ground. The optimum is clearly peaked, and there are no local maxima. This is interesting, as it suggests that relatively simple online optimization algorithms could be used to adapt the CPG parameters to the current environment in which the robot is moving (this will be addressed in chapter 7).

Compared to the simulation, the obtained maximal speed is 1.2 times larger. The difference is mostly due to the fact that the simulated friction model is only an approximation of the passive wheels used on the robot. Furthermore, the speed of locomotion is sensitive to friction coefficient values. For example, a change of the $\mu_{\|}$ friction coefficient of the simulation from 0.025 to 0.05 decreases the maximum obtained speed to 0.25 m/s (i.e., a drop of 24%). Another difference is that with the real robot, the optimal phase was correlated to the frequency; this effect was not visible in simulation and is likely to have the same origin of the observed speed difference (i.e., the approximated friction model).

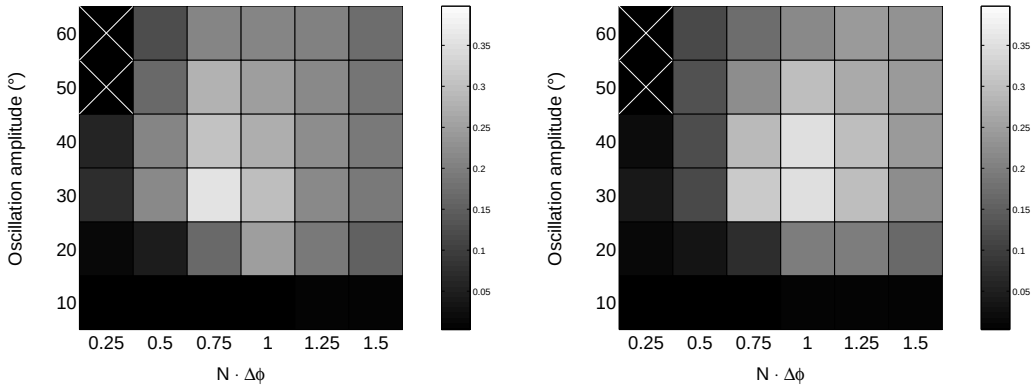
4.3.3 Swimming with the real robot

The speed for a given set of parameters was measured as follows: the robot was started at the beginning of the aquarium. A chronometer was started when it reached a first



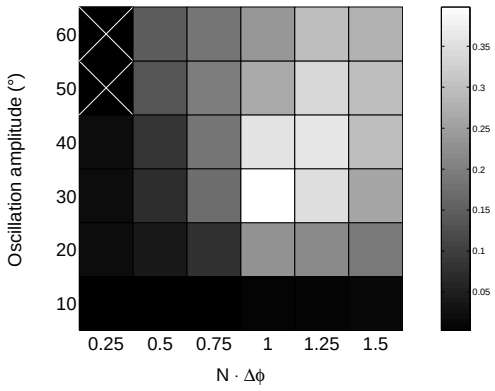
(a) $\nu = 0.2$ Hz

(b) $\nu = 0.4$ Hz



(c) $\nu = 0.6$ Hz

(d) $\nu = 0.8$ Hz



(e) $\nu = 1.0$ Hz

Figure 4.5: Real robot crawling locomotion speeds (in m/s) with the different parameters.

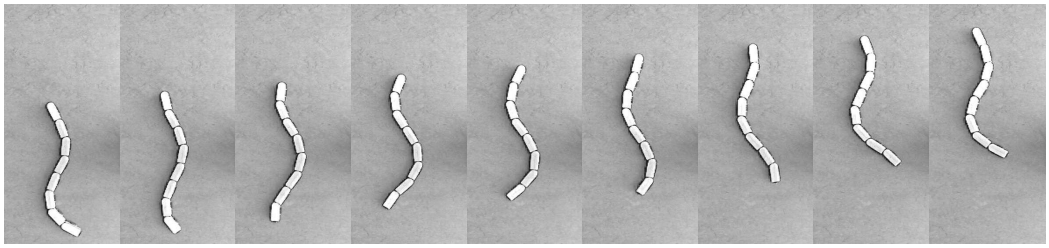


Figure 4.6: The robot crawling at $A = \pm 30°$, $N \cdot \Delta\phi = 1.0$ and $\nu = 1.0$ Hz. The time step between the snapshots is 0.12 s.

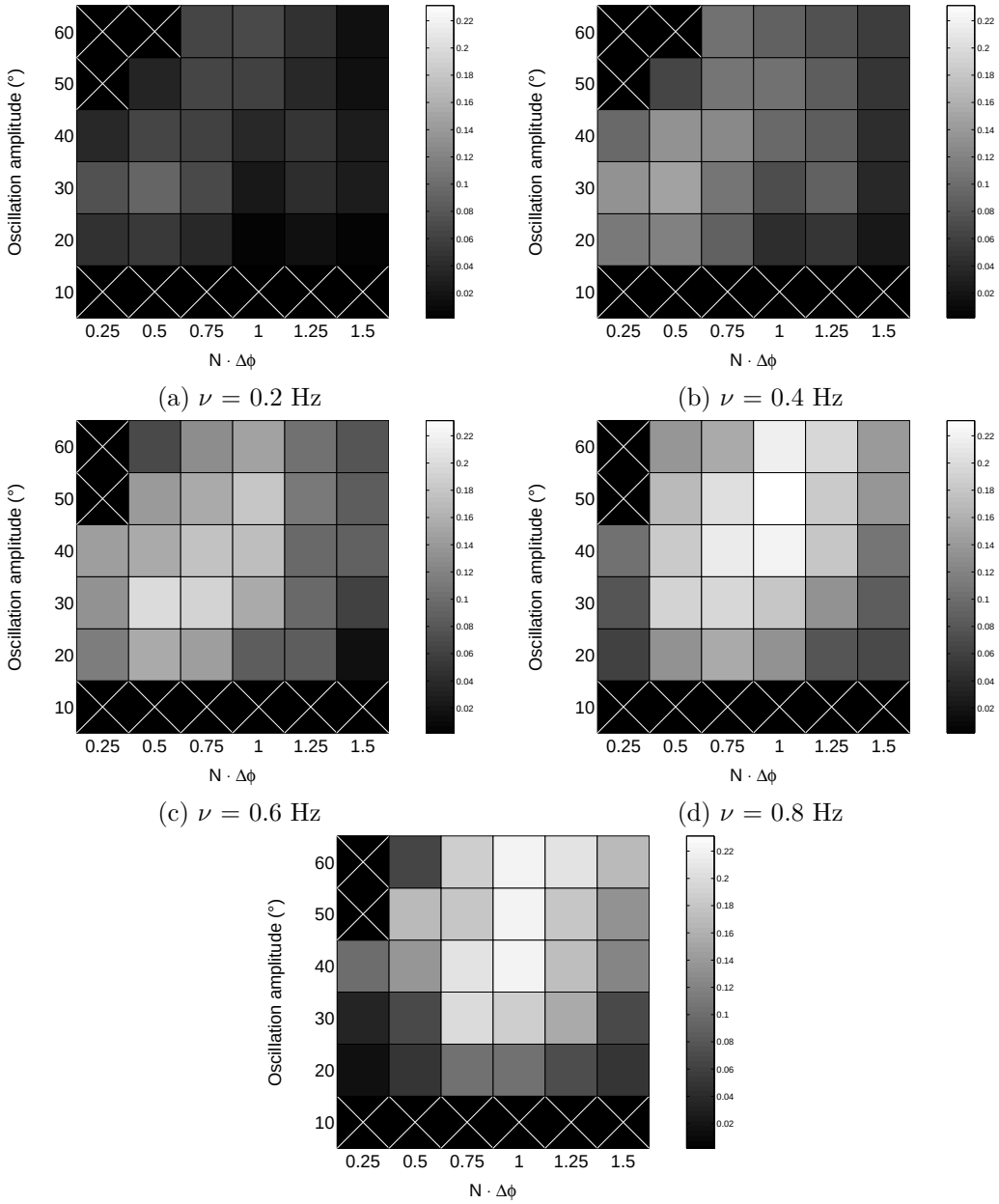
horizontal line placed at 1.15 m from the border, and stopped when crossing the second line, placed at 1.00 m from the first one. For some parameters producing very low speeds, the chronometer has been stopped before the second line and the distance manually measured. No measure has been taken for $A = \pm 10°$, as the robot did not stay vertically in this configuration, due to small asymmetries of the center of mass. A small aquarium pump has been used to inject low pressure air inside the robot (through a highly flexible silicon tube) to reduce the risk of water leakages due to the extensive use of the robot during these tests.

Depending on the parameters used, the swimming speed varied between 0 and 0.23 m/s (0.30 body lengths/s). The optimal amplitude was $A = \pm 30°$ at low frequencies, moving to $A = \pm 50°$ for $\nu \geq 0.8$ Hz. Similarly, the optimal phase moves with the frequency: the optima at $\nu < 0.8$ Hz have $N \cdot \Delta\phi = 0.5$, and those at $\nu \geq 0.8$ Hz have $N \cdot \Delta\phi = 1.0$. The complete results are plotted in figure 4.7. The optimum is peaked, although less remarkably than in crawling. Snapshots from the video of the locomotion producing the maximal swimming speed are visible in figure 4.8.

Compared to the results of the simulation, the maximal swimming speed appears to be almost 4 times lower; this is not really surprising, as a really simplified hydrodynamic force model has been used and more work is needed to calibrate the drag coefficient values. Even if the position of the optimum is not the same, especially at high frequencies, the structure of the results is similar, although the existence of small local maxima at $\nu < 0.6$ Hz has not been predicted by the simulation and is maybe the effect of turbulence induced by the robot when swimming.

4.4 Interface for the control parameters

More details on the results presented in this section can be found in Ijspeert and Crespi (2007). To simplify the control of the robot by a human operator, it is useful to reduce the number of commands to two, one for speed and one for direction, instead of the four



(a) $\nu = 0.2$ Hz

(b) $\nu = 0.4$ Hz

(c) $\nu = 0.6$ Hz

(d) $\nu = 0.8$ Hz

(e) $\nu = 1.0$ Hz

Figure 4.7: Real robot swimming locomotion speeds (in m/s) with the different parameters.

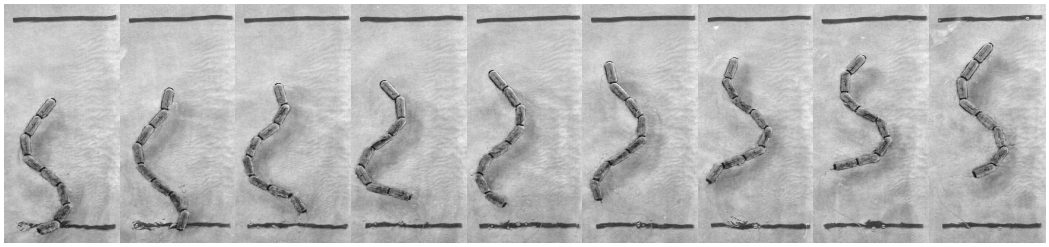


Figure 4.8: The robot swimming at $A = \pm 50°$, $N \cdot \Delta\phi = 1.0$ and $\nu = 0.8$ Hz. The time step between the snapshots is 0.16 s.

control parameters for the CPG.[1]

Turning can be induced by modulating $A_L - A_R$, i.e., by adding offsets to the setpoint oscillations. The robot will then make undulations around a bent posture and turn towards the side with higher amplitude. We can therefore introduce the turning command T which determines the difference between left and right amplitudes normalized by the total amplitude, namely $T = \frac{A_L - A_R}{A_L + A_R}$.

As we have seen in section 4.3, the control of speed is more difficult because the speed of locomotion depends jointly on the frequency ν, the amplitude $A = A_L + A_R$ and the phase lag $\Delta\phi$ of the travelling wave, as well as on the type of environment (e.g., the type of friction with the ground, the slope, etc.). The outcome of the study is that, in the explored parameter space, the speed of locomotion always monotonically increases with the frequency when the two other parameters are kept fixed at any value. The amplitude and phase lag show a more complex, non-monotonic, influence on the speed. For a given frequency, for instance, the dependence of speed on the amplitude and phase parameters is a smooth function with a single optimum. The location of the optimum varies with the frequency. For instance, on ground with $\nu = 0.2$ Hz the maximum speed (0.15 m/s) is obtained with $A = 30°$ and $\Delta\phi = 0.5/N$, while at $\nu = 1.0$ Hz the maximum speed (0.40 m/s) is obtained with $A = 30°$ and $\Delta\phi = 1.0/N$. In order words, with our robot it is better to make C-shaped undulations ($\Delta\phi = 0.5/N$) at low frequencies and S-shaped undulations ($\Delta\phi = 1.0/N$) at higher frequencies. The same is true for swimming.

We therefore choose the frequency as a single command parameter for speed, and design two functions $[A, \Delta\phi] = f_{ground}(\nu)$ and $[A, \Delta\phi] = f_{water}(\nu)$ for setting the amplitude and phase lag for a given frequency. These piece-wise linear functions are simple linear interpolations between the observed optima. The functions thus ensure that the travelling wave produced at a given frequency remains close to the fastest locomotion at that frequency. When the robot makes a transition from ground to water or vice-versa, the

[1]Note that by design the robot is only capable of planar locomotion and therefore does not require control of vertical motion.

human operator makes a manual switch from one function to the other.[2] Note that, since these functions depend on the environment, it is useful to implement online optimization algorithms to identify good parameters for a given (possibly unknown) environment, instead of systematic search (see chapter 7).

With this interface, the speed and direction of locomotion of the snake robot can now easily be adjusted in real-time by a human operator by setting the frequency ν and the turning command T.

4.4.1 Control of direction on ground

To evaluate the turning ability of the robot on ground, we used video tracking of a green LED mounted on the head of the robot. When a non zero turning command T is sent to the robot, it will on average progress on a circle. Figure 4.9 shows the trace that the head element makes. A circle is fitted to the outer bounds of the trace to provide an estimation of the turning ability: the shorter the radius R, the sharper the turning. Figure 4.10 shows how the inverse of the radius varies with the T command. Interestingly, the relation between $1/R$ and T is almost linear. The sharpest turning is obtained at T=1, where the radius of the curvature is 25 cm. Turning is therefore quite sharp for a 72 cm long robot.

4.4.2 Results

We systematically tested the speed of serpentine locomotion using different values of our command parameter ν. Figure 4.11 shows the resulting values. The speed is evaluated by measuring the time needed by the robot to travel a given distance (1 m), and repeating the measure four times.

Results show that the speed increases monotonically with ν. The highest speed at 1.0 Hz is approximately 0.4 m/s, which corresponds to 0.55 bodylength/s. Higher speeds can be reached at higher frequencies, but tracking errors in the PD controllers become significative above 1.0 Hz due to motor torque limits. As explained in Section 4.4, because of the function f_{ground}, the types of undulations are quite different between low frequencies where the undulations make C-shapes ($\Delta\phi$=0.5/N) and high frequencies where the undulations make S-shapes ($\Delta\phi$=1.0/N). Note that while the speed measures have been made at fixed ν values, ν can be continuously and interactively adjusted to produce locomotion with smooth accelerations and decelerations.

Similarly to locomotion on ground, we tested how the speed of swimming depends on the command ν. Speed was measured by taking the time necessary to travel a given distance. Since accelerations are slower in water than on ground, we waited enough time (approximately 5 seconds) before the beginning of the measurement such as to be close

[2] In the future, this switch will be done automatically using an external water sensor.

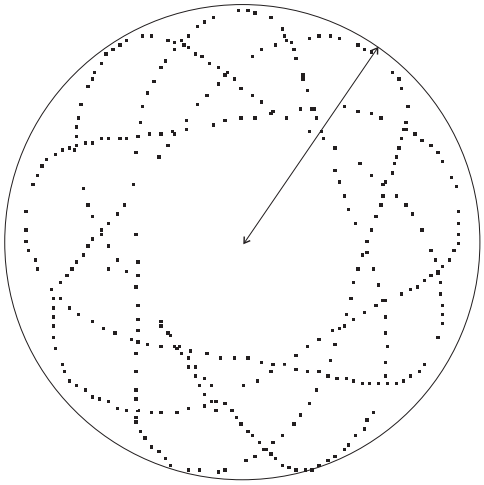


Figure 4.9: Tracking of the robot while turning on ground. The dotted line is the trace left by the head element. The radius of the circle fitting the outer bounds is used to measure the curvature.

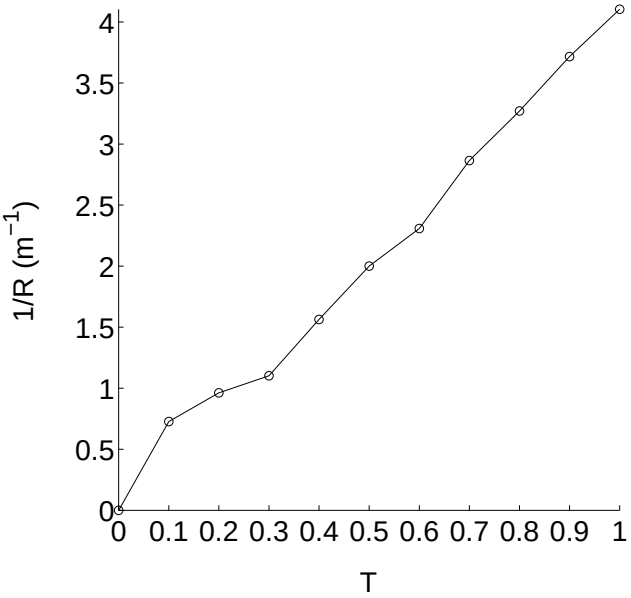


Figure 4.10: Control of direction during serpentine locomotion. The horizontal and vertical axes are respectively the turning command T and the inverse of the radius $1/R$.

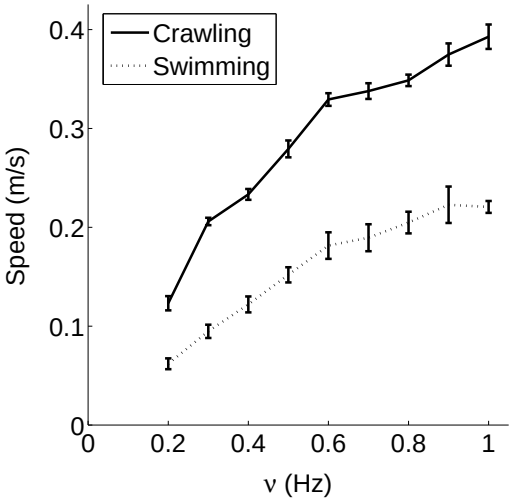


Figure 4.11: Control of speed during serpentine crawling (continuous line) and swimming (dotted line). Each data point is the average of 4 speed measures, and the error bars correspond to the standard deviation.

to steady-state swimming. Figure 4.11 shows the results of the measurements. Speed increases monotonically with ν up to $\nu = 0.9$ Hz, where it saturates. A maximum speed of 0.23 m/s, i.e., 0.32 bodylengths/s, is attained. Compared to serpentine locomotion, the speeds are lower and the measurements show a larger variability, which is related to the fact that water in motion makes experiments less reproducible because of the complex dynamics of waves bouncing against the small swimming pool walls.

4.5 Discussion

The systematic tests showed (1) that the parameters producing the optimal velocity of the robot for a given environment are peaked (which is meaning that small deviations of the parameters from this optimum rapidly result in suboptimal gaits), (2) that the resulting speed is directly proportional to the oscillation frequency, and (3) that the position of the optimum (in terms of oscillation amplitude and phase difference) depends on the frequency.

Using these results, we showed that it is possible to implement an interface function which allows a users to interactively modulate the speed and direction of the robot without having to deal with too many parameters (the speed is directly controlled by varying the frequency and setting the other parameters appropriately). This type of interface function, for instance, simplifies the control of the robot to a human user, who can simply control the speed and direction of locomotion, without dealing with the underlying loco-

motion parameters. A restriction of this interface function is that it is limited to a given environment (although it is possible to have multiple interface functions —one for each possible environment— this implies that systematic tests would have to be carried out for each of these environments). This limitation can be overcome with the use of online optimization techniques (see chapter 7).

Chapter 5

Fish robot

This chapter describes a fish robot, inspired from the Boxfish (fishes belonging to the Ostraciidae family). It has been built as a semester project, using the robot elements described in chapter 3, by Daisy Lachat. Subsequently, a new controller implementing several random behaviours, and user interaction has been developed by Ariane Pasquier for a semester project, the aim of which was adapting the robot to the public exhibition ("forum découvertes") at the School of Computer and Communication Sciences at EPFL. My contribution to this project as been the supervision of the semester projects, as well as the development of the electronics and PC programs used for the exhibition. The results presented in this chapter have been published (Lachat et al., 2006) or submitted for review (Crespi et al., 2007).

The main goal motivating the construction of this robot is to show that the elements described in chapter 3 could be used to build other types of robots than snakes and salamanders. Another goal is to use sensory information to close the control loop, to obtain a simple phototaxis behaviour, obstacle avoidance, and reaction to user knocks.

5.1 Hardware description

The fish robot, that has been named BoxyBot, is built using two of the elements described in chapter 3: a body element, connected to the caudal fin, and a limb element, used for pectoral fins. All the fins are realized using cut PVC and rubber glued together. The circuits inside the limb element have been slightly adapted to include an ADXL203 two-axis accelerometer and a PIC18F2580 microcontroller (master on the I^2C bus and running the control program implementing the controller described in the following section), which has several inputs. Two TSLG257 light sensors and a simple contact sensor have also been added and are connected to the microcontroller.

[1]distributed under Creative Commons license, see `http://creativecommons.org/licenses/by-nc-nd/2.0/`

Figure 5.1: A real boxfish (picture: Adrian Moody)[1].

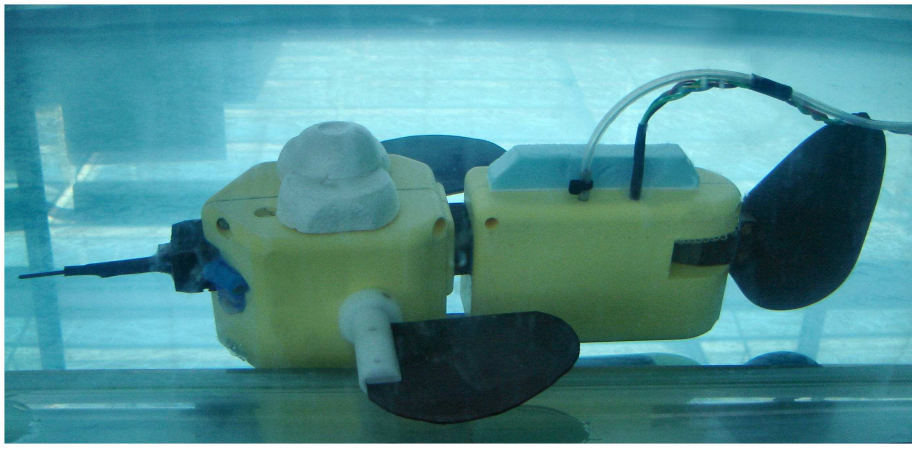


Figure 5.2: The BoxyBot fish robot in its aquarium.

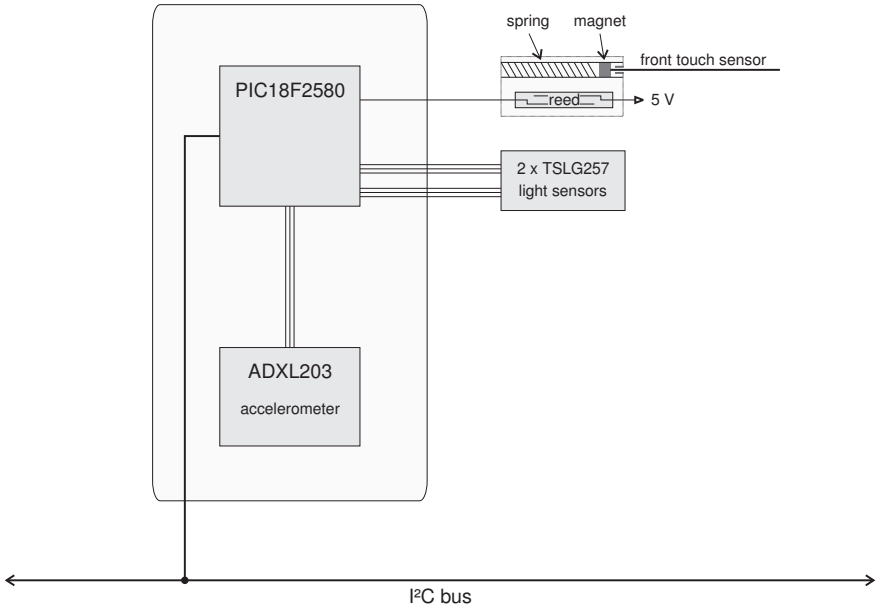


Figure 5.3: Additional hardware mounted in the limb module (pectoral fins) of the fish robot.

Accelerometer The two-axis accelerometer, mounted vertically, is connected to two analog inputs of the PIC18F2580 microcontroller through a low-pass filter to minimize high-frequency noise on the measures. It allows the control program of the robot to measure the accelerations around the pitch and roll axes; this information could for instance be used to determine the inclination of the robot with respect to the ground.

Light sensors A couple of TSLG257 sensors is mounted inside two plastic tubes sealed with silicone and fixed externally at the front of the robot. The field of vision of the sensors is partially overlapped, each sensor measuring the intensity level of the light at one side. The sensitivity of the sensor is reduced by placing two blue non-shrunk shrink gains over the tubes. The obtained sensitivity greatly reduces the impact of ambient light (except for direct sunlight), but requires the use of strong light sources (e.g., halogen lamps). These light sensor will be used for phototaxis (light tracking).

Contact sensor The contact sensor allows the robot to detect obstacles in front of it, and thus to change its behaviour accordingly. This sensor is built using a small reed contact placed in a sealed tube and connected to a power source. This tube is placed at the side of a system composed by a spring, a magnet and a thin PVC cylinder (see figure 5.3). In normal conditions (i.e., when the PVC cylinder is not pressed) the magnet

is far enough from the reed contact not to close it. When an obstacle is touched (pressing on the PVC cylinder), the magnet moves at the side of the reed contact, thus closing it and allowing the microcontroller to detect the obstacle.

5.2 Locomotion control

The locomotion controller is composed of a CPG model for producing coordinated oscillations extended by a finite state machine for modulating the CPG activity and implementing various locomotor behaviours.

5.2.1 CPG model

The locomotion controller is based on a CPG model implemented as a system of three coupled amplitude-controlled phase oscillators, one per fin (figure 5.5). This oscillator is very similar to the one used for the snake robot (see chapter 4), with the main difference that there is an explicit offset variable (e.g., for turning), instead of having two oscillators per degree of freedom, and that the connections between the oscillators are different. This limits the total number of oscillators (and thus of state variables), simplifying the implementation on a microcontroller.

An oscillator i is implemented as follows:

$$\begin{aligned} \dot{\phi}_i &= \omega_i \\ &+ \sum_j (w_{ij}\, r_j \sin(\phi_j - \phi_i - \varphi_{ij}) \end{aligned} \tag{5.1}$$

$$\ddot{r}_i = a_r(\frac{a_r}{4}(R_i - r_i) - \dot{r}_i) \tag{5.2}$$

$$\ddot{x}_i = a_x(\frac{a_x}{4}(X_i - x_i) - \dot{x}_i) \tag{5.3}$$

$$\theta_i = x_i + r_i \cos(\phi_i) \tag{5.4}$$

where θ_i is the oscillating set-point (in radians) extracted from the oscillator, and ϕ_i, r_i, and x_i are state variables that encode respectively the phase, the amplitude, and the offset of the oscillations (in radians). The parameters ω_i, R_i, and X_i are control parameters for the desired frequency, amplitude and offset of the oscillations. The parameters w_{ij} and φ_{ij} are respectively coupling weights and phase biases which determine how oscillator j influences oscillator i. The parameters a_r and a_x are constant positive gains ($a_r = a_x = 20$ rad/s). The reference position (i.e., corresponding to a zero offset) for the pectoral fins is when these fins are turned backwards in a horizontal position. The reference position for the caudal fin is when that fin is in the sagittal plane.

These equations were designed such that the output of the oscillator θ_i exhibits limit cycle behaviour, i.e., produces a stable periodic output. Equation 5.1 determines the time evolution of the phases of the oscillators. Here, we use the same frequency parameter $\omega_i = \omega$ for all oscillators. The coupling parameters are $w_{ij} = 0.5$ Hz, $\varphi_{ij} = 0.0$ Hz for all $i \neq j$ and $w_{ii} = 0.0$ Hz, $\varphi_{ii} = 0.0$ Hz otherwise (i.e., there are no self-couplings). Oscillators 1,2,3 respectively correspond to the left-pectoral, right-pectoral, and caudal fins. With these parameters, the phases will converge to a regime in which the phases grow linearly with a common rate ω and with a zero phase difference between all three oscillators (i.e., $\Delta\phi_{ij} = \varphi_{ij} = 0.0$) from almost any initial conditions.[2]

Equations 5.2 and 5.3 are critically damped second order linear differential equations which have respectively R_i and X_i as stable fixed points. From any initial conditions, the state variables r_i and x_i will asymptotically and monotonically converge to R_i and X_i. This allows one to smoothly modulate the amplitude and offset of oscillations.

With these settings, the CPG therefore asymptotically converges to a limit cycle $\theta_i^{\infty}(t)$ for the i^{th} actuated joint that is defined by the following closed form solution:

$$\theta_i^{\infty}(t) = X_i + R_i \cdot \cos(\omega t + \phi_0) \tag{5.5}$$

where ϕ_0 depends on the initial conditions of the system. This means that the system stabilizes into oscillations that are synchronous for all three degrees of freedom, and that can be modulated by 7 control parameters, namely ω for setting the common frequency, X_i ($i \in \{1, 2, 3\}$) for setting the individual amplitudes, and R_i ($i \in \{1, 2, 3\}$) for setting the individual offsets. Figure 5.4 illustrates how the system converges to the stable oscillations starting from random initial conditions and after a random perturbation.

Such a CPG model has several nice properties. The first interesting property is that the system exhibits limit cycle behaviour, i.e., oscillations rapidly return to the steady-state oscillations after any transient perturbation of the state variables (Figure 5.4). The second interesting property is that this limit cycle has a closed form solution.[3] The function is sine-based and has control parameters (ω, R_i, and X_i) that are explicit and are directly related to relevant features of the oscillations. This facilitates the design of locomotion controllers. A third interesting property is that these control parameters can be abruptly and/or continuously varied while inducing only smooth modulations of the set-point oscillations (i.e., there are no discontinuities nor jerks). This property will extensively be used in the Results section for varying the locomotor behaviours (Section 5.3). Finally, a fifth interesting feature is that feedback terms can be added to Equations 1-3

[2]The only exceptions are initial conditions in which two oscillators i, j are exactly in phase, i.e., $\Delta\phi_{ij} = \phi_j - \phi_i = 0$, and the third oscillator k is exactly in antiphase, i.e., $\Delta\phi_{ik} = \pi$. For those conditions, the system evolves to a regime which keeps these particular phase differences. In other words, this particular case represents an unstable fixed point for the differential equations that determine the time evolution of the phase differences.

[3]Very few types of oscillators have a closed form solution for their limit cycle.

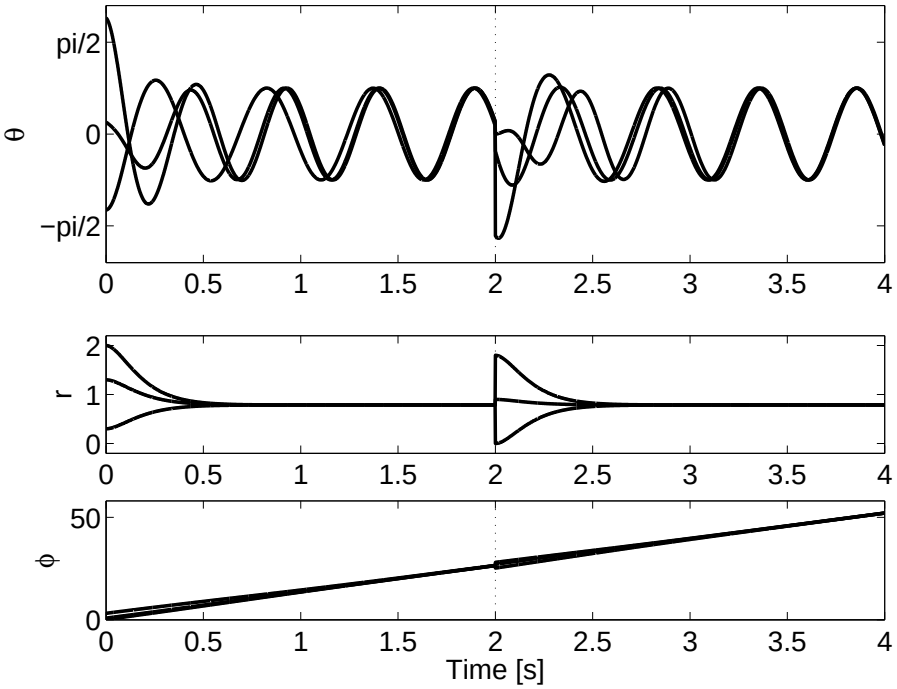


Figure 5.4: Limit cycle behaviour of the CPG. Starting from random initial conditions, the system quickly stabilizes in synchronous oscillations with controlled amplitude. A $t = 2s$, random perturbations are applied to the state variables ϕ_i and r_i, and the system rapidly returns to the steady state oscillations.

in order to maintain entrainment between control oscillations and mechanical oscillations (however this will not be explored here).

5.2.2 Complete control architecture

The diagram of the complete control architecture is given in Figure 5.5. The CPG model produces the set-points θ_i for PD controllers of the three fins. Different locomotor behaviours can be obtained by modulating the CPG control parameters ω, R_i, and X_i for the three fins.

Examples of locomotor behaviours include:

- swimming *forwards*, by oscillating only the caudal fin, both pectoral fins, or all fins, with all offsets X_i set to zero.
- swimming *backwards*, by turning the pectoral fins forward (i.e., by setting the pectoral offsets X_1 and X_2 to π) and stopping the oscillations of the caudal fin ($R_3 = 0$).
- *spinning* around the roll axis, by setting the pectoral offsets X_1 and X_2 to $\pi/2$ and $-\pi/2$ (i.e., by turning one pectoral fin up and the other down).
- *turning* (around the yaw axis) while swimming, by having a non zero offset X_3 for the caudal fin.
- *turning on the spot*, by oscillating the pectoral fins, with one of the pectoral offset to π.
- swimming *up* (or *down*), by setting an offset for both pectoral fins ($X_1 = X_2$) between 0 and $\pi/2$ ($-\pi/2$), proportionally to the desired vertical speed.
- *crawling*, by stopping the oscillations of the fins ($R_1 = R_2 = R_3 = 0$), and applying a continuously increasing offset (X_1 and X_2) to both pectoral fins. Two possibilities are with $X_1 = X_2$ (both pectoral fins rotate in phase) or $X_1 = X_2 + \pi$ (pectoral fins rotate in anti-phase).

For all these behaviours, the speed of locomotion can be varied by adjusting the frequency ω and/or the amplitudes R_i of oscillations. Typically the speed of locomotion increases with those parameters until the torque limits of the motors are reached.

We made two types of experiments for testing these different locomotor behaviours. In a first set of experiments, the choice of behaviour is done sequentially in a prefixed order without sensory inputs to test the different locomotor behaviours and the transitions between them.

In a second set of experiments, the behaviour controller is programmed as a finite state machine to implement a simple phototaxis both in water and on the ground. A strong

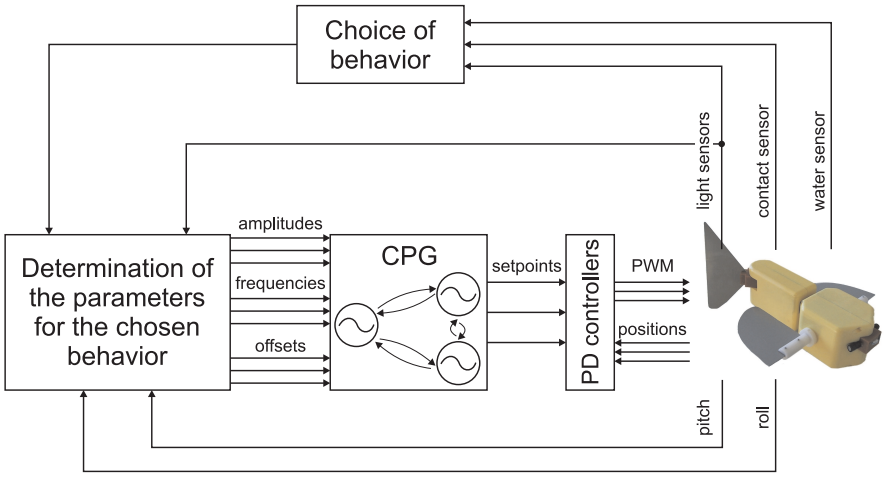


Figure 5.5: Diagram of the complete control architecture. While using a predefined behaviour the values from light sensors are not used. The values of pitch and roll were not used during the experiments described in this paper.

halogen lamp is used as a movable light source and a behaviour is chosen on the basis of the values of both light sensors and of the water sensor. The default behaviour is to track the light. But if the robot is not in water, it starts to crawl. If the light sensors' signal is too weak, it turns on the spot until it finds the light source again. And if the signals are saturated (i.e., the robot is too close to the lamp), the robot stops. When a contact with an obstacle is detected with the front touch sensor, the backwards behaviour overrides all other behaviours for a few seconds.

Once a behaviour has been chosen, a second finite state machine determines the 7 control parameters (frequency, amplitude, and offset of each motor) to obtain that behaviour. For example, if light tracking is chosen, the speed of the robot is controlled inversely proportionally to the amplitude of light by adjusting both the frequency (eq. 5.6) and the amplitude of the oscillations (eq. 5.7). The caudal offset is controlled proportionally to the difference of light (eq. 5.8).

$$\omega_i = k_{\omega i} \cdot \frac{1}{l_1 + l_2} \quad i = 1, 2, 3 \tag{5.6}$$

$$R_i = k_{Ri} \cdot \frac{1}{l_1 + l_2} \quad i = 1, 2, 3 \tag{5.7}$$

$$X_3 = k_{X3} \cdot (l_1 - l_2) \quad X_1 = 0, X_2 = 0 \tag{5.8}$$

where the k_{ij} are gains of the regulator and l_1, l_2 the amplitudes of the two light sensors. Note that the CPG never needs any resetting and is continuously running while the control parameters are modified.

5.3 Experiments and results

5.3.1 Sequentially testing the locomotor behaviours

We tested the ability of the CPG to produce the different types of locomotor behaviours presented above. Figure 5.6 presents a sequence of transitions from one behaviour to the other. In that sequence, the CPG makes transitions between swimming straight with both pectoral and caudal fins ($t \leq 2$ s), turning with a caudal offset ($2 < t \leq 4$ s), swimming straight with only pectoral fins ($4 < t \leq 6$ s), swimming backwards ($6 < t \leq 8$ s), swimming upwards ($8 < t \leq 10$ s), rolling ($10 < t \leq 12$ s), slow swimming straight with pectoral and caudal fins ($12 < t \leq 14$ s), crawling ($14 < t \leq 18$ s), and swimming straight with small amplitudes ($18 < t \leq 20$ s). Figure 5.7 illustrates forward swimming with pectoral fins. Figure 5.8 shows the crawling gait using $X_1 = X_2$, it crawls straight forward. If only one pectoral fin is actuated the robot crawls to the left or right. With $X_1 = X_2 + \pi$, it crawls forward zigzagging.

Figure 5.9 shows a turning maneuver by modulating the offset of the tail fin (turn to the right followed by a turn to the left). The minimal radius of turning for this type of turning (with caudal offset) is 0.12 m. Even sharper turns can be made with the *turning on the spot* maneuver. Movies of the robot can be viewed at http://birg2.epfl.ch/boxybot.

All these transitions are obtained with abrupt changes of the control parameters ω, R_i, and X_i. Despite these abrupt changes, smooth oscillations are produced by the CPG (as shown on Figure 5.9). Note also that all oscillations remain phase-locked with a zero phase difference thanks to the inter-oscillator couplings.

5.3.2 Evaluating the speed of locomotion

The speed of locomotion can be adjusted by gradually increasing both the frequency and/or amplitude parameters of the CPG. Figure 5.10 shows the activity of the CPG when both are increased simultaneously.

In order to test how the speed of locomotion depends on the frequency and amplitude of oscillations, we carried out a series of swimming tests. Steady-state speed was measured at different levels of frequencies and amplitudes of all fins. Figure 5.11 shows the results for variations of frequency at a fixed amplitude (on the left) and for variations of amplitude at a fixed frequency (on the right). As could be expected, the speed of swimming increases with the frequency until the motors reach their torque limits. Similarly, at a fixed frequency, the speed of swimming increases with the amplitude until the oscillations become too large and create braking wakes. Overall, the robot can swim up to 0.37 m/s (i.e., 1.4 body lengths per second) at a frequency of 8 Hz and amplitudes of $\pm 40^\circ$ with both pectorals and caudal fins.

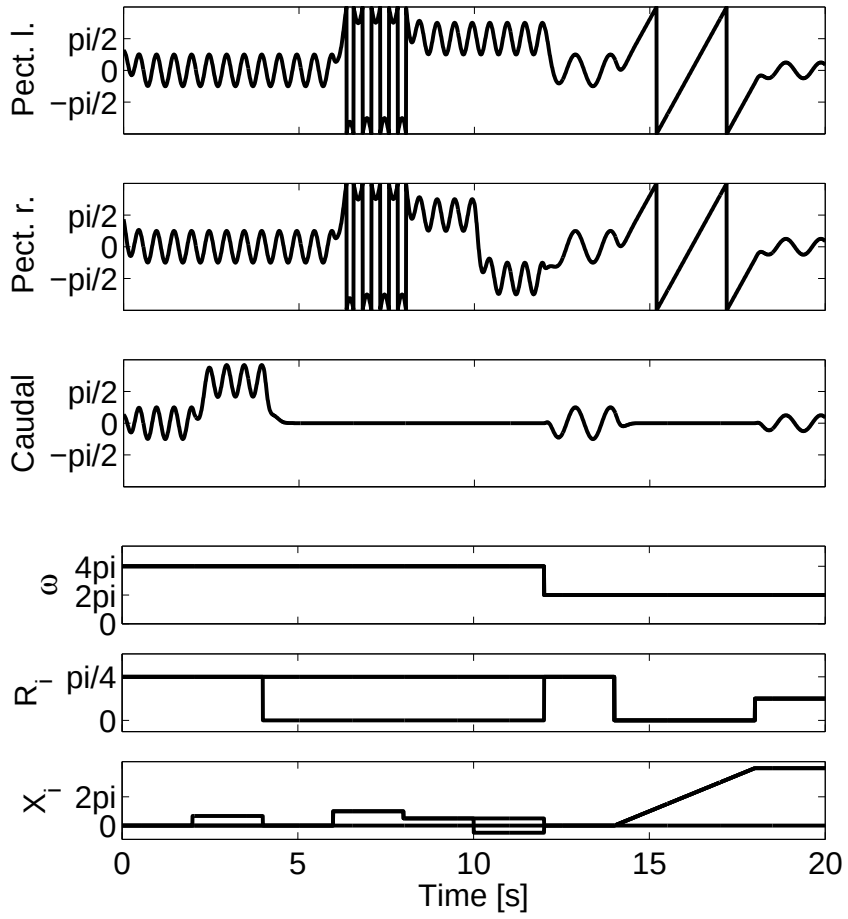


Figure 5.6: Sequence of different locomotor behaviours. The graphs show the set-points in radians sent to the three fins. See text for details.

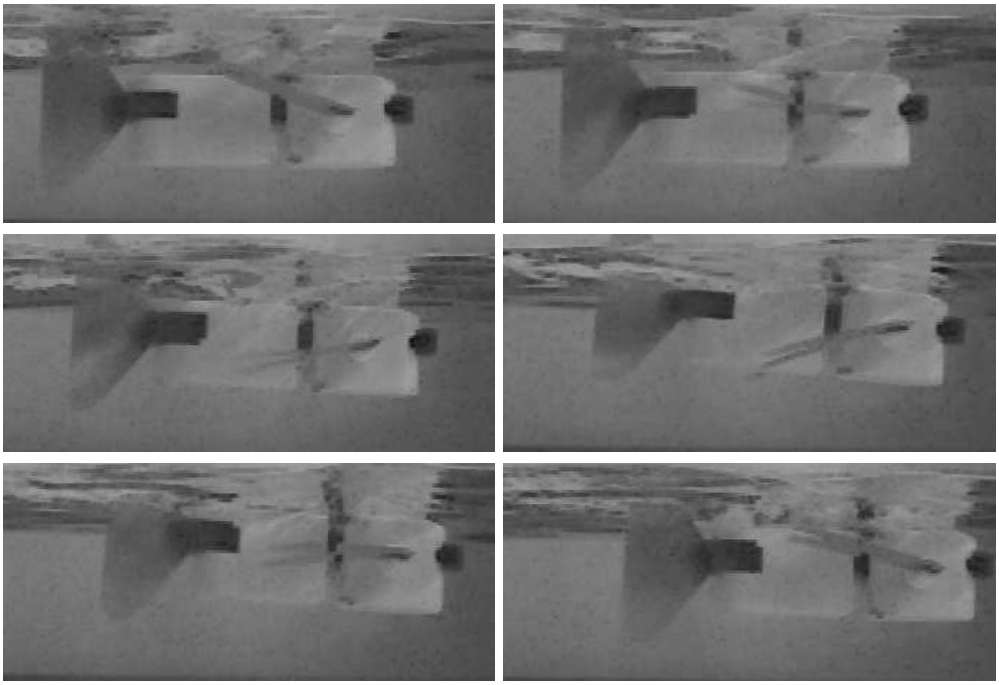


Figure 5.7: Snapshots of swimming forwards with both pectoral fins (from top left to bottom right).

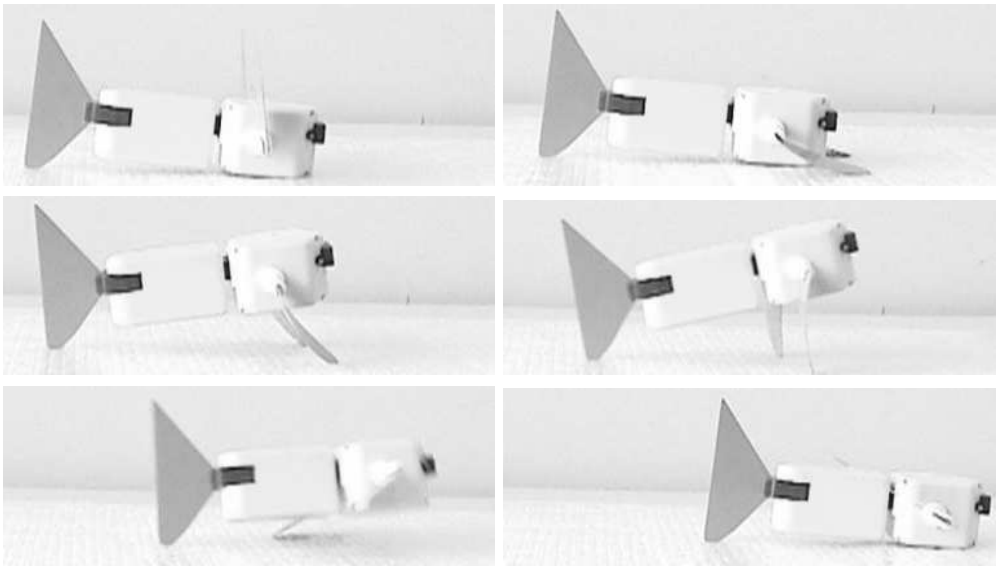


Figure 5.8: Snapshots of crawling using continuous rotation of pectoral fins $X_1 = X_2$ (from top left to bottom right).

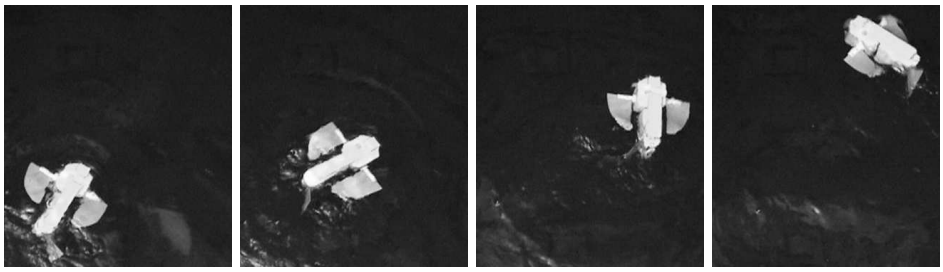


Figure 5.9: Snapshots of turning transition.

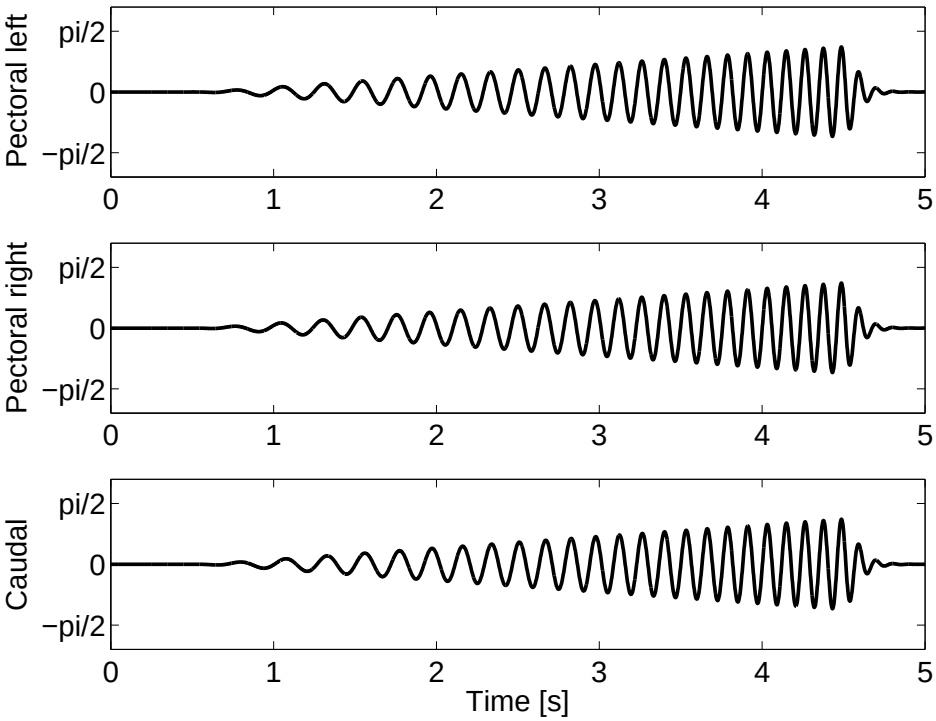


Figure 5.10: Acceleration during swimming.

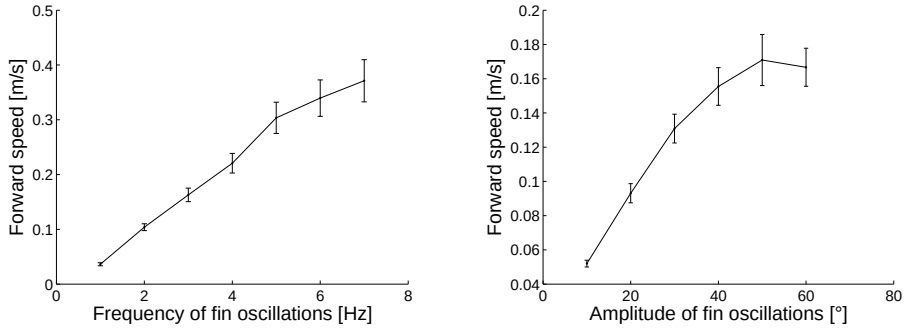


Figure 5.11: Variation of forward velocity with pectoral fins. On the left, variation with oscillations frequency at a fixed amplitude of 20°. On the right, variation with oscillation amplitude at a fixed frequency of 2 Hz. Speed is obtained from the measure of distance covered and time using video recordings. Error bars are calculated from the estimated precision of those two measures (± 0.02 m for the distance and ± 0.08 s for the time).

5.3.3 Phototaxis

Using the phototaxis behaviour described above, the fish robot is able to reach a static bright light (brighter than the environment) from a maximal distance of 50 cm and to keep station near the light. It is also able to follow a light that moves slowly (figure 5.12). If the light moves too quickly on the side, the robot cannot track it because the control law for choosing the speed and caudal offset is very basic (only proportional gains are used). The robot is programmed to then slowly turn on itself until the light comes into view again, in which case it resumes the light tracking behaviour. The same phototaxis behaviour is also implemented on ground (Figure 5.13).

5.4 Exhibition

Since March 2006 BoxyBot is part of a public exhibition ("forum découvertes") at the School of Computer and Communication Science at EPFL. The aim of this exhibition is to present some research projects carried out in the school to the general public. The robot stays day and night in an aquarium and different means of interacting with it are provided to the visitors. The robot is programmed with essentially the same control architecture as used in the previous experiments (Figure 5.5) with the exception that the robot is permanently connected through a tether to an offboard PC for monitoring and for receiving information from external sensors. The batteries are also permanently recharged through that tether to allow the robot to be active 16 hours per day. Because of this the robot is not truly autonomous anymore. For us the purpose of this exhibition is to demonstrate that the CPG-based control architecture is well suited for interactive

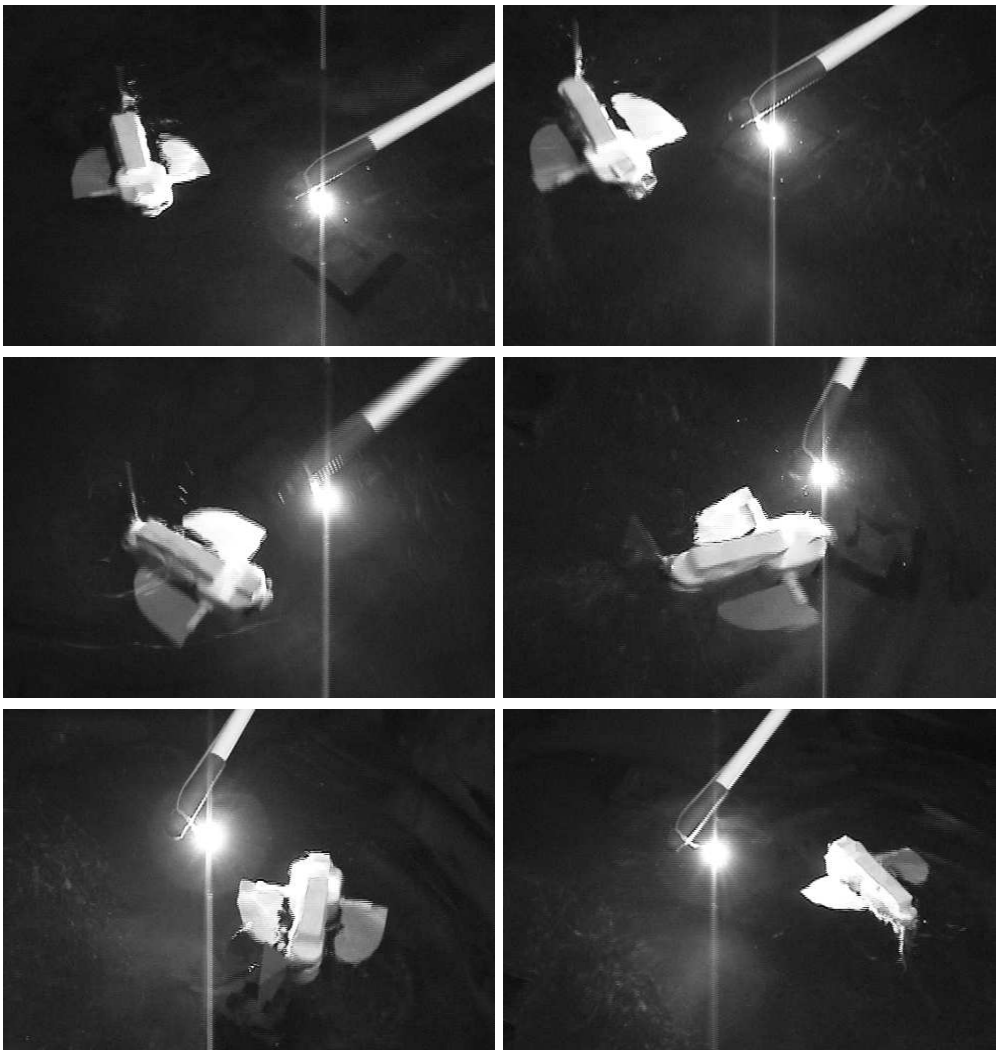


Figure 5.12: Snapshots of phototaxis during swimming (from top left to bottom right).

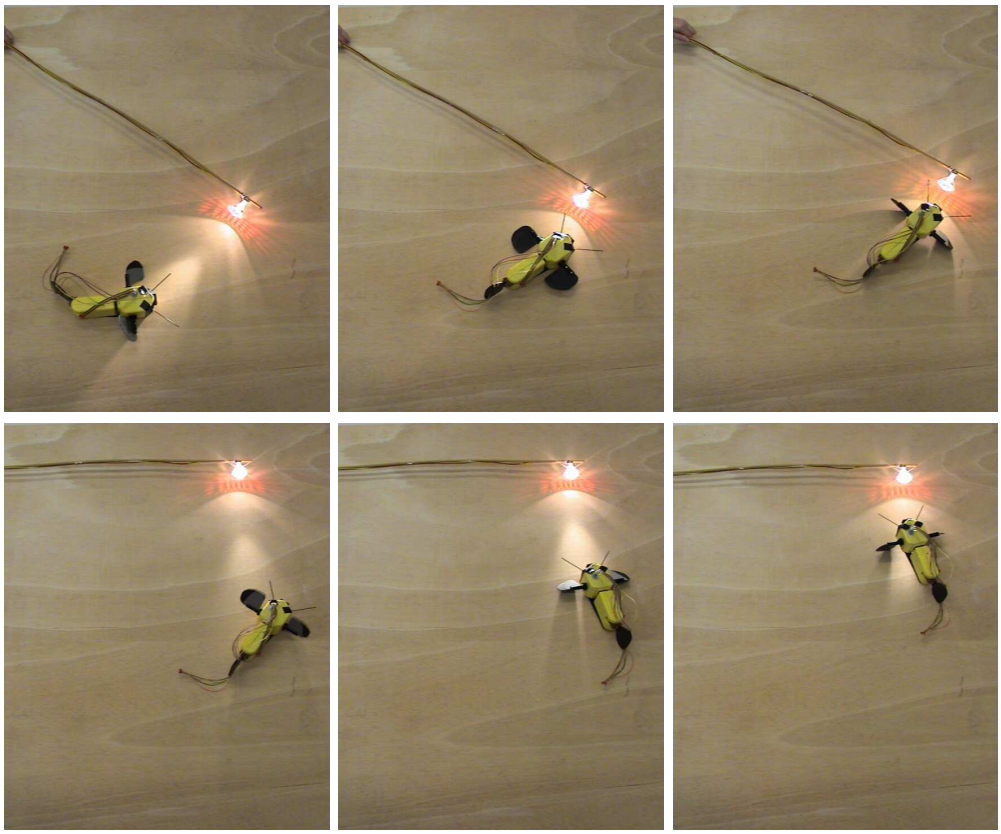


Figure 5.13: Snapshots of phototaxis during crawling (from top left to bottom right).

Figure 5.14: BoxyBot in its aquarium at the exhibition (picture: Alain Herzog).

control with a human in the loop. The exhibition is also a demonstration that the robot is robust enough for long and extensive use.

The environment is an aquarium (150×75 cm) filled with water, inside which the fish robot swims. Four halogen lamps are placed externally to the aquarium, near the corners of the short side. The whole setup is protected by a plexiglas cover, which restricts the visitors from directly manipulating it.

5.4.1 Hardware description

The overall structure of the system designed for the exhibition is depicted in figure 5.17. A standard aquarium filter is placed inside the aquarium to constantly clean the water; moreover, a small amount of sodium hypochlorite is added to avoid the development of algae. A fan is placed on the top of the plexiglas cover to remove the moisture, thus avoiding the formation of condensation.

The robot is connected to an interface card through a 5-wire cable and a rotating contact. The cable supplies the robot with external power when needed (24 V), has a signal to completely turn off the robot (i.e., to disconnect the batteries from the circuits) and also contains the I^2C signals used for communication. A small aquarium pump, like the one used during systematical tests for the snake robot (see section4.3.3), injects low pressure air inside the robot through a highly flexible silicon tube, to avoid water

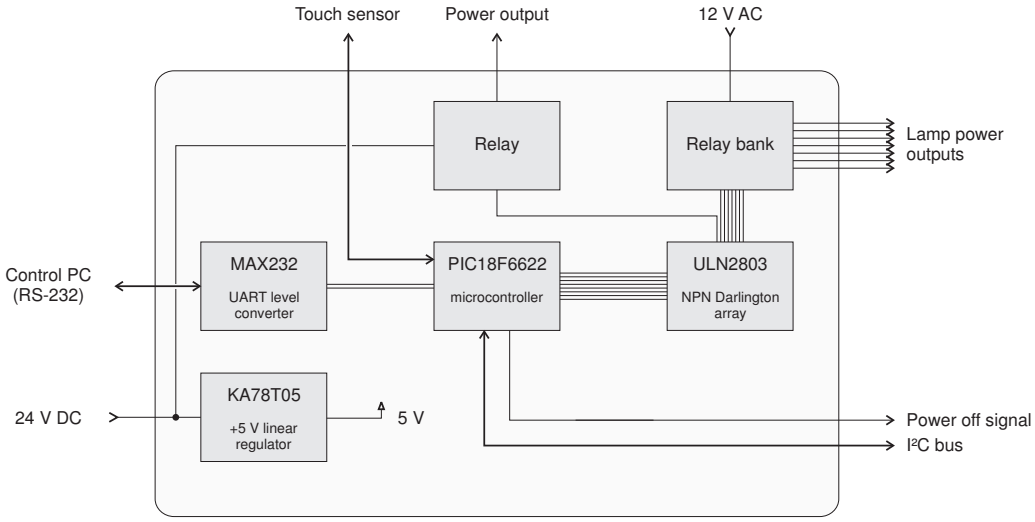


Figure 5.15: Block schematic drawing of the interface card used at the exhibition.

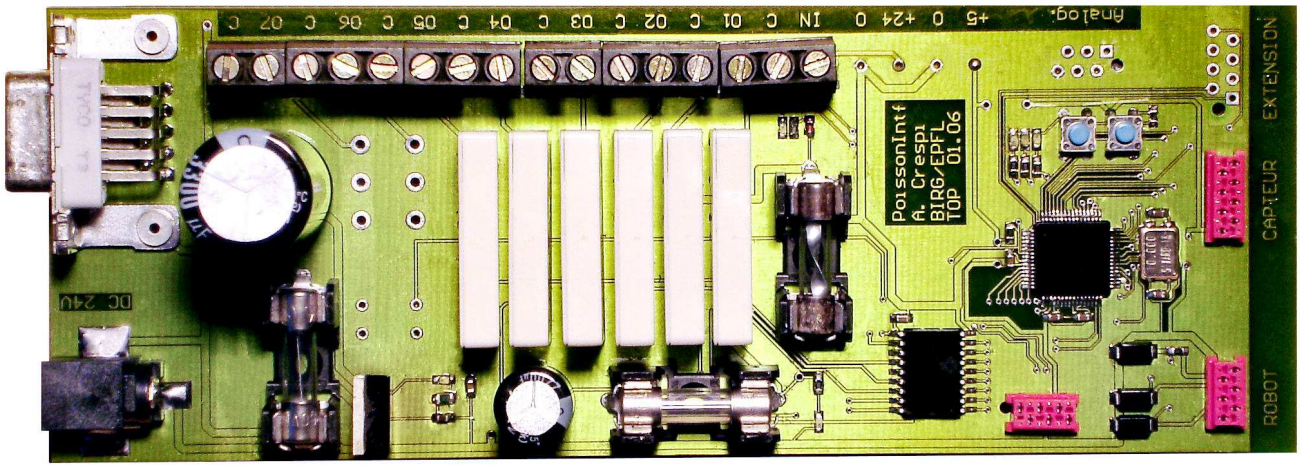


Figure 5.16: Interface card printed circuit (real size, top layer; only a few components are mounted on the bottom layer).

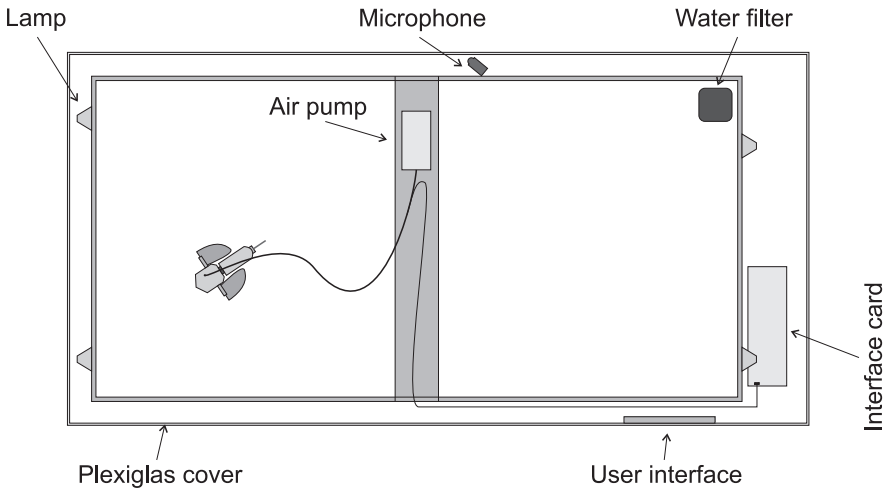


Figure 5.17: Schematic drawing of the whole setup at *forum découvertes* (top view).

leakages.[4]

The interface card (is based on a PIC18F6622 microcontroller, configured as an I^2C slave and connected to the internal bus of the robot through an I^2C driver (the internal drivers of the PIC are too weak, due to the cable length). The software on the microcontroller implements a register bank that can be read and written both over I^2C and using a RS-232 line connected to a PC. The interface card is powered with a 24 V switching power supply, whose output is also supplied to the robot through a relay. The card has seven other relays (four of which are used to power the halogen lamps, the others can be used for future extensions), which are connected to a 12 V transformer. The state of all the relays is directly controlled by one of the registers implemented on the microcontroller, and can thus be read and modified both by the robot and by the control PC.

Three buttons, implemented as capacitive touch sensors connected to the interface card, are fixed inside the plexiglas cover to implement a simple user interface (see below).

A small microphone connected to the PC is placed at the side of the aquarium to detect when users knock on the cover. The detection is done with a very simple but effective threshold function on the average intensity of the sound.

[4]The robot is normally waterproof, but for long term usage (e.g., over several months) the increased air pressure inside the robot helps preventing leakages which were inevitably occurring without it. The internal pressure also helps identifying possible leakage points, since these become visible through air bubbles.

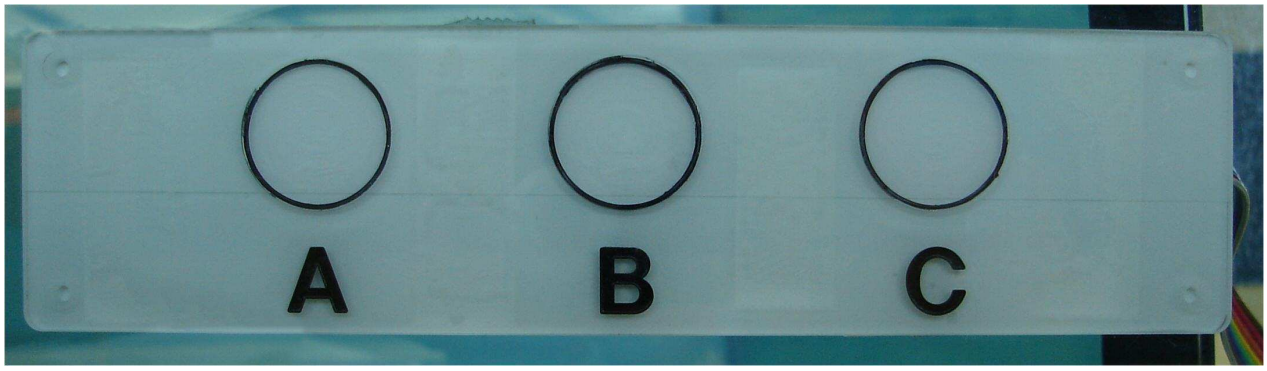


Figure 5.18: The user interface (real size) fixed inside the plexiglas cover. The capacitive touch sensor is placed behind the front panel.

5.4.2 User interaction

Visitors can interact with the robot in several ways: (1) by turning on the lights located around the aquarium, (2) by forcing the robot to perform a particular action by pressing a button, and (3) by knocking on the plexiglas cover. The three buttons of the user interface (see figure 5.18) have different functions. One of the them cyclically turns on one of the lights. By pressing it, the user can therefore induce the phototaxis behaviour. When one of the lights is turned on, the robot swims in its direction using the light sensors, and turns it off (using the communication with the interface card) when touching the border of the aquarium in front of the light. The two other buttons allow the user to force the robot to perform two particular behaviours: diving to the bottom of the aquarium and making it swim backwards. Knocks on the cover are detected and are also used for interacting with the system: a single hit makes the robot swim forwards faster than normal, and a double hit makes it do the spinning behaviour. When no user input is detected, the robot is programmed to randomly switch between the different locomotor behaviours described in Section 4.2. When no user activity is observed for more than two minutes, the robot is automatically turned off, and is reactivated when any type of activity is detected.

The amount of time the robot is powered each day is counted by the monitoring PC and plotted in figure 5.19. During peak days, the robot has been active up to almost 6 hours.

5.5 Discussion

BoxyBot has demonstrated its capacity of maneuverability. Using only three fins, it can move in 3D with different types of maneuvers and go out of water using a crawling gait. It can avoid obstacles by going backwards for a few seconds. Finally, the robot can reach a bright light and follow it slowly.

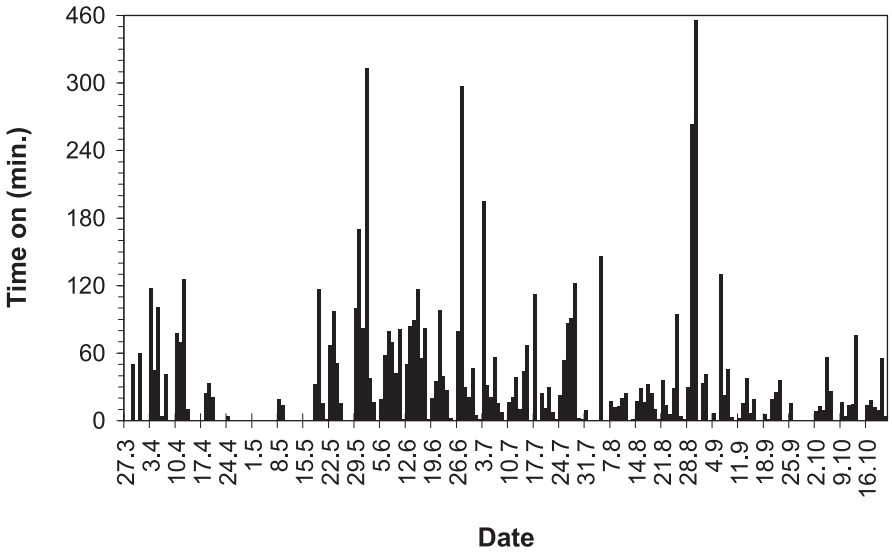


Figure 5.19: Daily robot activity plot.

One of the purposes of the experiments presented in this chapter was to demonstrate that the CPG model can be very useful for the online generation of the fin trajectories. Like for the snake robot, it provides the possibility to abruptly change control parameters while ensuring smooth variations of behaviour. Producing continuous and smoothly varying setpoints is indeed important to limit mechanical damage to the motors and gearboxes, but also to avoid jerks that could destabilize the swimming and crawling gaits. In addition, the phototaxis experiment showed that the CPG model can be continuously modulated and can therefore readily be used by higher level behaviour controllers. This is not unlike locomotion control in vertebrate animals where CPGs in the spinal cord produce the rhythmic patterns necessary for locomotion, and higher control centers such as the motor cortex and the cerebellum generate signals for the modulation of speed and direction.

The presentation of BoxyBot at the *forum découvertes* exhibition showed that the robot is able to swim for long periods of time. The waterproofing problems which were present during the first weeks were solved with the addition of the external air pump, as correcting them mechanically would imply modifications on the molded parts. Moreover, the possibility for any unexperienced user to control the robot behaviour demonstrates the validity of the CPG approach for interactive robot locomotion with a human in the loop.

Chapter 6

Salamander robot

This chapter presents a salamander robot that has been constructed using the elements described in chapter 3. It has 6 body elements, 2 limb elements (i.e., 4 rotating legs), and one head element containing the locomotion controller. It has been designed for testing the CPG models explaining the transition between swimming and walking in the salamander. The CPG model presented in this chapter has been designed by Auke Ijspeert. My contribution consists in its implementation and testing on the trajectory generator, the electronic development of the robot elements, and all the experiments and measures.

6.1 Central pattern generator model

The swimming motion of salamanders is similar to the one of lampreys, using axial undulations which propagate as travelling waves from head to tail. The walking motion has a different pattern: the salamander moves the diagonally opposed limbs together, generating at the same time a S-shaped standing wave (which has nodes at the girdles) with the body.

If the swimming motion can be generated by the same CPG presented in chapter 4 for the snake robot, the walking motion requires additional oscillators and connections: the CPG has in fact to generate signals for the limbs, and a standing wave for the body.

As for the snake robot, the body CPG model is a double chain of oscillators with nearest neighbor coupling (figure 6.1). The chain is designed to generate a travelling wave, from the head to the tail of the robot. This wave is used to achieve anguilliform swimming in water. In addition to this body CPG, limb oscillators have been added to the model (one per limb); they are bidirectionally coupled together and unidirectionally coupled to all body oscillators (see figure). During swimming, these oscillators are stopped (they don't oscillate), and thus do not influence the behaviour of the body CPG, which continues to produce a travelling wave. During walking, the oscillators are enabled and

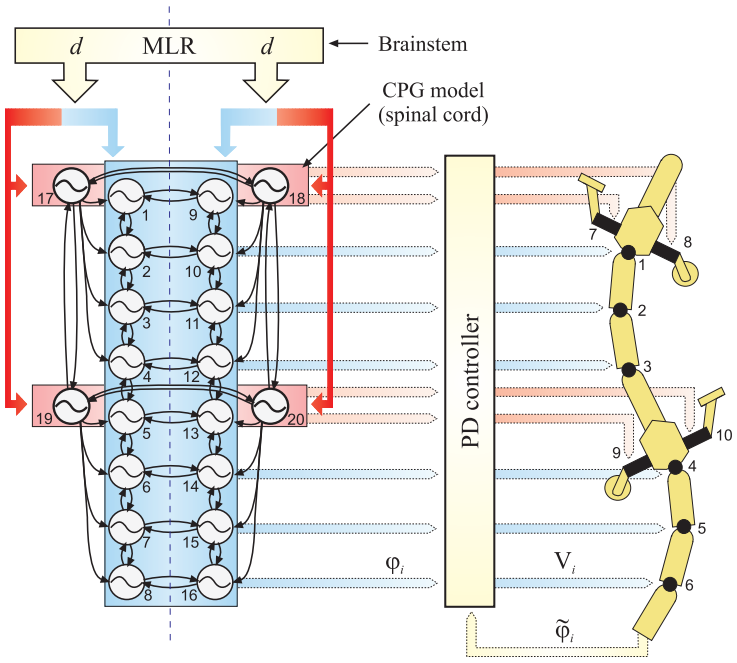


Figure 6.1: Structure of the CPG used in the robot.

influence the body oscillators, which begin to produce an S-shaped standing wave that can be used for walking.

The total number of oscillators is $N = 20$: $N_B = 16$ oscillators (i.e., 8 couples) for the body CPG (which controls 6 real elements and 2 fictive joints placed in the limb elements), and $N_L = 4$ oscillators for the limbs. Body joints (both real and fictive) are numbered 1 to 8 from head to tail. Oscillators in the left chain of the CPG are numbered 1 to 8 and those on the right side are numbered 9 to 16 from head to tail. Limb oscillators are numbered 17 to 20.

To allow the control of the whole CPG with a single parameter, reproducing the output of the MLR in the animal, a *saturation function* has been introduced:

$$\begin{pmatrix} \nu_i \\ R_i \end{pmatrix} = g(d) = \begin{pmatrix} g_\nu(d) \\ g_R(d) \end{pmatrix} \tag{6.1}$$

This function is a stepwise linear function, defined by the following equations:

$$g_\nu(d) = \begin{cases} c_{\nu,1}d + c_{\nu,0} & \text{if } d_{\text{low}} \leq d \leq d_{\text{high}} \\ \nu_{\text{sat}} & \text{otherwise} \end{cases} \tag{6.2}$$

$$g_R(d) = \begin{cases} c_{R,1}d + c_{R,0} & \text{if } d_{\text{low}} \leq d \leq d_{\text{high}} \\ R_{\text{sat}} & \text{otherwise} \end{cases} \tag{6.3}$$

Body and limb oscillators use different saturations functions, as they have to saturate at different levels of drive (see figure 6.3).

The CPG, a system of 20 coupled oscillators, is implemented with the same oscillator equations used for the snake robot (see section 4.2):

$$\begin{cases} \dot{\theta}_i &= 2\pi\nu_i + \sum_j w_{ij} \sin\Big(\theta_j - \theta_i - \phi_{ij}\Big) \\ \ddot{r}_i &= a_i\Big(\frac{a_i}{4}(R_i - r_i) - \dot{r}_i\Big) \\ x_i &= r_i\Big(1 + \cos(\theta_i)\Big) \end{cases} \tag{6.4}$$

where the state variables θ_i and r_i represent, respectively, the phase and the amplitude of the i^{th} oscillator, the parameters ν_i and R_i determine the intrinsic frequency and amplitude, and a_i is a positive constant. The coupling between the oscillators is defined by the weights w_{ij} and the phase biases ϕ_{ij}. The variable x_i is the rhythmic and positive output signal extracted out of oscillator i. For more details about the oscillator and a proof of convergence, see section 4.2.

For the body, similarly to the snake robot, the setpoints φ_i, i.e., the desired angles for the 6 actuated joints, are obtained by taking the difference between the x_i signals from the left and right oscillators. A standard PD motor controller is then used to compute the voltage τ_i (i.e., torque) applied to the motor (using a PWM signal):

$$\begin{aligned} \varphi_i &= x_i - x_{N+i} \\ \tau_i &= K_p e_i + K_d \dot{e}_i \end{aligned} \tag{6.5}$$

where $e_i = \varphi_i - \tilde{\varphi}_i$ is the tracking error between the desired angles φ_i and the actual angles $\tilde{\varphi}_i$ measured by the motor incremental encoders, and K_p and K_d are the proportional and derivative gains.

Because the limbs need to make complete rotations, their setpoints should monotonically increase, instead of having rhythmic movements like the body joints. The setpoints φ_i are therefore directly calculated from the phases θ_i of the limb oscillators using a non-uniform rotation function (which accelerates the movement of the leg when it is not in contact with the ground):

$$\begin{aligned} \varphi_i &= h(\theta_i) \\ \tau_i &= K_p e_i + K_d \dot{e}_i \end{aligned} \tag{6.6}$$

The $h(\theta)$ function used on the robot is plotted in figure 6.2.

The frequency and amplitude parameters of all oscillators are determined, on the base of the input drive d, by the saturation function:

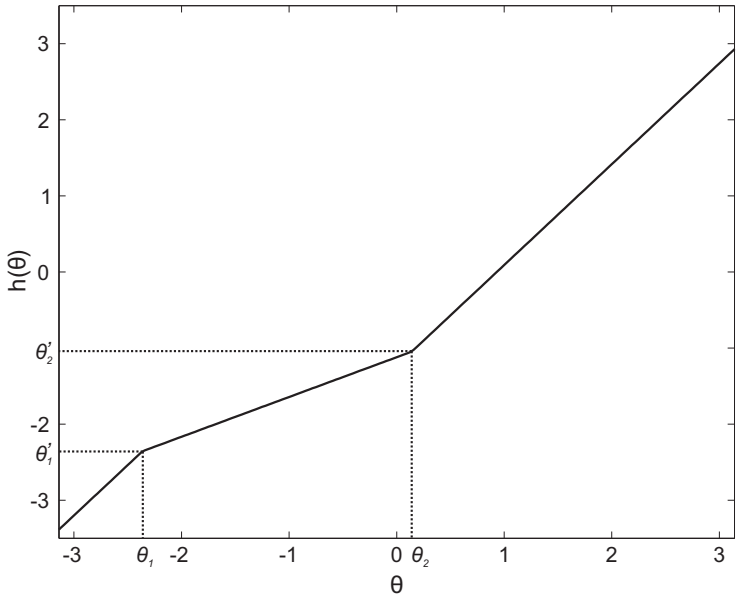


Figure 6.2: Leg rotation function used on the robot. θ_1 and θ_2 correspond, respectively, to the begin and end of the stance phase of the leg.

$$\begin{pmatrix} \nu_{\text{body}} \\ R_{\text{body}} \end{pmatrix} = g_{\text{body}}(d) \qquad \begin{pmatrix} \nu_{\text{limb}} \\ R_{\text{limb}} \end{pmatrix} = g_{\text{limb}}(d) \tag{6.7}$$

The actual saturation function is plotted in figure 6.3.

To achieve turning, different drives d_L and d_R can be applied to the left and right sides of the body oscillator.

In the body oscillator, the phase biases ϕ_{ij} are chosen to be equal to π between left and right oscillators (i.e., these will oscillate in anti-phase). The phase biases between neighbor oscillators are set to $\frac{2\pi}{N_B/2} = \frac{2\pi}{8}$ for the descending connections and to $-\frac{2\pi}{N_B/2} = -\frac{2\pi}{8}$ for the ascending connections; this produces a complete wave of the body. In the limb oscillator and for the couplings between body and limb oscillators, $\phi_{ij} = \pi$ is used for all connections.

We use $w_{ij} = 10$ for all connections, with the exception of the couplings from limb to body oscillators, which need to be stronger, for which a value of $w_{ij} = 30$ has been used. For all oscillators, $a_i = 20$. The PD coefficients K_p and K_d are tuned manually for each element (e.g., elements in middle of the chain require larger gains than those at the extremities for good trajectory tracking).

The d parameter can be modified online by a human operator from a control PC using the wireless connection. The CPG will rapidly adapt to any parameter change and

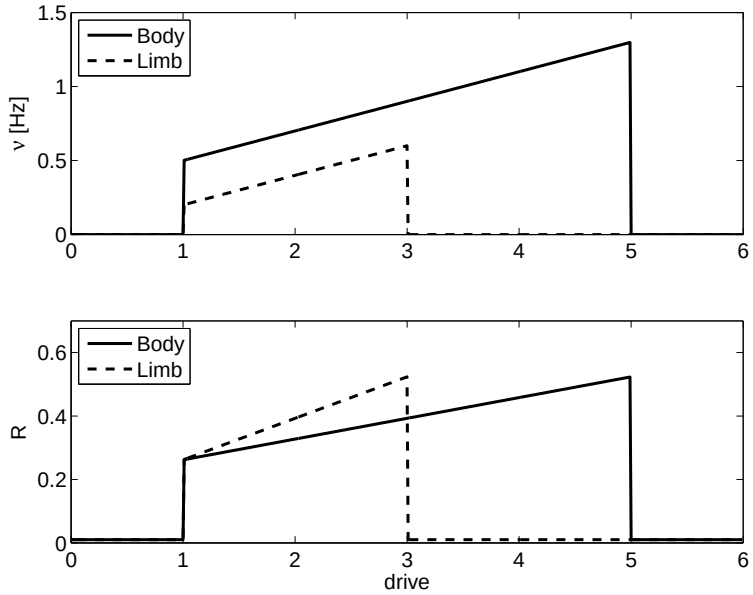


Figure 6.3: Saturation function used on the robot.

converge to the modified travelling or standing wave after a short transient period. An example of how the CPG reacts to parameter changes can be observed in figure 6.4: even with a continuously changing input drive, the oscillator generates smooth trajectories without any discontinuities in the outputs.

The differential equations are integrated by the microcontroller of the head using the Euler method, with a time step of 10 ms and using fixed point arithmetics. As the current trajectory generator has a limited computing power (10 MIPS with one 8-bit register), the code heavily uses lookup tables for calculating functions.

6.2 Locomotion characterization

6.2.1 Speed as function of drive

The speed of the robot has been measured for 18 different drive values (between 1.0 and 3.0 for walking, and between 3.001 and 5.0 for swimming, with a step of 0.25). Each measure has been repeated 5 times, giving a total of 90 measures.

For walking, the speed has been measured by taking the time used to travel a given distance (i.e., 2 m). For swimming, the procedure was the same, but the distance was reduced to 1 m and the measure only started after an acceleration space of approximately 50 cm, to approach steady-state swimming as close as possible. The results are plotted in figure 6.5.

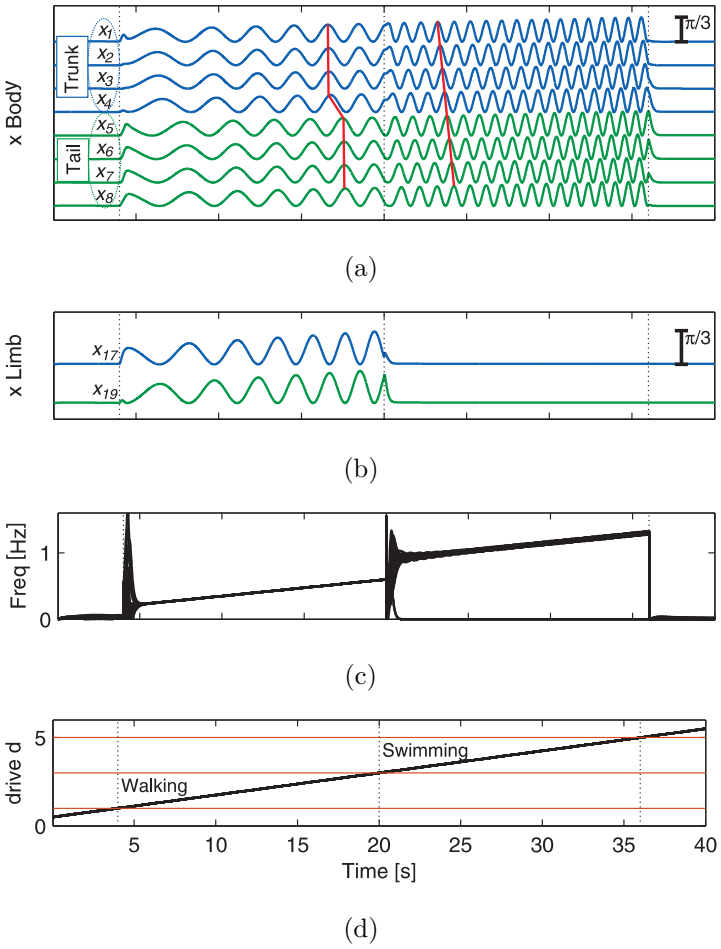


Figure 6.4: Switching from walking to swimming; activity of the CPG model when the drive signal is progressively increased. (a) x_i signals from the left body CPG oscillators (oscillators on the right side are exactly in antiphase). Units are in radians (scale bar on the top right). Note the transition from standing waves (with synchrony in the trunk, synchrony in the tail, and an antiphase relation between the two, 4 s $< t <$ 20 s) to travelling waves (20 s $< t <$ 36 s). (b) x_i signals from the left-limb CPG oscillators. Ipsilateral fore- and hindlimbs are in antiphase. (c) Instantaneous frequencies measured as $\frac{\dot{\theta}_i}{2\pi}$ in cycles/s. The variations in the instantaneous frequencies among individual oscillators at times $t = 4$ s and $t = 20$ s correspond to brief accelerations and decelerations before resynchronization. (d) Linear increase of the drive d applied to all oscillators. The horizontal lines correspond to the lower ($d^{\text{limb}}_{\text{low}} = d^{\text{body}}_{\text{low}} = 1$) and upper ($d^{\text{limb}}_{\text{high}} = d^{\text{body}}_{\text{high}} = 5$) oscillation thresholds for limb and body oscillators in arbitrary drive units.

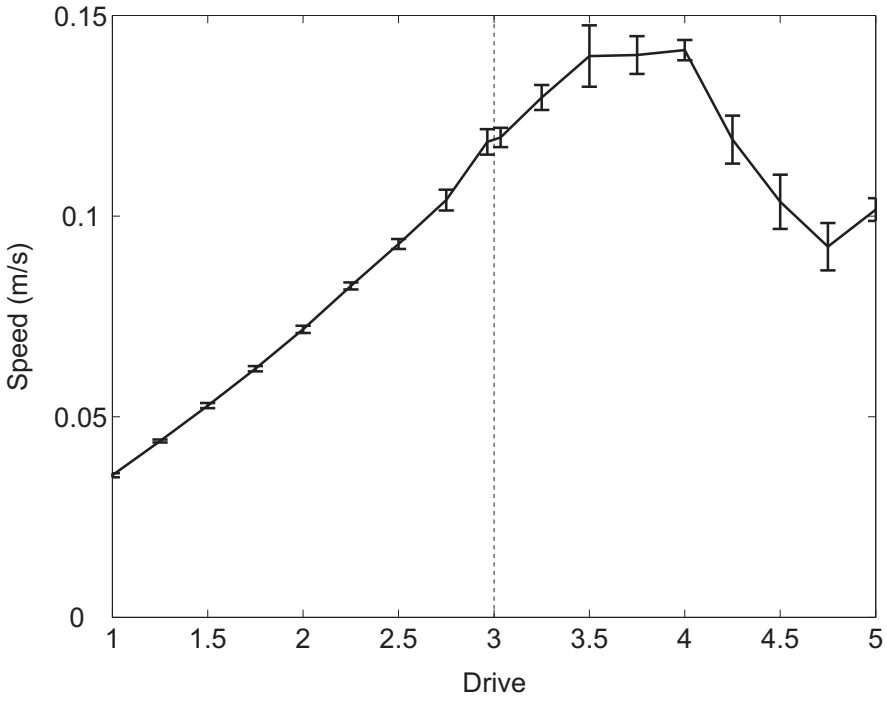


Figure 6.5: Speed of the salamander robot for walking ($d \leq 3$) and swimming ($d > 3$).

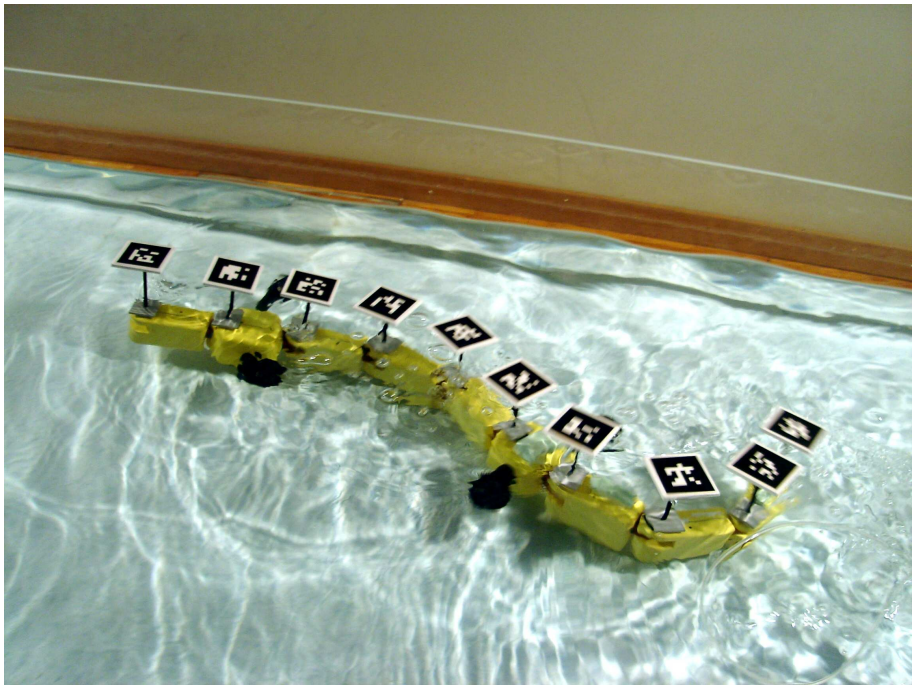


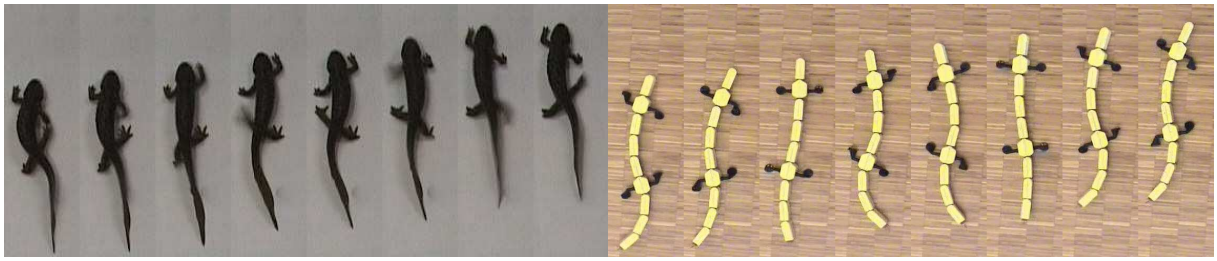
Figure 6.6: The salamander robot swimming with the tracking markers fixed on it.

For walking, the speed almost linearly increases over the whole range of drives, from $3.54 \cdot 10^{-2}$ m/s for $d = 1$ to $1.19 \cdot 10^{-1}$ m/s for $d = 3$. For swimming, the speed increases linearly from $1.20 \cdot 10^{-1}$ m/s for $d = 3.001$ to $1.40 \cdot 10^{-1}$ m/s for $d = 3.5$, then stabilizes around this value up to $d = 4$. For $d > 4$, the speed decreases, with a minimal value of $9.24 \cdot 10^{-2}$ m/s for $d = 4.5$. The decrease is mainly owing to the torque of the motors, which is not enough to follow the desired trajectories in water, with the consequence that the speed saturates, and then for higher drives, the travelling wave begins to deform, with the resulting decrease of speed.

6.2.2 Kinematic measurements

To compare the movements of the robot with those of the real animal, kinematic measurements have been done on the robot, using a custom video tracking system. These data have been compared with kinematic recordings of *Pleurodeles waltlii* salamanders, which have been provided by Isabelle Delvolvé (INSERM, Bordeaux).

The robot was filmed from above at 15 frames/s with a Basler A602fc-2 camera using a 8 mm C-mount lens. The frame data acquired over an IEEE1394 link was processed in real time with a custom program using the ARTag library (Fiala, 2004) to extract the (x, y) coordinates of the markers (sort of 2D barcodes, see fig. 6.6) placed on the robot. For body elements, the markers were placed at the rotation center of the output axis;



(a) (b)

Figure 6.7: Snapshots from videos for salamander (a) and robot (b) walking. The time step between the snapshots is 0.12 s for the salamander, and 0.20 s for the robot.

(a) (b)

Figure 6.8: Snapshots from videos for salamander (a) and robot (b) swimming. The time step between the snapshots is 0.04 s for the salamander, and 0.12 s for the robot.

for head and limb elements, they were placed at the same distance from the element's border than on body elements. The coordinates have been exported in CSV files and then imported in MATLAB for processing and analysis, like for the salamander. The tracking markers had a size of 55×55 mm. To minimize motion blur, the exposure time of the camera was set around 2 ms, using two 500 W halogen projectors for lighting. For walking, they were fixed on the top of the robot with double-sided adhesive tape. For swimming, they were fixed on a PVC support having the same size of the marker and placed 75 mm above the robot (using a rigid PVC cylinder of diameter 4 mm), to ensure that the markers were always out of the water during tracking (figure 6.6). The measures were repeated five times for each drive level. For walking, the camera field of view was always containing two complete cycles; for swimming, this varied between two and five cycles.

For illustration, snapshots from videos (without the tracking markers) for one locomotion cycle for walking and swimming can be seen in figures 6.7 and 6.8, respectively.

The envelopes of lateral displacement of each marker (relative to the direction of motion) measured with the video tracking are plotted in figures 6.9 (walking) and 6.10

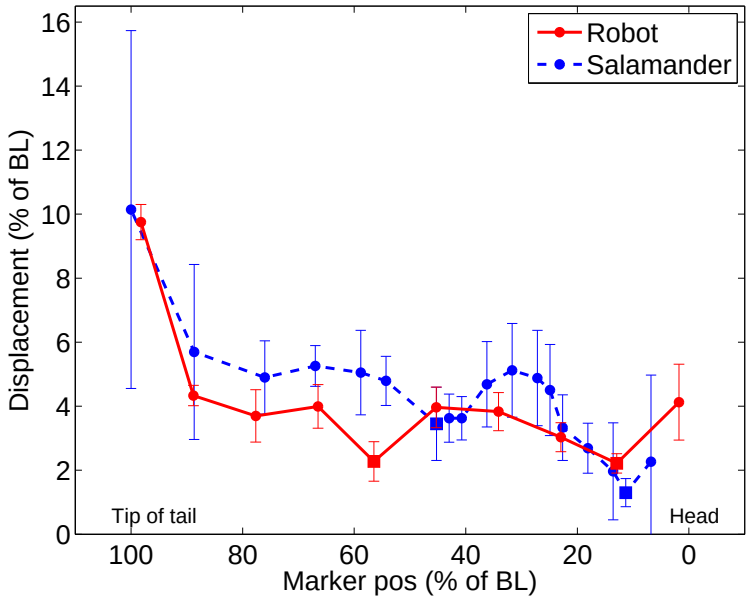


Figure 6.9: Lateral displacements of the robot and real salamander during walking.

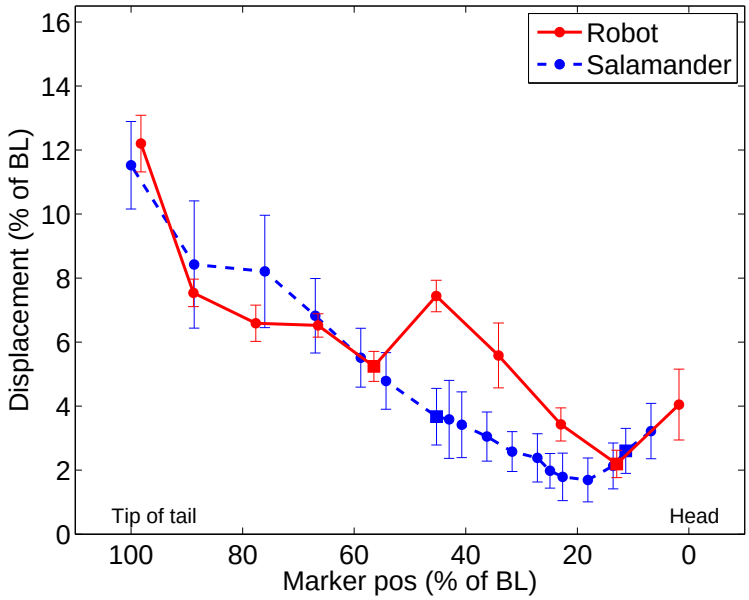
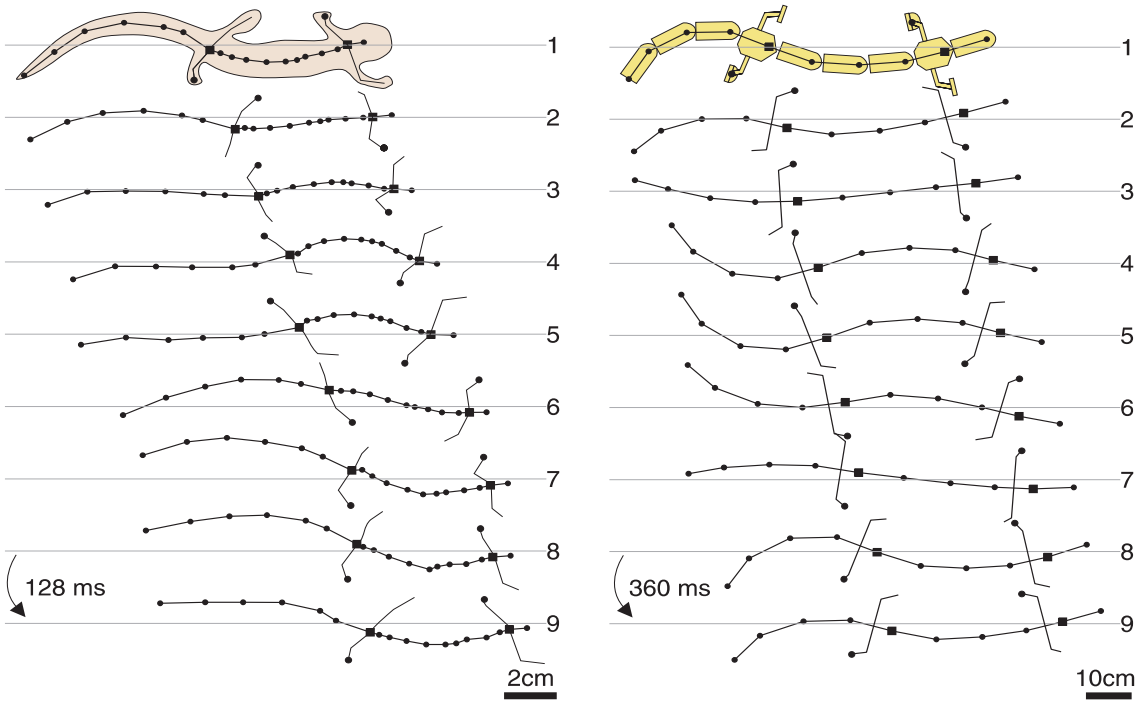
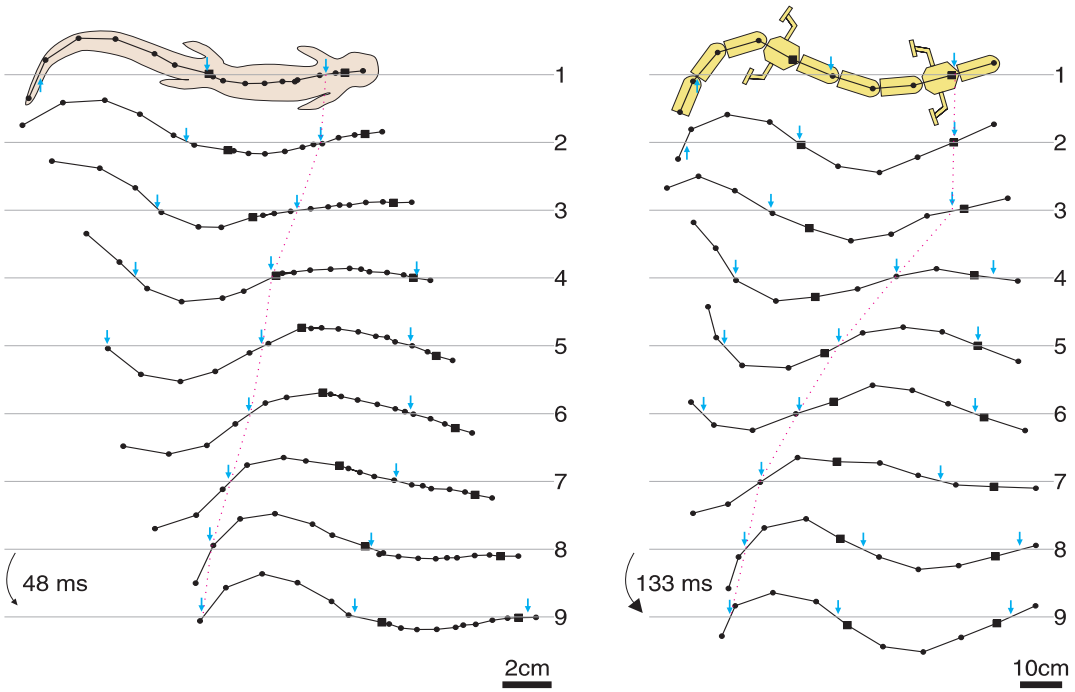


Figure 6.10: Lateral displacements of the robot and real salamander during swimming.



(a) (b)

Figure 6.11: Comparison of lateral displacements of the salamander (a) and robot (b) during walking. Velocities were 0.06 m/s (0.34 body lengths/s) for the animal, and 0.06 m/s (0.07 body lengths/s) for the robot ($d = 2.0$).



(a) (b)

Figure 6.12: Comparison of lateral displacements of the salamander (a) and robot (b) during swimming. Velocities were 0.17 m/s (0.89 body lengths/s) for the animal, and 0.11 m/s (0.13 body lengths/s) for the robot ($d = 4.0$).

(swimming), with the corresponding data of the animal. For walking, the motion is qualitatively similar for the robot and the animal; both have minimal lateral displacements at the girdles (which are not at the same relative position). The main differences are the displacements of the queue (the salamander maintains the tip of its queue mostly straight, whereas the robot moves it) and of the head (the robot lacks a joint in the neck, therefore producing a greater lateral displacement of the head). For swimming, the motion is also similar for the robot and the salamander, but with more differences than for walking. Particularly, the lateral displacements of the robot are higher than those of the salamander between the girdles. This can be explained by the lack of hinge joints in the limb elements of the robot, and by their increased weight.

A detailed comparison of the lateral displacements during a complete locomotion cycle can be found in figures 6.11 and 6.12. As it can be seen in the figures, the type of waves generated along the body is similar for the salamander and the robot, with the aforementioned differences.

6.3 Discussion

Considering the results presented in section 6.2, we can consider that the neurobiological model presented in section 6.1 has been successfully tested on a real robot and can indeed produce forward locomotion for both walking and swimming gaits. More details about the model can be found in Ijspeert et al. (2007).

In addition to the testing of neurobiological models, this robot (and its improvement) could find useful practical applications, for example for inspection/exploration tasks, where its agility and capability to deal with different environments could be a clear advantage. Compared to the snake robot, the presence of the legs can help the locomotion on solid grounds (possibly with small obstacles), removing at the same time the need for special contacts (e.g., passive wheels) to generate the asymmetric friction it requires to crawl with serpentine locomotion.

The presented CPG model will be useful for remote operation of the robot by a human: the operator only has to send high-level commands, like the upper parts of the brain in the animal; the CPG coordinates the trajectories of all the degrees of freedom for the control of speed, direction and type of gait.

Chapter 7

Online learning

In this chapter, a method to do online optimization of the locomotion parameters of the snake robot presented in chapter 4 is described. This optimization is based on the Powell's method, which is an heuristic optimization algorithm in multiple dimensions, which rapidly converges for smooth functions.

Systematic tests on different types of ground have been carried out (see chapter 4). The main outcomes of these experiments are that (1) optimal gaits are significantly different from one medium to the other, (2) the optimums are usually peaked, i.e., velocity rapidly becomes suboptimal when the parameters are moved away from the optimal values. This is clearly visible, for example, by comparing the results of the systematic crawling tests on linoleum presented later in this chapter, to the ones done on a wooden floor (parquet) that have been detailed in chapter 4. Thus, if the robot has to autonomously deal with different types of environment, it is clearly necessary to have an optimization function which can rapidly determine the appropriate locomotion parameters. Being able to learn gaits *online*, as opposed to do *offline* optimization with a model or a simulator for instance, is of great importance for biomimetic robotics. Indeed it might be one of the only solutions to tackle the problem of adapting gaits to complex, possibly unknown, environments. Keeping a realistic and up-to-date model of the interaction forces with such environments might be impossible or not accurate enough to allow alternative (e.g., model-based) approaches. The results presented in this chapter can be found in Crespi and Ijspeert (2007).

7.1 Video tracking

To run an optimization algorithm, we need an estimation of the performance of the robot (the velocity in the experiments presented here) for a given set of locomotion parameters. Several solutions to this problem exist. For simplicity, we chose video tracking (in future work, we are planning to provide the robot means to estimate its velocity on its own). The tracking system that has been developed for these experiments is relatively simple: a

bright 48 lm green led having an irradiation angle of 130° and powered by an independent Li-Ion battery is fixed on the head of robot. The experimental setup is filmed using a Basler a622f camera connected through a IEEE 1394 interface to a PC on which a simple tracking program is running. The whole system is depicted in figure 7.1.

The tracking program acquires the data from the camera at 15 fps, with a resolution of 800x600 pixels and a depth of 8 bits per pixel. The used image processing algorithm is trivial: the coordinates (S_x, S_y) of the LED spot (in pixels) are calculated as the average coordinates of all the pixels having a lightness higher than a given threshold (currently 192). The coordinates are then converted to the real (homogeneous) coordinates of the robot on the plane $(R_x/R_w, R_y/R_w)$ by using a 2D transformation matrix:

$$\begin{pmatrix} R_x \\ R_y \\ R_w \end{pmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \cdot \begin{pmatrix} S_x \\ S_y \\ 1 \end{pmatrix} \tag{7.1}$$

The coefficients $a, b, ..., i$ are obtained (for a given placement and orientation of the camera) by solving a linear system:

$$\begin{gathered} aP_{n,x} + bP_{n,y} + c - gD_{n,x}P_{n,x} - hD_{n,x}P_{n,y} - iD_{n,x} = 0 \\ dP_{n,x} + eP_{n,y} + f - gD_{n,y}P_{n,x} - hD_{n,y}P_{n,y} - iD_{n,y} = 0 \\ (\forall n \in [1...4]) \end{gathered} \tag{7.2}$$

where $D_1...D_4$ are the real coordinates of four reference points (aligned on two parallel lines), and $P_1...P_4$ their coordinates in pixels. The system is currently solved numerically by writing it into matrix form and using SVD decomposition.

The tracking system includes a TCP/IP server, allowing the coordinates of the robot (and its visibility status) to be remotely retrieved in real time.

7.2 Central pattern generator model

The CPG model used for the online learning experiments is essentially the same presented in chapter 4, with the exception that a new α_1 parameter has been introduced, which modulates the amplitudes R_i along the body to obtain an amplitude slope. The amplitudes are therefore $R_i = \alpha_i \cdot A_L$ for the left side ($i = [1, ..., N]$) and $R_i = \alpha_{i-N} \cdot A_R$ for the right side ($i = [N + 1, ..., 2N]$). The α_i parameters are linearly interpolated between the open parameter α_1 (head) and $\alpha_N = 1.0$ (tail). The parameter α_1 therefore acts as an amplitude gain, and allows the CPG to make undulations of increasing amplitude from head to tail, as is often seen during anguilliform swimming.

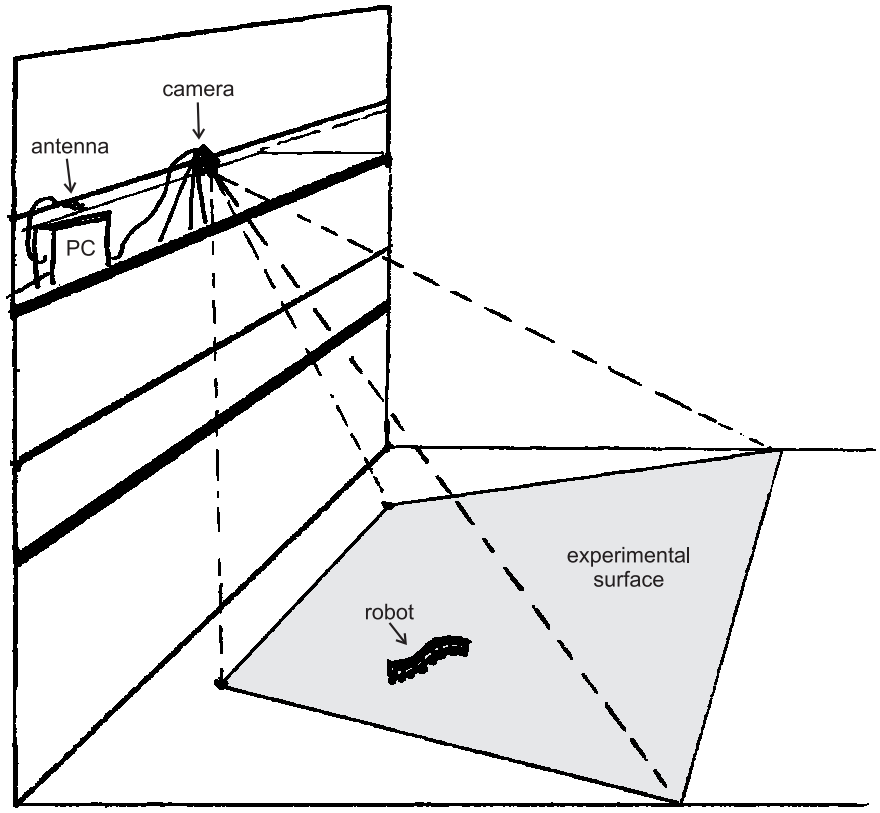


Figure 7.1: The video tracking system used for crawling. The setup for swimming is very similar.

7.3 Optimization algorithm

The function we want to optimize is the locomotion velocity $v(\vec{x})$ of the robot, where $\vec{x}$ is the parameter vector containing the parameters to be optimized (oscillation amplitude A, total phase lag $N\Delta\phi$ and amplitude gain α_1). The value of the function for a given set of parameters can be automatically estimated using the video tracking system (the parameters $\vec{x}$ can be sent to the robot using a TCP/IP gateway, see below).

As the convergence time is critical in this context (online optimization of locomotion parameters), methods requiring a large number of function evaluations (e.g., genetic algorithms) have to be avoided. Moreover, we do not have any gradient information for $v(\vec{x})$, and are therefore limited to gradient-free methods. The algorithm we chose is Powell's method (Press et al., 1994), which is an heuristic optimization algorithm that rapidly converges for smooth functions. The main risk associated with this kind of algorithm is the possibility to converge to a local optimum of the function, rather than to the global one; however, systematical tests with the snake robot show that the velocity function $v(\vec{x})$ is rather smooth with typically a single global optima for a given frequency. A brief description of the algorithm, inspired from the one found in Press et al. (1994), follows.

7.3.1 One dimensional optimization

The goal of function optimization is to find x such that $f(x)$ is the highest or lowest value in a finite neighborhood. From now on we just consider the problem of function minimization. Note that function maximization is trivially related because it is equivalent to a minimization $-f(x)$. The main idea of one-dimensional function optimization is to bracket the minimum with three points $a < b < c$ such that $f(b)$ is less than both $f(a)$ and $f(c)$. In this case and if f is nonsingular, f must have a minimum between a and c. Now suppose that a new point x is chosen between b and c. If $f(b) < f(x)$, the minimum is bracketed by the triplet (a, b, x). In the other case if $f(x) < f(b)$, the new bracketing points are (b, x, c). In both cases, the bracketing interval decreases and the function value of the middle point is the minimum found so far. Bracketing continues until the distance between the two outer points is tolerably small (Press et al., 1994). The challenge is finding the best strategy for choosing the new point x in the bracketing interval at each iteration. We use Brent's method, which is a combination of golden section search and parabolic interpolation (Brent, 1973; Press et al., 1994).

7.3.2 Multi-dimensional optimization

Consider a line defined by a starting point P and a direction $\vec{n}$ in N-dimensional space. It is possible to find the minimum of a multidimensional function f on this line using a one-dimensional optimization algorithm (Press et al., 1994) (e.g., Brent's method, see above). Direction-set methods for multidimensional function minimization consist of sequences of

such line minimizations. The methods differ by the strategies in choosing a new direction for the next line minimization at each stage. Powell's method (Brent, 1973; Press et al., 1994) is best explained with an example. Consider a function with a "valley" along $x = y$ that descends to the origin:

$$f(x, y) = x^2 + y^2 + (x - y)^2 \tag{7.3}$$

Powell's method starts with the unit vectors e_1, e_2, ..., e_N of the N-dimensional search space as a set of directions. One iteration of the algorithm does N line minimizations along the N directions in the set. The algorithm is illustrated in Figure 7.2 for the two-dimensional function introduced above (Eq. 7.3). Starting at the initial point $P_0 = (2, 5)$, the first line minimization along the direction given by the unit vector $[1, 0]^T$ takes us to the point P_1. From this point the second line minimization along $[0, 1]^T$ takes us to P_2 and completes the first iteration. As you can see on Figure 7.2, repeated line minimizations along the unit vectors would involve many iterations because the minimum would be approached in small steps. After each iteration, Powell's method checks if it is beneficial to replace one of the directions in the set by $v_i = P_0 - P_N$ where P_0 was the starting point at the current iteration and P_N the new point after the N line minimizations. In the example of Figure 7.2, v_2 replaces $[1, 0]^T$ in the second iteration. The algorithm correctly decides not to include new directions in all other iterations as this would actually slow down convergence. The mechanisms for deciding whether or not to include the new direction v_i after each iteration and which direction in the set should be replaced are described in Brent (1973); Press et al. (1994). Note that there is no learning rate; the algorithm simply always goes to the optimum in the next direction.

7.4 Results

Several optimization experiments, both with the real snake robot and in simulation, have been done, using two fixed frequencies, $\nu = 0.4$ Hz and $\nu = 1.0$ Hz. The frequency has not been included in the optimized parameters as the systematic tests showed a direct dependence of the speed on the frequency (Crespi and Ijspeert, 2006; Ijspeert and Crespi, 2007). The optimization has been done with the real robot for crawling on a horizontal plane and for swimming, and with the simulator for crawling on a horizontal plane and on a slope (ascending and descending). All the experiments with the real robot have been repeated five times. Note that all the presented systematic tests have been done with a fixed value of $\alpha_1 = 1.0$ (and thus in a 2D space, for a given frequency). The comparison of the optimization with the systematic tests permits a validation of the optimization algorithm.

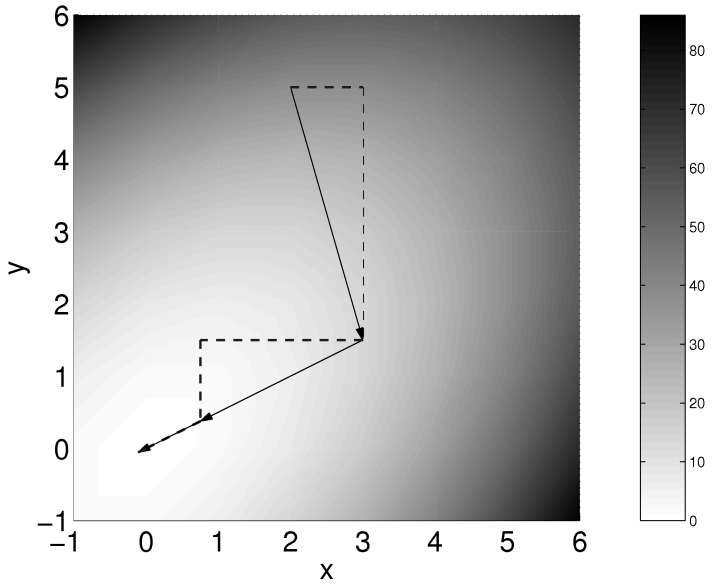


Figure 7.2: Example of function optimized with the Powell's method.

7.4.1 Optimization of crawling

The parameters A, $N\Delta\phi$ and α_1 have been optimized at fixed frequencies of $\nu = 0.4$ Hz and $\nu = 1.0$ Hz, on a horizontal linoleum experimental surface. The speed function was evaluated automatically, using the video tracking system, by running the robot for a fixed period of 10 s with the parameters to be evaluated and then measuring its distance from the initial position. Whenever the robot left the experimental surface (i.e., when it was not anymore visible by the tracking camera), the measure was automatically stopped and then restarted from the beginning after a manual repositioning of the robot. The optimization has been run five times, starting from a point at the center of the parameter space ($A = 30°$, $N\Delta\phi = 0.75$ and $\alpha_1 = 0.5$). For comparison, systematic tests have been done for the same frequencies, with a fixed value of $\alpha_1 = 1.0$, amplitudes between 10° and 60° (with a step of 10°) and a total phase lag between 0.25 and 1.50 (with a step of 0.25).

The results are plotted in figure 7.4. For $\nu = 0.4$ Hz, the algorithm converged to two slightly different optima: the average parameter values are $A = 52.3°$, $N\Delta\phi = 0.87$ and $\alpha_1 = 0.70$, with an average velocity of 0.178 m/s (0.231 BL/s) for the first optimum, and $A = 52.3°$, $N\Delta\phi = 1.05$ and $\alpha_1 = 0.50$, with an average velocity of 0.169 m/s (0.220 BL/s) for the second optimum. The maximal velocity obtained during systematic tests was 0.201 m/s (0.261 BL/s). For $\nu = 1.0$ Hz, the algorithm converged to a single optimal result having average parameter values of $A = 52.8°$, $N\Delta\phi = 1.24$ and $\alpha_1 = 0.81$, and an average obtained velocity of 0.296 m/s (0.384 BL/s). This is better than the maximal

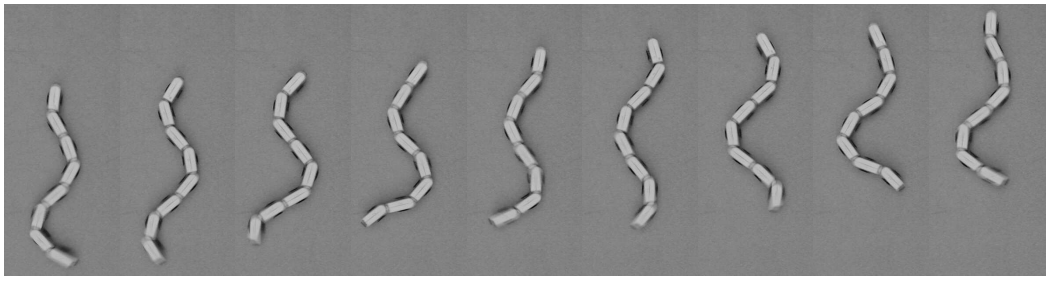


Figure 7.3: The robot crawling at $A = 53°$, $N \cdot \Delta\phi = 1.24$, $\nu = 1.0$ Hz and $\alpha_1 = 0.90$. The time step between the snapshots is 0.12 s. Videos of the robot are available at `http://birg.epfl.ch/amphibot`.

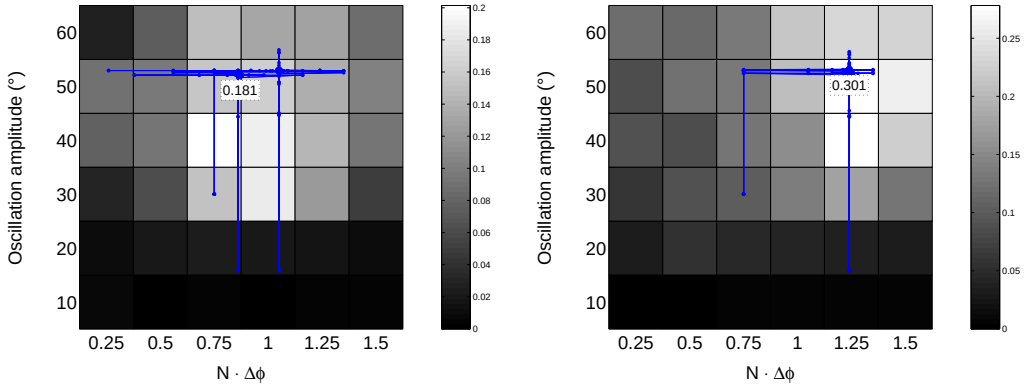
velocity obtained during systematic tests, which was 0.278 m/s (0.361 BL/s). There were always two iterations of the algorithm, with an average of 32.2 evaluations at $\nu = 0.4$ Hz and of 29.2 evaluations at $\nu = 1.0$ Hz. Figure 7.3 shows snapshots of the optimal crawling gait at $\nu = 1.0$ Hz.

The obtained gaits have a high amplitude for both frequencies. The other parameters depend on the frequency, which is in agreement with the results obtained during systematic tests; the wavelength is shorter (i.e., $N\Delta\phi$ is larger) for $\nu = 1.0$ Hz than for $\nu = 0.4$ Hz, and α_1 increases with the frequency.

7.4.2 Optimization of swimming

The parameters A, $N\Delta\phi$ and α_1 have been optimized at fixed frequencies of $\nu = 0.4$ Hz and $\nu = 1.0$ Hz, in an aquarium measuring 2.5 x 0.8 m. The speed function was evaluated automatically, using the video tracking system, by running the robot with the parameters to be evaluated and then measuring its distance from the point reached after an acceleration phase of 1.50 s. For each measure, the robot was placed at the beginning of the aquarium, and stopped when it reached the position threshold (15 cm before the end of the aquarium), or a maximum run time of 10 s (whichever came first). As for crawling, the optimization has been run five times, with the same starting point. Systematic tests have been done for the same frequencies (see chapter 4 and Ijspeert and Crespi (2007)), with the same parameter range than for crawling.

The results are plotted in figure 7.6. For $\nu = 0.4$ Hz, the algorithm converged to several distinct optima (having similar amplitudes but different values of $N\Delta\phi$ and α_1) with similar resulting velocities and an average of 0.132 m/s (0.171 BL/s). The parameter values for the best result are $A = 42.0°$, $N\Delta\phi = 0.45$ and $\alpha_1 = 0.50$, with a resulting velocity of 0.136 m/s (0.177 BL/s). The maximal velocity obtained during systematic tests was 0.147 m/s (0.191 BL/s). For $\nu = 1.0$ Hz, the algorithm found four optimal results, all having the same amplitude ($A = 52.9°$), but with different values of $N\Delta\phi$



(a) 0.4 Hz (b) 1.0 Hz

Figure 7.4: Results of the optimization for crawling. The data from the systematic tests with $\alpha_1 = 1.0$ are plotted in the background, and the evaluations done by five runs of the optimization algorithm are represented by the small dots (the α_1 dimension is not visible in the plot). All the speeds are in m/s. The speed indicated in the caption inside the plot is the highest velocity obtained with optimization.

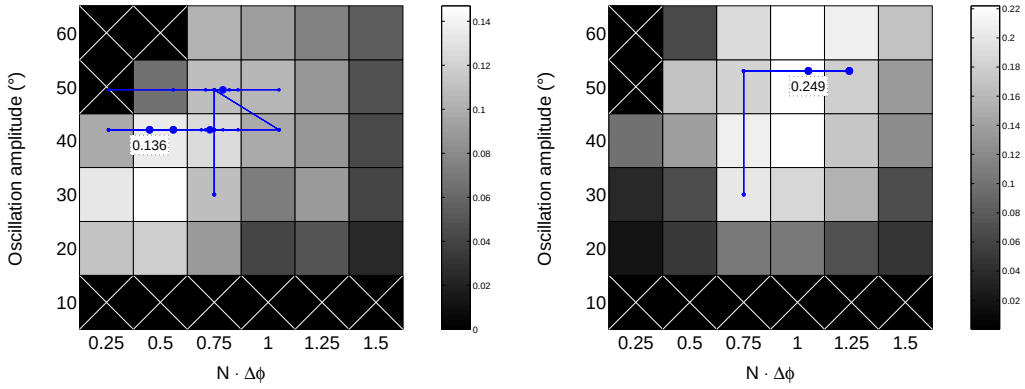
and α_1. The resulting average velocity is 0.220 m/s (0.286 BL/s). The best result has $N\Delta\phi = 1.05$ and $\alpha_1 = 0.82$, and the obtained velocity is 0.249 m/s (0.323 BL/s). The average velocity obtained with the optimization is therefore very similar to the one found during systematic tests, which was 0.222 m/s (0.288 BL/s). There was always only one iteration of the algorithm, with an average of 14.4 evaluations at $\nu = 0.4$ Hz and of 10.4 evaluations at $\nu = 1.0$ Hz. Figure 7.5 shows snapshots of the optimal swimming gait.

The parameters of the obtained optimal gaits clearly depend on the frequency, like it was the case during the systematic tests (see chapter 4). As for crawling, the wavelength is shorter when the frequency is higher; the amplitude increases with the frequency, and there is only a slight change of α_1. The fact that all optimal values of α_1 are smaller than 1.0 is in accordance with the anguilliform swimming with increasing amplitude observed in animals (Gillis, 1996).

7.4.3 Optimization of simulated crawling

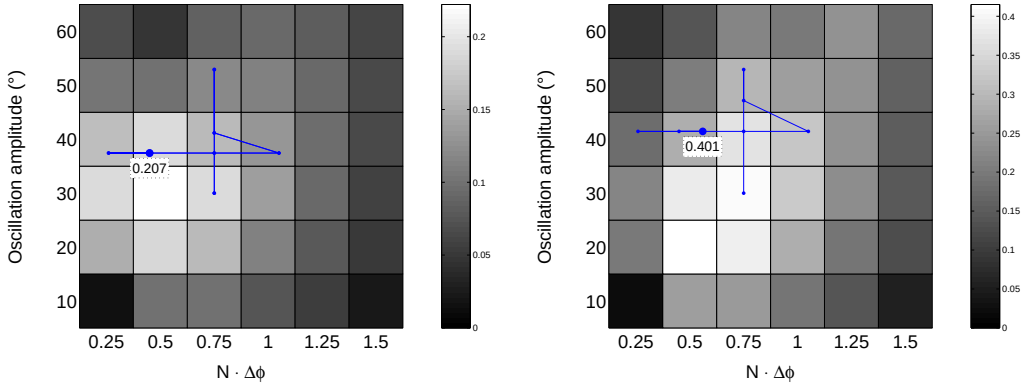
We reproduce here in simulation the optimization of crawling done with the real robot. The use of a simulation allows us to test the optimization in environments that are difficult to realize (see next subsection). The parameters A, $N\Delta\phi$ and α_1 have been optimized at fixed frequencies of $\nu = 0.4$ Hz and $\nu = 1.0$ Hz, in an environment with friction coefficients $\mu_\perp = 1.0$ and $\mu_\parallel = 0.05$. For each evaluation, the simulator was started with the robot in the initial position, and its average velocity measured over 20 s after a stabilization time of 10 s. The starting from was at the center of the parameter space ($A = 30°$, $N\Delta\phi = 0.75$

Figure 7.5: The robot swimming at $A = 53°$, $N \cdot \Delta\phi = 1.05$, $\nu = 1.0$ Hz and $\alpha_1 = 0.82$. The time step between the snapshots is 0.12 s. Videos of the robot are available at `http://birg.epfl.ch/amphibot`.



(a) 0.4 Hz (b) 1.0 Hz

Figure 7.6: Results of the optimization for swimming at $\nu = 0.4$ Hz and $\nu = 1.0$ Hz. The speed indicated in the caption is the highest velocity obtained during optimization.



(a) 0.4 Hz (b) 1.0 Hz

Figure 7.7: Results of the optimization for simulated crawling at $\nu = 0.4$ Hz and $\nu = 1.0$ Hz. The speed indicated in the caption is the highest velocity obtained during optimization.

and $\alpha_1 = 0.5$). Systematic tests have been done for the same frequencies, with a fixed value of $\alpha_1 = 1.0$, amplitudes between 10° and 60° (with a step of 10°) and a total phase lag between 0.25 and 1.50 (with a step of 0.25).

The results are plotted in figure 7.7. For $\nu = 0.4$ Hz, the algorithm converged to an optimum with $A = 37.4°$, $N\Delta\phi = 0.45$ and $\alpha_1 = 0.82$, with a velocity of 0.207 m/s (0.269 BL/s). The maximal velocity obtained during systematic tests was slightly higher, 0.222 m/s (0.288 BL/s). For $\nu = 1.0$ Hz, the algorithm found optimal parameter values of $A = 41.5°$, $N\Delta\phi = 0.56$ and $\alpha_1 = 0.45$, and a resulting velocity of 0.401 m/s (0.521 BL/s), slightly lower than the maximal one found with systematic tests, which was 0.415 m/s (0.539 BL/s). There was always only one iteration of the algorithm, with 9 evaluations at $\nu = 0.4$ Hz and 14 evaluations at $\nu = 1.0$ Hz.

The amplitude and wavelength of the optimum are similar for the two frequencies, and only the α_1 parameter decreases with the frequency. This is a clear difference compared to the results obtained with the real robot: the obtained maximal velocities are higher than the real ones, and the position of the optima in the systematical tests is clearly different. This mostly owes to the used friction model, which is too simplified.

7.4.4 Optimization of simulated crawling on a slope

The movement of a snake on a slope has different parameters than on a flat ground (Hirose, 1993), and it is clearly expected that this will be also the case for a snake robot.

The parameters A, $N\Delta\phi$ and α_1 have been optimized in the same way of the simulated crawling, using an environment in which the ground was rotated of a given angle θ. Systematic tests have also been done with the same parameter range.

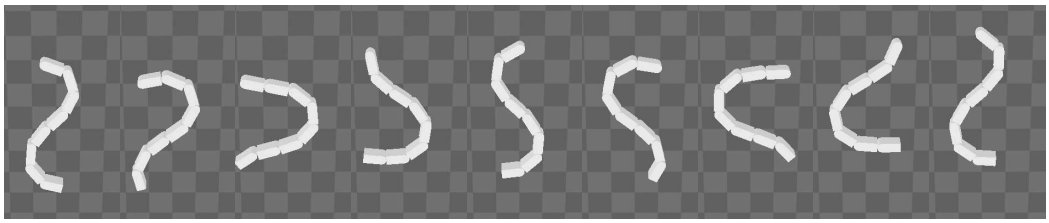
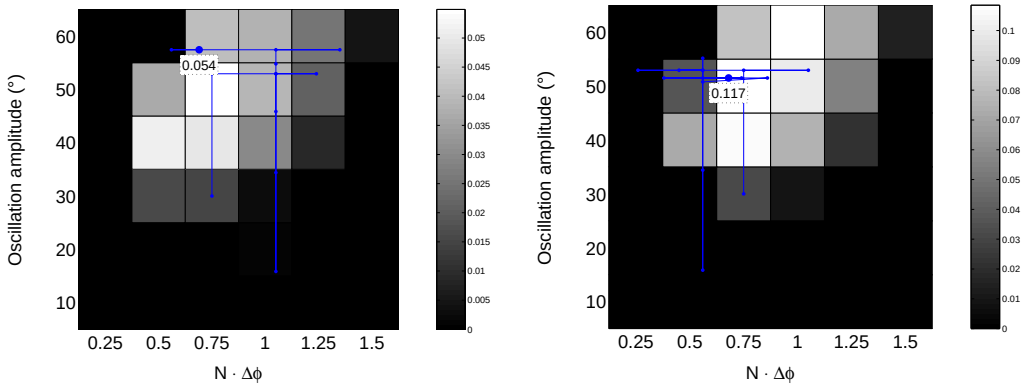


Figure 7.8: The simulated robot crawling on a slope ($\theta = 15°$) at $A = 51°$, $N \cdot \Delta\phi = 0.68$, $\nu = 1.0$ Hz and $\alpha_1 = 0.78$. The time step between the snapshots is 0.12 s.



(a) 0.4 Hz

(b) 1.0 Hz

Figure 7.9: Results of the optimization for simulated crawling on slope ($\theta = 15°$).

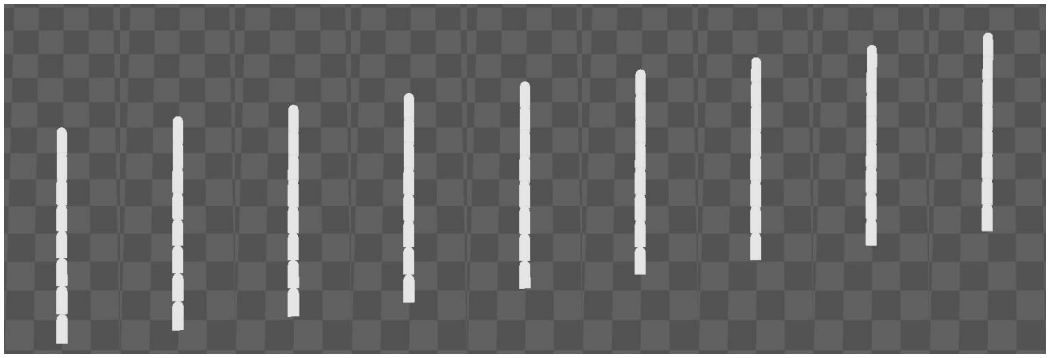
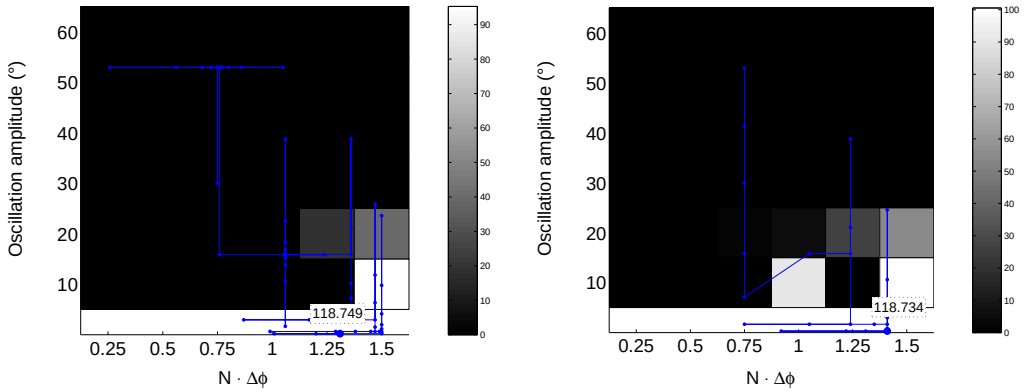


Figure 7.10: The simulated robot descending a slope ($\theta = -15°$) at $A = 0.3°$, $N \cdot \Delta\phi = 1.41$, $\nu = 1.0$ Hz and $\alpha_1 = 0.11$. The time step between the snapshots is 0.12 s.



(a) 0.4 Hz (b) 1.0 Hz

Figure 7.11: Results of the optimization for simulated crawling on a slope ($\theta = -15°$).

The results for $\theta = 15°$ (where a positive angle means that the robot climbs on the slope) are plotted in figure 7.9. For $\nu = 0.4$ Hz, the algorithm converged to an optimum with $A = 57.4°$, $N\Delta\phi = 0.69$ and $\alpha_1 = 0.70$, with a velocity of 0.054 m/s (0.070 BL/s), which is very similar to the maximal velocity obtained with systematic tests, 0.055 m/s (0.071 BL/s). For $\nu = 1.0$ Hz, the algorithm found optimal parameter values of $A = 51.5°$, $N\Delta\phi = 0.68$ and $\alpha_1 = 0.78$, and a resulting velocity of 0.117 m/s (0.152 BL/s), slightly higher than the maximal one found during systematic tests, 0.109 m/s (0.142 BL/s). There always were two iterations of the algorithm, with a total of 17 evaluations at $\nu = 0.4$ Hz and 22 evaluations at $\nu = 1.0$ Hz. Figure 7.8 shows snapshots of the resulting optimal gait.

The parameters of the found optima are similar for both frequencies (with a slightly higher amplitude at $\nu = 0.4$ Hz). The algorithm clearly found waves with higher amplitudes than for crawling on a plane, as it is the case for real snakes (Hirose, 1993). For instance, the higher oscillation amplitudes increase the number of elements perpendicular to the slope, which therefore reduces slipping.

The results for $\theta = -15°$ are plotted in figure 7.11. For $\nu = 0.4$ Hz, the algorithm converged to an optimum with $A = 0.2°$, $N\Delta\phi = 1.31$ and $\alpha_1 = 0.18$, with a velocity of 118.7 m/s (154.2 BL/s). The maximal velocity obtained with systematic tests is 95.3 m/s (123.8 BL/s). For $\nu = 1.0$ Hz, the algorithm found optimal parameter values of $A = 0.3°$, $N\Delta\phi = 1.41$ and $\alpha_1 = 0.11$, and a resulting velocity of 118.7 m/s (154.2 BL/s), whereas the maximal one found during systematic tests is 100.6 m/s (130.6 BL/s). There always were two iterations of the algorithm, with a total of 17 evaluations at $\nu = 0.4$ Hz and 22 evaluations at $\nu = 1.0$ Hz.

The obtained speeds are clearly physically unrealistic and are caused by the simplicity of the physical model that does not include velocity-dependent friction terms. The optimal gaits are very similar for both frequencies, and can be summarized as having the robot as

straight as possible and letting it freely roll down the slope. These parameters are very different from those found for climbing.

7.5 Discussion

This work has shown that the fastest gaits are considerably different from one medium to the other. For instance crawling up a slope requires undulations with large amplitudes, while crawling down a slope requires very small amplitudes. And slow swimming requires shorter phase lags than slow crawling. This dependence on the environment is in agreement with observations made by others (Hirose, 1993). In agreement with our previous studies (presented in chapter 4 and in Crespi and Ijspeert (2006); Ijspeert and Crespi (2007)), frequency is the parameter whose influence on the speed of locomotion is the simplest: with all other parameters fixed (i.e., amplitude, phase lag, and amplitude gain), increasing the frequency generally leads to an increase of speed (in the range tested). This makes the frequency a useful control parameter. But one should notice that the optimal gaits change with the frequency. It is therefore important to adapt all parameters when the frequency is changed. Another important observation is that the optima are peaked. For a given medium and a given frequency, the speed of locomotion drops rapidly when the parameters are changed compared to their optimal values. In other words, two seemingly very similar gaits might result in dramatically different speeds of locomotion.

All these observations confirm the importance to finely adapt gaits to the environment. There is not a single gait which performs satisfactorily in all conditions, and a robot that would rely on a single gait for various environments would be strongly suboptimal for most conditions. These empirical tests with our robot have therefore confirmed the necessity of designing online optimization methods for snake robots.

Any method for doing online optimization requires to fulfill at least two characteristics: (1) to allow parameters to be changed online —i.e., to have a control mechanism which smoothly adapts to parameter changes and does not need to be reset between evaluations— and (2) to be fast —in order to avoid excessive wear and tear, and prohibitive testing durations. The results presented here show that our control mechanism satisfactorily fulfills the two requirements. The CPG is a useful building block that is well-suited for optimizing the locomotion and modulating it (e.g., adapting the speed and the direction, see chapter 4 and Ijspeert and Crespi (2007)), and for optimizing it. The Powell's method proved to be a useful algorithm for rapidly finding the optimal parameters of the CPG in a given environment. It is significantly faster than doing extensive systematic evaluations of the robot velocity on the parameter space. For instance, a systematic exploration of the three-dimensional parameter space considered during our experiments, with 6 steps for each parameter, would require 216 evaluations of the function, whereas the Powell's method can obtain similar results with an order of magnitude less evaluations (between 6 and 37 during the described experiments). In preliminary studies, it has also

be found to be one or two orders of magnitude faster than alternative methods such as genetic algorithms (Marbach and Ijspeert, 2005). An analysis of the results shows that there is clearly space for improvements: particulary, the stopping conditions of the one dimensional optimization and of the Powell's algorithm itself have to be carefully calibrated, in order to minimize the number of evaluations needed and to avoid stopping the algorithm too early (or too late), as it seem to have been the case during some of the experiments. Compared to related work on learning (e.g., Dowling (1997); Kulali et al. (2002)), our approach is very empirical and is fast enough to learn gaits directly on the robot without requiring a simulator or a model. As mentioned in the introduction, we believe this empirical approach is the only viable for many situations, for instance, for complex terrains that not be modelled or simulated accurately enough.

In previous work (Ijspeert and Crespi (2007); see chapter 4), we used results from systematic searches to design interface functions to maintain an optimal gait for a given frequency. Frequency is used as the control parameter that monotonously adjusts velocity, and the interface functions adjust the other parameters (amplitude and phase lag) by linearly interpolating between the optimal values measured with the systematic search. Two interface functions, one for locomotion on a wooden floor and one for locomotion in water, were designed. A human operator could thus easily control the speed (and direction) of locomotion by adjusting the frequency (and the asymmetry of amplitudes), without having to worry about the other control parameters. This is done transparently for the human operator, except for the switch between functions for different environments.

This work extend the previous results by allowing to find optimal interface functions for a given environment much faster. There are two interesting outcomes: (1) It is much less tedious to create a database of interface functions for a variety of environments. This database can be used by the human operator to rapidly switch between different locomotion modes (ideally this decision should be made by the robot itself, see below). (2) The optimization is fast enough to be run during operation time for novel environments. For instance, if the robot is brought to a new terrain for a specific mission (e.g., search and rescue), and one notices that locomotion is slow, the operator could rapidly run the optimization process. The optimization takes in average 20 evaluations (i.e., less than 4 minutes) which seems acceptable for finding a good gait.

Chapter 8

Conclusions

8.1 Conclusion

Amphibious robot elements Amphibious robotics is a rather challenging topic: the complete waterproofing requirements of amphibious robots, for instance, completely influence all aspects in the design process, rendering it more complex. However, the advantages of amphibious robots are evident: if a robot has amphibious properties, it can deal with difficult environments which include water in any form (for example rain, partially flooded terrains, mud, etc.). This is a clear advantage, for instance, for outdoor robots used for inspection or exploration tasks. In this work, we used a modular approach to construct waterproof robots. The actual robots that have been realized are a snake, a boxfish, and a salamander.

Central pattern generators in robots Central pattern generators are more and more used for the control of robots. This biologically inspired control technique is well suited for the control of complex robots having multiple degrees of freedom, as they can generate coordinated control signals for all the joints, receiving only simple inputs. This means that CPGs are a good control method for implementing interfaces to be used by human operators for the interactive control of such robots. They also provide a sort of "abstraction layer" that can hide the complexity of the robot to the end user, also rendering it possible to control different types of robots with the same set of control signals. Finally, CPGs can also easily deal with "difficult" input signals, like those that can be provided by a human operator or learning algorithm. In this work, CPG models for controlling a snake, a fish and a salamander robot have been developed and successfully implemented to run on their onboard locomotion controller. The input signals of the CPGs are very similar for all the models, thus allowing the different robots to be controlled with a limited number of parameters. The design of CPGs remains a complex problem; the models presented here have been inspired from those found in real animals (lamprey and salamander). For

more complex robot structures, a possible solution to the problem is the usage of learning algorithms: as it has been shown in chapter 7, learning algorithms can be a powerful technique to automatically adapt the generated trajectories to the environment. Here the learning was done only on a limited number of parameters, but it can indeed be extended to the whole CPG structure. One of the advantages of the use of CPGs is the possibility to easily include sensory feedback in the control loop; however, this has still not been implemented on the robots presented in this thesis, with the exception of the simple phototaxis behaviour of the fish robot.

Contributions to biology Finally, robots proved to be useful tools for verifying biological hypotheses. As explained in chapter 2, they provide for example a "body" with which models of locomotion controllers can be tested to verify whether they actually generate locomotion in a real environment. In the article reproduced in chapter 6, we presented a model of central pattern generator that explains the locomotion control in salamanders and the transition between walking a swimming, using a single control parameter. This model has been successfully implemented and tested on a salamander robot, demonstrating that it can actually generate forward motion of a body, using real actuators, in a real environment.

8.2 Future work

Although most of the limitations of the first generation of elements were addressed and solved with the current generation, some problems remain partially unsolved, and new weaknesses (related to previously unavailable features) appeared.

Despite the efforts to correct problems with waterproofing, small water leakages are still possible. They mainly concern body elements, and owe to two main sources:

- The absence of a pair of screws (due to lack of space) at the horizontal center of the elements causes an insufficient compression of the O-ring. This problem is partially solved by using a silicone based sealant around the O-ring when closing the elements.
- The forces applied to the output axis during vertical movements (e.g., when lifting the robot, or when it has to overcome a small obstacle) can detach the brass axis from the flexible polyurethane rubber into which it is inserted, thus opening the way for possible water leakages.

These waterproofing problems will be addressed in the future third prototype (which is currently under development), using a new closing system based on permanent magnets (instead of screws), thus also reducing the damages to the elements when unmounting them.

The online learning algorithm (see chapter 7) has been used to optimize the locomotion velocity. However, it would also be interesting to apply this algorithm to other functions, e.g., the power consumption or a combination of power consumption and velocity.

The limited computational capabilities of the PIC18F2580 microcontroller used for the locomotion controller (a 8-bit microcontroller running at 40 MHz, obtaining a speed of 10 MIPS) required strong optimization of the code implementing the CPG locomotion controller (for a description of the controller, see chapters 4 and 6). For instance, the CPG had to be implemented with fixed-point arithmetics and a large use of lookup tables, therefore obtaining an acceptable execution speed (mostly entirely using the available RAM and program memory). The use of a fastest microcontroller (a 60 MHz 32-bit ARM microcontroller) is planned for the next version of the locomotion controller.

Other needed improvements to the current elements include:

- The replacement of the current global I^2C bus with a faster and more reliable CAN bus. The I^2C would be used only inside elements for local communications between the different components.

- Adding the possibility to retrieve status information from the battery monitoring/protection circuit, therefore allowing the user to know the charging state of all the batteries, and to estimate the energy consumption of the robot and of each individual element.

- Integrating a new microcontroller (e.g., a microcontroller of the PIC18 family with CAN support) in the elements. This will be necessary to implement the CAN bus (as the currently used PIC16F876A does not have any CAN support), and can ease the integration of sensors in the elements.

- Simplify and enhance the reprogramming possibilities of the robot. It would be really useful to give the user the possibility of reprogramming most of the microcontrollers (central locomotion controller, or local microcontrollers inside each element) using the radio link.

- Implement a really distributed CPG, with each couple of oscillators placed in their own element. With the actual hardware this would be too difficult to implement (this would require a multi-master I^2C bus, with all the implied problems, e.g., bus arbitration).

- Having more status information and diagnostic possibilities (both locally on the elements, for example with LEDs, and remotely).

- Adding a degree of freedom to the limb elements, which are currently rigid and interrupt the wave on the element chain; this would result in a better locomotion.

- Adding a supplementary degree of freedom to have the possibility to lift the body (i.e., to remove the current limitation to planar locomotion).
- Adding sensors. The current design does not integrate any sensors, except for the motor incremental encoder. Among the possible sensor types to be included, the following seem the most interesting: infrared distance sensors, accelerometers, light sensors and cameras. The addition of sensors will allow the control loop to be closed, by including sensory feedback in the CPG.

Remerciements

Je tiens tout d'abord à remercier Auke Ijspeert pour avoir supervisé ma thèse, étant toujours disponible pour répondre à toutes mes questions et me conseiller, ainsi que pour avoir développé les modèles de CPG utilisés tout au long de ce travail. Le projet de construire un robot salamandre est sa création, et c'est grâce à ce projet que cette thèse existe.

Pour qu'un projet tel que celui-ci puisse aboutir, la collaboration de plusieurs personnes ayant des compétences différentes est primordiale. Je remercie donc toutes les personnes qui ont d'une manière quelconque contribué à ce projet, et en particulier :

- André Guignard, qui a fait un travail énorme en concevant, construisant et réparant toutes les parties mécaniques des différents robots (et pas uniquement). Sa grande expérience nous a permis de trouver des solutions intéressantes à un grand nombre de problèmes rencontrés lors du développement des robots. Bien que tu ne sois parti à la retraite que depuis un mois, tu nous manques déjà...
- André Badertscher (« Chico »), qui a monté une grande partie des cartes électroniques des robots, ainsi que plusieurs parties mécaniques. Il a aussi la grande capacité de trouver des solutions simples à des problèmes complexes, et des solutions complexes à des problèmes simples...
- Georges Vaucher et Peter Brühlmeier, qui ont routé la plupart des circuits imprimés utilisés dans les robots, ainsi que les différents circuits utilisés en phase de prototypage.
- Jean-Jacques Moreillon, qui a aidé à monter les parties mécaniques de la deuxième génération de robots.
- Francesco Mondada, qui a mis à disposition le contrôleur PD développé au LSA, ce qui nous a permis de commencer rapidement à contrôler des moteurs.
- Daniel Burnier, qui nous a fourni l'interface RS-232–I^2C qui était utilisée pour le contrôle de la première génération de robots.
- Daisy Lachat, qui a imaginé et construit la première version du robot poisson qu'elle a nommé BoxyBot.
- Ariane Pasquier, qui a travaillé au-delà de ce qui lui était demandé pour nous aider à mettre en place l'aquarium avec le robot poisson au forum découvertes.
- Adamo Maddalena, qui a contribué au développement du logiciel qui contrôle la

communication radio des robots.
– Jérôme Braure, qui a développé la simulation de la première génération de robot salamandre, ainsi que Yvan Bourquin, qui a adapté cette simulation à la deuxième génération.
– Fabien Vannel, qui a conçu le capteur tactile employé comme interface utilisateur pour le robot poisson au forum découvertes.

Je remercie également Arianna Menciassi, Hiroshi Kimura et Francesco Mondada pour avoir accepté d'être rapporteurs pour cette thèse et avoir donné leurs conseils pour améliorer la première version de ce document.

Un bon environnement de travail a clairement une grande importance. Avec le temps j'ai pu découvrir que la bonne ambiance qui règne au BIRG (et qui régnait auparavant au LSL avant sa fermeture) n'est malheureusement pas la règle partout... un grand merci donc à tous mes collègues : les contributions de chacun d'entre-vous sont indéfinissables, mais le résultat est excellent.

Avoir des relations sociales positives et équilibrées est essentiel. Je tiens à remercier en particulier Achille, Chiara, Danielle, Fabiana, Fabien, Nicole, Patrik, Vicky et Yariv pour tous les moments passés ensemble : qu'il s'agisse de faire des discussions plus ou moins profondes, de partager un repas ou de se balader au bord du lac, la contribution de chacun a été importante. Merci également à toutes les personnes que je n'ai pas citées ici mais qui ont pu m'apporter quelque chose : lister tout le monde serait pratiquement impossible.

Merci enfin à mes parents, qui m'ont toujours soutenu tout au long de mes études et de ma thèse.

Curriculum vitae

Personal data

Name	Alessandro Crespi
Birth	Locarno (TI), October 21st, 1979
Nationality	Swiss
Address	Biologically Inspired Robotics Group (BIRG)
	EPFL – IC – ISIM – GRIJ
	Station 14
	CH-1015 Lausanne
Phone	+41 21 693 66 30
Fax	+41 21 693 37 05
e-mail	alessandro.crespi@epfl.ch

Education

- **PhD student in computer science (2003–2007)**
  Biologically Inspired Robotics Group (BIRG)
  Ecole Polytechnique Fédérale de Lausanne (EPFL)
- **Computer engineering (1998–2003)**
  Diplôme d'ingénieur informaticien (equivalent to a BSc/MSc in computer science) obtained in March 2003.
  Ecole Polytechnique Fédérale de Lausanne (EPFL)

Publications

Journal papers

- A. Crespi, A. Badertscher, A. Guignard, and A.J. Ijspeert. AmphiBot I: An amphibious snake-like robot. *Robotics and Autonomous Systems*, 50(4):163–175, 2005.

- A.J. Ijspeert, A. Crespi, and J.-M. Cabelguen. Simulation and robotics studies of salamander locomotion. Applying neurobiological principles to the control of locomotion in robots. *Neuroinformatics*, 3(3):171–196, 2005.
- A. Crespi, D. Lachat, A. Pasquier, and A.J. Ijspeert. Controlling swimming and crawling in a fish robot using a central pattern generator. *Autonomous Robots*. Submitted for review.
- A. Crespi, A.J. Ijspeert. Online optimization of swimming and crawling in an amphibious snake robot. *IEEE Transactions on Robotics*. Submitted for review.
- A.J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen. From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315(5817):1416–1420, 2007.

Conference papers

- A. Crespi, A. Badertscher, A. Guignard, and A.J. Ijspeert. An amphibious robot capable of snake and lamprey-like locomotion. In *Proceedings of the 35th international symposium on robotics (ISR 2004)*, 2004.
- A. Crespi, A. Badertscher, A. Guignard, and A.J. Ijspeert. Swimming and crawling with an amphibious snake robot. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*, pages 3035–3039, 2005.
- D. Lachat, A. Crespi, and A.J. Ijspeert. BoxyBot: a swimming and crawling fish robot controlled by central pattern generator. In *Proceedings of the First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob 2006)*, 2006.
- A. Crespi and A.J. Ijspeert. AmphiBot II: An amphibious snake robot that crawls and swims using a central pattern generator. In *Proceedings of the 9th International Conference on Climbing and Walking Robots (CLAWAR 2006)*, 2006.
- A.J. Ijspeert and A. Crespi. Online trajectory generation in an amphibious snake robot using a lamprey-like central pattern generator model. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA 2007)*, 2007.

Supervised student projects

- Stephan Singh, *Simulation neuromécanique de la salamandre en Java*, winter semester 2003–2004.

- Adamo Maddalena, *Développement d'une interface sans fil RS232–I^2C*, winter semester 2004–2005.
- Daisy Lachat, *BoxyBot: design and realization of a fish robot*, summer semester 2005.
- Sacha Contantinescu, *Design of fish robot sensory system*, summer semester 2005.
- Sarah Marthe, *Java Applet for the Locomotion Controller of the Salamander Robot*, winter semester 2005–2006.
- Ariane Pasquier, *BoxyBot, le robot poisson – Finitions et Présentation*, winter semester 2005–2006.

Bibliography

P. Arena. A mechatronic lamprey controlled by analog circuits. In *Proceedings of the 9th IEEE Mediterranean Conference on Control and Automation (MED '01)*, 2001.

J. Ayers and J. Crisman. The lobster as a model for an omnidirectional robotic ambulation control architecture. In R.D. Beer, R.E. Ritzmann, and T.M. McKenna, editors, *Biological neural networks in invertebrate neuroethology and robotics*, pages 287–316. Academic Press, 1993.

J. Ayers, J. Witting, N. McGruer, C. Olcott, and D. Massa. Lobster robots. In T. Wu and N. Kato, editors, *Proceedings of the International Symposium on Aqua Biomechanisms*, 2000.

R. Breithaupt, J. Dahnke, K. Zahedi, J. Hertzberg, and F. Pasemann. Robo-salamander – an approach for the benefit of both robotics and biology. In *Proceedings of the 5th International Conference on Climbing and Walking Robots (CLAWAR 2002)*, 2002.

R. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall, 1973.

G.S. Chirikjian and J.W. Burdick. Design, implementation, and experiments with a thirty-degree-of-freedom 'hyper-redundant' robot. In *ISRAM 1992*, 1992.

H.R. Choi and S.M. Ryew. Robotic system with active steering capability for internal inspection of urban gas pipelines. *Mechatronics*, 12:713–736, 2002.

A.H. Cohen, P.J. Holmes, and R. Rand. The nature of coupling between segmented oscillations and the lamprey spinal generator for locomotion: a mathematical model. *Journal of Mathematical Biology*, 13:345–369, 1982.

J.E. Colgate and K.M. Lynch. Mechanics and control of swimming: A review. *IEEE Journal of Oceanic Engineering*, 29(3):660–673, 2004.

J. Conradt and P. Varshavskaya. Distributed central pattern generator control for a serpentine robot. In *ICANN 2003*, 2003.

A. Crespi and A.J. Ijspeert. AmphiBot II: An amphibious snake robot that crawls and swims using a central pattern generator. In *Proceedings of the 9th International Conference on Climbing and Walking Robots (CLAWAR 2006)*, 2006.

A. Crespi and A.J. Ijspeert. Online optimization of swimming and crawling in an amphibious snake robot. *IEEE Transactions on Robotics*, 2007. Submitted for review.

A. Crespi, A. Badertscher, A. Guignard, and A.J. Ijspeert. An amphibious robot capable of snake and lamprey-like locomotion. In *Proceedings of the 35th international symposium on robotics (ISR 2004)*, 2004.

A. Crespi, A. Badertscher, A. Guignard, and A.J. Ijspeert. AmphiBot I: An amphibious snake-like robot. *Robotics and Autonomous Systems*, 50(4):163–175, 2005a.

A. Crespi, A. Badertscher, A. Guignard, and A.J. Ijspeert. Swimming and crawling with an amphibious snake robot. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*, pages 3035–3039, 2005b.

A. Crespi, D. Lachat, A. Pasquier, and A.J. Ijspeert. Controlling swimming and crawling in a fish robot using a central pattern generator. *Autonomous Robots*, 2007. Submitted for review.

H. Date, Y. Hoshi, M. Sampei, and N. Shigeki. Locomotion control of a snake robot with constraint force attenuation. In *Proceedings of the American Control Conference*, pages 113–118. AACC, June 2001.

F. Delcomyn. Neural basis for rhythmic behaviour in animals. *Science*, 210:492–498, 1980.

X. Deng and S. Avadhanula. Biomimetic micro underwater vehicle with oscillating fin propulsion: System design and force measurement. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*, pages 3312–3317, 2005.

K. Dowling. *Limbless Locomotion: Learning to Crawl with a Snake Robot*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 1997.

G. Dudek, P. Giguere, C. Prahacs, S. Saunderson, J. Sattar, L.-A. Torres-Mendez, M. Jenkin, A. German, A. Hogue, A. Ripsman, J. Zacher, E. Milios, H. Liu, P. Zhang, M. Buehler, and C. Georgiades. AQUA: An amphibious autonomous robot. *Computer*, 40:46–53, 2007.

D. Duff, M. Yim, and K. Roufas. Evolution of PolyBot: A modular reconfigurable robot. In *Proceedings of the Harmonic Drive Intl. Symposium*, Nov. 2001.

M. Fiala. ARTag revision 1. A fiducial marker system using digital techniques. Technical Report NRC 47419, National research council Canada, institute for information technology, 2004.

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

C. Gans. *Biomechanics: an approach to vertebrate biology*. University of Michigan Press, 1974.

G.B. Gillis. Undulatory locomotion in elongate aquatic vertebrates: Anguilliform swimming since sir james gray. *American Zoologist*, 36:656–665, 1996.

J. Gray. The mechanism of locomotion in snakes. *Journal of Experimental Biology*, 23: 101–120, 1946.

S. Grillner, T. Degliana, Ö. Ekeberg, A. El Marina, A. Lansner, G.N. Orlovsky, and P. Wallén. Neural networks that co-ordinate locomotion and body orientation in lamprey. *Trends in Neuroscience*, 18(6):270–279, 1995.

A. Hiraoka and H. Kimura. A development of a salamander robot - design of a coupled neuro-musculoskeletal system. In *Proceedings of the Annual Conference of the Robotics Society of Japan, Osaka*, 2002. (Paper in Japanese).

S. Hirose. *Biologically Inspired Robots (Snake-like Locomotors and Manipulators)*. Oxford University Press, 1993.

S. Hirose and E.F. Fukushima. Snakes and strings: New robotic components for rescue operations. In B. Siciliano and D. Paolo, editors, *Experimental Robotics VIII: Proceedings of the 8th International Symposium ISER02*, pages 48–63. Springer-Verlag, 2002.

A.J. Ijspeert and A. Crespi. Online trajectory generation in an amphibious snake robot using a lamprey-like central pattern generator model. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA 2007)*, pages 262–268, 2007.

A.J. Ijspeert and J. Kodjabachian. Evolution and development of a central pattern generator for the swimming of a lamprey. *Artificial Life*, 5(3):247–269, 1999.

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

A.J. Ijspeert, A. Crespi, and J.-M. Cabelguen. Simulation and robotics studies of salamander locomotion. Applying neurobiological principles to the control of locomotion in robots. *Neuroinformatics*, 3(3):171–196, 2005.

A.J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen. From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315:1416–1420, 2007.

B.C. Jayne. Swimming in constricting (*Elaphe g. guttata*) and nonconstricting (*Nerodia fasciata pietiventris*) colubrid snakes. *Copeia*, pages 195–208, 1985.

B.C. Jayne. Kinematics of terrestrial snake locomotion. *Copeia*, 4:915–927, 1986.

M.W. Jørgensen, E.H. Ostergaard, and H.H. Lund. Modular ATRON: Modules for a self-reconfigurable robot. In *Proceedings of IEEE/RSJ International Conference on Robots and Systems (IROS 2004)*, pages 2068–2073, 2004.

N. Kato. Control performance in the horizontal plane of a fish robot with mechanical pectoral fins. *IEEE Journal of Oceanic Engineering*, 25(1):121–129, 2000.

B. Klaassen and K.L. Paap. GMD-SNAKE2: A snake-like robot driven by wheels and a method for motion control. In *Proceedings of 1999 IEEE International Conference on Robotics and Automation (ICRA 1999)*, pages 3014–3019. IEEE, 1999.

G. Konidaris, T. Taylor, and J. Hallam. HydroGen: Automatically generating self-assembly code for Hydron units. In *Proceedings of the Seventh International Symposium on Distributed Autonomous Robotic Systems (DARS04)*, 2004.

G.M. Kulali, M. Gevher, A.M. Erkmen, and I. Erkmen. Intelligent gait synthesizer for serpentine robots. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA 2002)*, pages 1513–1518, 2002.

D. Lachat, A. Crespi, and A.J. Ijspeert. BoxyBot: a swimming and crawling fish robot controlled by central pattern generator. In *Proceedings of the First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob 2006)*, 2006.

T. Lee, T. Ohm, and S. Hayati. A highly redundant robot system for inspection. In *Proceedings of the conference on intelligent robotics in the field, factory, service, and space (CIRFFSS '94)*, pages 142–149, Houston, Texas, 1994.

J. Liu, I. Dukes, R. Knight, and H. Hu. Development of fish-like swimming behaviours for an autonomous robotic fish. In *Proceedings of Control 2004*, 2004.

J. Liu, I. Dukes, and H. Hu. Novel mechatronics design for a robotic fish. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, pages 807–812, 2005.

J.H. Long, Jr., J. Schumacher, N. Livingston, and M. Kemp. Four flippers or two? Tetrapodal swimming with an aquatic robot. *Bioinspiration and Biomimetics*, 1:20–29, 2006.

Z. Lu, B. Ma, S. Li, and Y. Wang. Serpentine locomotion of a snake-like robot controlled by cyclic inhibitory CPG model. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, pages 96–101, 2005.

D. Marbach and A.J. Ijspeert. Online optimization of modular robot locomotion. In *Proceedings of the 2005 IEEE International Conference on Mechatronics and Automation (ICMA 2005)*, pages 248–253, 2005.

F. Matsuno and K. Suenaga. Control of redundant 3D snake robot based on kinematic model. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA 2003)*, pages 2061–2066, 2003.

K. McIsaac and J. Ostrowski. Motion planning for anguilliform locomotion. *IEEE Transactions on Robotics and Automation*, 19(4):637–652, 2003.

K.A. McIsaac and J.P. Ostrowski. A geometric approach to anguilliform locomotion: Simulation and experiments with an underwater eel-robot. In *Proceedings of 1999 IEEE International Conference on Robotics and Automation (ICRA 1999)*, pages 2843–2848. IEEE, 1999.

O. Michel. Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, 1(1):39–42, 2004.

G.S.P. Miller. *Neurotechnology for biomimetic robots*, chapter Snake robots for search and rescue. Bradford/MIT Press, Cambridge London, 2002.

S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji. M-TRAN: Self-reconfigurable modular robotic system. *IEEE/ASME Transactions on Mechatronics*, 7(4):431–441, 2002.

J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato. An empirical exploration of phase resetting for robust biped locomotion with dynamical movement primitives. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, pages 919–924, 2004.

J. Ostrowski and J. Burdick. Gait kinematics for a serpentine robot. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation (ICRA 1996)*, pages 1294–1299, 1996.

K.L. Paap, M. Dehlwisch, and B. Klaassen. GMD-snake: a semi-autonomous snake-like robot. In *Distributed Autonomous Robotic Systems 2*. Springer-Verlag, 1996.

C. Prahacs, A. Saunders, M.K. Smith, D. McMordie, and M. Buehler. Towards legged amphibious mobile robotics. *Journal of Engineering Design and Innovation*, 1P, 2005.

P. Prautsch and T. Mita. Control and analysis of the gait of snake robots. In *Proceedings of the 1999 IEEE International Conference on Control Applications (ICRA 1999)*, pages 502–507, 1999.

W. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical recipes in C: the art of scientific computing, 2nd edition*. Cambridge University Press, 1994.

R D. Quinn, G.M. Nelson, R.J. Bachmann, D.A. Kingsley, J. Offi, and R.E. Ritzmann. Insect designs for improved robot mobility. In *Proceedings of the 4^{th} International Conference on Climbing and Walking Robots (CLAWAR 2001)*, 2001.

M. Saito, M. Fukaya, and T. Iwasaki. Serpentine locomotion with robotic snakes. *IEEE Control Systems Magazine*, 22:64–81, 2002.

U. Saranli, M. Buehler, and D.E. Koditschek. RHex – a simple and highly mobile hexapod robot. *The International Journal of Robotics Research*, 20(7):616–631, 2001.

M. Sfakiotakis, David M. Lane, and J.B.C. Davies. Review of fish swimming modes for aquatic locomotion. *IEEE Journal of Oceanic Engineering*, 24(2):237–252, 1999.

W. Shen, B. Salemi, and P. Will. Hormone-inspired adaptive communication and distributed control for self-reconfigurable robots. *IEEE Transactions on Robotics and Automation*, 18(5):1–12, 2002.

W. Shen, M. Krivokon, H. Chiu, J. Everist, M. Rubenstein, and J. Venkatesh. Multimode locomotion for reconfigurable robots. *Autonomous Robots*, 20(2):165–177, 2006.

C. Stefanini, G. Orlandi, A. Menciassi, Y. Ravier, G. La Spina, S. Grillner, and P. Dario. A mechanism for biomimetic actuation in lamprey-like robots. In *Proceedings of the First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob 2006)*, pages 579–584, 2006.

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

T. Takayama and S. Hirose. Development of HELIX: a hermetic 3D active cord with novel spiral swimming motion. In *Proceedings of TITech COE/Super Mechano-Systems Symposium 2001*, pages D-3, 2001.

T. Takayama and S. Hirose. Amphibious 3D active cord mechanism "HELIX" with helical swimming motion. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002)*, pages 775–780. IEEE, 2002.

M.S. Triantafyllou and G.S. Triantafyllou. An efficient swimming machine. *Scientific American*, 272(3):40–48, 1995.

D.P. Tsakiris, M. Sfakiotakis, A. Menciassi, G. La Spina, and P. Dario. Polychaete-like undulatory robotic locomotion. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*, pages 3029–3034, 2005.

D.P. Tsakiris, M. Sfakiotakis, and A. Vlakidis. Biomimetic centering for undulatory robots. In *Proceedings of the First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob 2006)*, pages 744–749, 2006.

Y. Umetani and S. Hirose. Biomechanical study of active cord mechanism with tactile sensors. In *Proceedings of the 6^{th} international symposium on industrial robots*, pages c1-1–c1-10, Nottingham, 1976.

J. Ute and K. Ono. Fast and efficient locomotion of a snake robot based on self-excitation principle. In *Proceedings of the 7^{th} International Workshop on Advanced Motion Control*, pages 532–539, July 2002.

B. Webb. What does robotics offer animal behaviour? *Animal Behaviour*, 60:545–558, 2000.

B. Webb. Can robots make good models of biological behaviour? *Behavioral and brain sciences*, 24:1033–1050, 2001.

B. Webb. Robots in invertebrate neuroscience. *Nature*, 417:359–363, 2002.

B. Webb and R. Reeve. Reafferent or redundant: Integration of phonotaxis and optomotor behavior in crickets and robots. *Adaptive Behavior*, 11(3):137–158, 2003.

C. Wilbur, W. Vorus, Y. Cao, and S.N. Currie. *Neurotechnology for biomimetic robots*, chapter A Lamprey-Based Undulatory Vehicle. Bradford/MIT Press, Cambridge London, 2002.

R. Worst. Robotic snakes. In *Third German Workshop on Artifical Life*, pages 113–126. Verlag Harri Deutsch, 1998.

H. Yamada, S. Chigisaki, M. Mori, K. Takita, K. Ogami, and S. Hirose. Development of amphibious snake-like robot ACM-R5. In *Proceedings of the 36th International Symposium on Robotics*, 2005.

J. Yu, M. Tan, S. Wang, and E. Chen. Development of a biomimetic robotic fish and its control algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(4):1798–1810, 2004.