Hierarchical Control of Limbless Locomotion Using a Bio-inspired CPG Model

Dissertation

zur Erlangung des akademischen Grades
Dr. rer. nat
an der Fakultät für Mathematik, Informatik und Naturwissenschaften
der Universität Hamburg

eingereicht beim Fach-Promotionsausschuss Informatik von
Guoyuan Li
aus Chongqing, China

May 2013

Gutachterinnen/Gutachter

Prof. Dr. Jianwei Zhang

Prof. Dr. Stefan Wermter

Hamburg, 10. Juli 2013 (Tag der Disputation)

Abstract

Limbless robots have the potential ability to perform various highly efficient movements in different environments, taking advantage of the features of limbless locomotion, such as a low center of gravity, a large contact area and a distributed mass. This thesis deals with the locomotion control of limbless robots, concentrating specifically on the study of a hierarchical control architecture as steps toward developing limbless robots capable of 3D locomotion, fast reflex responses and sophisticated responses to environmental stimuli.

First, an overview of limbless robots is presented. Various limbless robots found in the literature are investigated. The survey not only introduces some potential applications for limbless robots, but also establishes a classification of limbless locomotion according to the limbless robots' configurations and auxiliary equipment. Moreover, different approaches to autonomously generate motion patterns for limbless robots are discussed. One type of control approaches based on Central Pattern Generators (CPGs) is emphasized, since it is ideally suited to being applied to a hierarchical control architecture.

Then, a bio-inspired CPG model is proposed. The key problem for developing such a hierarchical control architecture is how to design a CPG based controller that can not only generate various gaits, but also provide a solution for realizing reflex mechanisms as well as integrating sensory feedback. To this end, a CPG model inspired by the neuronal circuit diagram in the spinal cord of swimming lampreys is designed. A set of interneurons described with sigmoid functions and leaky integrators is incorporated into the design of the neural oscillator for rhythmic signal generation. Furthermore, according to the connection between neural oscillators, a chained type and a cyclic type of CPG circuits are developed. The chained type CPG circuit is used for generating traveling waves between oscillators, while the cyclic type CPG circuit is used for producing synchronization and maintenance activities. Through numerical simulations, the control parameters over relevant characteristics of the two types of CPG circuits are studied in detail.

Next, the proposed CPG model is further designed for limbless gait implementation. Considering the configuration of limbless robots with pitch modules and yaw modules connected alternatively, two CPG circuits are applied to the pitch grouped modules and the yaw grouped modules, respectively. Both the necessary conditions for cooperation between the two CPG circuits and the control parameters for fast limbless locomotion are investigated. Four types of limbless gaits, i.e. side winding, rolling, turning and flapping are realized. Results of simulations and experiments show the effectiveness of the proposed CPG circuits in generating limbless locomotion.

After that, in order to realize fast sensory reflex responses, the concept of both sensory neurons and reflex arcs are utilized. Since the proposed CPG model is derived from neural circuit in the spinal cord of lampreys and the existence of

sensory neurons in lampreys has been proven, it is simple and natural to add sensory neurons into the proposed CPG model at the neuronal level. Based on the design of the sensory neurons, a reflex mechanism taking advantage of reflex arcs forms short pathways to bridge external stimuli and the CPG model. Thus fast responses can be made when the external stimuli are afferent to the CPG model. A ball hitting experiment and a corridor passing experiment confirm the feasibility of the reflex mechanism.

Finally, the development of sophisticated responses to environmental stimuli is presented. A framework that combines the CPG model with a learning method is proposed for achieving adaptive limbless locomotion. The key issue of the framework is to find a mapping that converts external stimuli to proper sensory input, so as to modify the output of the CPG model and thus enable the limbless robot to adapt to environments. Two types of learning methods, i.e. a genetic algorithm (GA) based method and a reinforcement learning (RL) based method are applied to the framework, respectively. Through a slope climbing experiment, it is verified that both of them can achieve adaptive limbless locomotion. Furthermore, the performance of adaptive limbless locomotion under the two methods is compared and analyzed, which provides future work with the possible solutions for promoting the performance of adaptive limbless locomotion.

From the results of simulations and experiments, the hierarchical control architecture is confirmed to be a solid platform for improving the locomotive behaviors of limbless robots.

Kurzfassung

Roboter ohne Gliedmaßen sind in der Lage verschiedene Bewegungen in unterschiedlichen Umgebungen auszuführen, während Eigenschaften der extremitätenlosen Fortbewegung, wie ein niedriger Schwerpunkt, eine große Kontaktfläche und eine verteilte Masse, berücksichtigt werden. Diese Arbeit beschäftigt sich mit der Fortbewegungssteuerung extremitätenloser Roboter, wobei sie sich besonders auf das Studium der hierarchischen Kontrollarchitektur, bis hin zu der Entwicklung von Robotern ohne Extremitäten, die Fortbewegung im dreidimensionalen Raum, schnelle Reflexe und fortgeschrittene Reaktionen zu Stimuli der Umgebung aufweisen, konzentriert.

Zunächst wird ein Überblick über extremitätenlose Roboter gegeben. Verschiedene Roboter ohne Arme und Beine werden untersucht. Der Überblick führt nicht nur mögliche Anwendungen extremitätenloser Roboter ein, sondern enthält auch eine Klassifikation von Fortbewegungsarten ohne den Gebrauch von Armen und Beinen, gemäß der Roboterkonfiguration und Hilfsausrüstung. Außerdem werden verschiedene Ansätze, um Bewegungsmuster autonom zu erzeugen, diskutiert. Eine Art der Steuerungsansätze basiert auf Central Pattern Generators (CPG) und wird besonders hervorgehoben, da diese sich sehr gut eignet, um in einer hierarchischen Kontrollstruktur angewendet zu werden.

Dann wird ein biologisch-inspiriertes CPG-Modell vorgeschlagen. Das Schlüsselproblem, um eine hierarchische Kontrollarchitektur zu entwickeln, liegt darin einen CPG basierten Controller zu definierne, der nicht nur unterschiedliche Gangarten, sondern auch eine Lösung bietet, um reflexartige Mechanismen, sowie Sensor-Feedback zu integrieren. Schließlich wird ein CPG-Modell, welches von dem neuronalen Kreislauf-Diagramm der Wirbelsäule von Lampreten (Neunauge) inspiriert ist, entwickelt. Eine Menge von Verbindungs-Neuronen, beschrieben durch die Sigmoid-Funktion und Leckintegratoren, ist in das Design eines neuronalen Oszillators zur Erzeugung von rhythmischen Signalen miteingeschlossen. Desweiteren wurde, entsprechend der Verbindung zwischen neuronalen Oszillatoren, ein kettenartiger und ein zyklischer Typ des CPG Netztes entwickelt. Der kettenartige Typ des CPG Netzes wird benutzt um Wellen zur Fortbewegung (travelling waves) zwischen Oszillatoren zu erzeugen, während der zyklische Typ des CPG Netzes die Rolle der Synchronisation und Aufrechterhaltung des Verhaltens innehält. Mit Hilfe numerischer Simulation wurden relevante Charakteristiken der Steuerungsparameter beider CPG Kreisläufe im Detail erforscht.

Das vorgestellte CPG Modell ist weiterhin geeignet, um Gangarten, die keinen Gebrauch von Extremitäten machen, zu implementieren. Berücksichtigt man Konfigurationen von Robotern ohne Extremitäten, welche abwechselnd aus Modulen zusammengesetzt sind, die im Euler'schen Sinne gieren und neigen, so werden zwei CPG-Kreisläufe zugewiesen; ein Kreislauf ist für die Gier-Gruppe und einer für die Neige-Gruppe angedacht. Die beiden Notwendigen Bedingungen zur Kooper-

ation beider CPG Kreisläufe und die Steuerparameter, die schnelle Fortbewegung ohne Arme und Beine ermöglichen, wurden ermittelt. Vier unterschiedliche Typen von Gangarten wurden entwickelt, wie z.B. seitliches Winden, Rollen, Drehen und Flappern. Ergebnisse von Simulationen und Experimenten zeigen die Effektivität des vorgestellten CPG Kreislaufes beim Erzeugen von Fortbewegung ohne Extremitäten.

Darauf wird das Konzept von sensorischen Neuronen und des Reflexbogens benutzt, um schnelle sensorische Reflexantworten zu realisieren. Da das vorgeschlagene CPG Modell vom neuronalen Kreislauf in der Wirbelsäule von Lampreten abgeleitet ist und die Existenz sensorischer Neuronen in Lampreten bewiesen ist, ist es naheliegend, dem vorgeschlagenen CPG Modell in der neuronalen Ebene, sensorische Neuronen hinzuzufügen. Basierend auf dem Design sensorischer Neuronen bildet ein Reflexmechanismus, der Reflexbögen berücksichtigt, kurze Wege, um externe Reize und das CPG Model zu verbinden. Somit kann schnell reagiert werden, wenn externe Reize dem CPG Modell gegenüber afferent sind. Ein Ballwurf-Experiment und das Passieren eines engen Korridors bestätigen die Realisierbarkeit der Reflexmechanismen.

Zum Abschluss wird die Entwicklung einer durchdachten Reaktion auf Stimuli der Umgebung präsentiert. Ein Rahmenwerk zur Erzielung von adaptiver extremitätenloser Fortbewegung, welches das CPG Model mit einer Lernmethode kombiniert, wird vorgeschlagen. Das Schlüsselproblem des Rahmenwerkes ist es, eine Abbildung zu finden, welche externe Reize sinngemäß zu Sensoreingaben konvertiert, sowie die Ausgaben des CPG Modell, welches den extremitätenlosen Roboter befähigt sich an seine Umgebung anzupassen. Zwei verschiedene Lernmethoden, eine auf genetische Algorithmen (GA) basierende Methode und bestärkendes Lernen (Reinforcement Learning – RL), sind in das Rahmenwerk integriert. Mit Hilfe eines Experiments, in dem ein Anstieg erklommen wird, wird gewährleistet, dass durch beide Methoden adaptive Fortbewegung ohne Extremitäten erreicht wird. Zudem wird die Effizienz der resultierenden adaptiven extremitätenlosen Fortbewegung beider Methoden verglichen, was zukünftige Aufgaben, mögliche Lösungen zu finden, die die Effizienz der adaptiven Fortbewegung ohne Arme und Beine verbessern.

Die hierarchische Kontrollarchitektur wurde mit Hilfe von Ergebnissen der Simulationen und Experimenten als solide Plattform zur Verbesserung des Verhaltens bei der Fortbewegung von extremitätenlosen Robotern bestätigt.

Acknowledgements

This thesis is the result of my doctoral study from September 2009 to May 2013 at the Department of Informatics at the University of Hamburg. The research belongs to the project "Biologically Inspired Modular Climbing Caterpillar Robot Using Passive Adhesion" and is supported by the Deutsche Forschungsgemeinschaft (DFG).

First I would like to thank my supervisor Prof. Dr. Jianwei Zhang who gives me the opportunity to join the project and provides me with all the necessary facilities. I am grateful to his academic support and encouragement throughout my doctoral study. Many thanks as well to Prof. Dr. Houxiang Zhang for his constructive feedback on my research. For me he is like a second mentor who not only guide me how to do research, but also teach me how to contribute scientific results in publications.

Thanks also go to my colleagues at the TAMS group. Specifically I would like to thank Lu, not only for helping me to deal with the affairs related to my study, but also for her concern in my daily life. I thank Bernd, Manfred and Gang Cheng for helping me to realize on-site experiments. I thank Norman, Andreas, Dennis, Hannes and Benjamin for their good comments and suggestions. I thank Junhao, Junyu, Hansong, Jian Chen and Bo Sun for their support and the happy time we had together.

In addition, I would like to thank Juan González Gómez for all the things I have learned from him. I am grateful to Fernando Herrero Carrón, from whom I got a lot of valuable advice and ideas. I also thank Dennis Krupke for his programming help.

Finally, I must give thanks to my parents for all the effort they have made so that I can face all the difficulties and finish my study.

Guoyuan Li
Hamburg, May 2013

Publications

The following is a list of publications produced during the doctoral study.

- Guoyuan Li, Houxiang Zhang, Fernando Herrero Carrón, Hans Petter Hildre and Jianwei Zhang (2011). A novel mechanism for caterpillar-like locomotion using asymmetric oscillation. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 164–169, Budapest, Hungary.

- Guoyuan Li, Jian Chen, Houxiang Zhang, Shengyong Chen and Jianwei Zhang (2013). Design of a lamprey spinal generator for reimplementation of sensory reflex mechanism and realisation of adaptive limbless locomotion. *IEEE Transactions on Robotics*, submitted.

- Guoyuan Li, Houxiang Zhang, Jianwei Zhang and Robin T. Bye (2013). Development of adaptive locomotion of a caterpillar-like robot based on a sensory feedback CPG model. *Advanced Roboticis*, submitted.

- Dennis Krupke, Guoyuan Li, Jianwei Zhang, Houxiang Zhang and Hans Petter Hildre (2012). Flexible modular robotic simulation environment for research and education. In *Proceedings of the 26th European Conference on Modeling and Simulation (ECMS)*, Koblenz, Germany.

- Houxiang Zhang, Gionata Salvietti, Wei Wang, Guoyuan Li, Junzhi Yu and Jianwei Zhang (2010). Efficient kinematic solution to a multi-robot with serial and parallel mechanisms. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6101–6106, Taibei, Taiwan.

- Jian Chen, Guoyuan Li and Jianwei Zhang (2013). A knitro-based realtime locomotion method for imitating the caterpillar-Like climbing strategy. Submitted to *the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Contents

1.1 Motivations

Animal movements are usually related to where they live, how they obtain food and how they escape from predation. Survival pressures force animals to move from one place to another in an efficient manner. Their movements therefore require reasonable coordination of articulation, so that animals can overcome friction and gravity and push themselves forward.

In biology, researchers have been long attracted to the principles of animal locomotion at the neural level. Neurobiological studies of various vertebrates have shown that rhythmic movements are generated in the spinal cord by a central pattern generator (CPG) (Grillner, 1985). CPGs can produce rhythmic signals that control muscular activity to generate rhythmic patterns, such as serpentine creeping, peristaltic crawling and anguilliform swimming. Besides that, CPGs also have the characteristic of neuromodulation and the ability of sensory feedback integration. On the one hand, CPGs can be modulated to shape locomotion behaviors, for example, to adjust the speed of locomotion or to change the length of a stride. On the other hand, CPGs can respond to sensory feedback to alter the pattern of locomotion, which help animals to adapt to their surroundings during locomotion.

In robotics, recently, with the understanding of the biological control architecture in natural creatures, robots can be physically modelled on the example of natural animals. A lot of interesting biologically inspired robots have been developed, such as humanoid robots (Hirai et al., 1998; Gouaillier et al., 2009), quadruped robots (Kolter et al., 2008; Raibert et al., 2008), snake-like robots (Hirose and Yamada, 2009; Transeth et al., 2008b; Hatton and Choset, 2010) and fish-like robots (Yu et al., 2004, 2011; Stefanini et al., 2012). For the motion control problem in bio-inspired robots, the employed approaches can be classified as numerical techniques, geometric methods, intelligent control techniques as well as CPG-based methods.

As a bio-inspired alternative to other algorithm-based control systems such as finite state machines, CPG-based methods have been increasingly used for locomotion control in robots with multiple joints, especially for online rhythmic trajectory generation (Ijspeert, 2008). Many appealing properties including distributed control, the ability to deal with redundancies, fast control loops and allowing the modulation of locomotion by simple control signals make CPG models suitable for different modes of locomotion motion. Furthermore, the combination of CPG with other control methods results in a higher locomotion adaptability. For instance, Arena et al. (2005b) used a multi-template approach to construct appropriate cellular neural networks to achieve a VLSI chip CPG controller; based on the Ekeberg's CPG model of lampreys, Ijspeert et al. (1998) and Patel et al. (2006a) adopted evolutionary algorithms to evolve the CPG's neural parameters and connection weights for improved performance; Crespi and Ijspeert (2008) utilized CPGs and a gradient-free optimization algorithm to accomplish online swimming and crawling gaits.

This thesis concentrates on the locomotion control of limbless robots. Even though limbless locomotion patterns have been investigated and implemented by some researchers using kinematic models or sinusoidal function based methods (Dowling, 1997; Gonzalez-Gomez et al., 2007; Tesch et al., 2009), CPG models are rarely employed on limbless robots for 3D locomotion. Considering the biological features of CPG models, it would be much easier to generate and modulate limbless gaits in a CPG-based control architecture. It is also possible to integrate some biological mechanisms such as the reflex mechanism into the control architecture to realize fast reflex response. Furthermore, since CPG models usually offer a good substrate for learning and optimization algorithms, they are suited to be applied to a hierarchical control architecture. Taking the CPG model as the foundation of the control architecture would facilitate the design of the higher-level control.

Based on the two points above, we will focus on designing and implementing a hybrid control hierarchy including the reflex level, CPGs with biological features and a higher intelligence level to considerably improve the locomotive behaviors of limbless robots.

1.2 Goals

The modular approach has many advantages for designing a multi-functional mobile robot. It enables robots to reconfigure, which is essential for tasks that are difficult for a fixed-shape robot. It also makes a mobile robotic system versatile, robust, cost-effective and fast to prototype, so that different robots can be reconfigured fast and easily for the exploration, testing and analysis of new ideas.

In this thesis, a limbless robot combined with the modular approach is selected as our test bed. The robot is supposed to (i) have full locomotion capabilities in a 3D environment and (ii) be able to interact with the environment. More precisely, the robot should be not only capable of generating various limbless gaits, but also able to react in a sophisticated way to environmental stimuli with the help of multiple

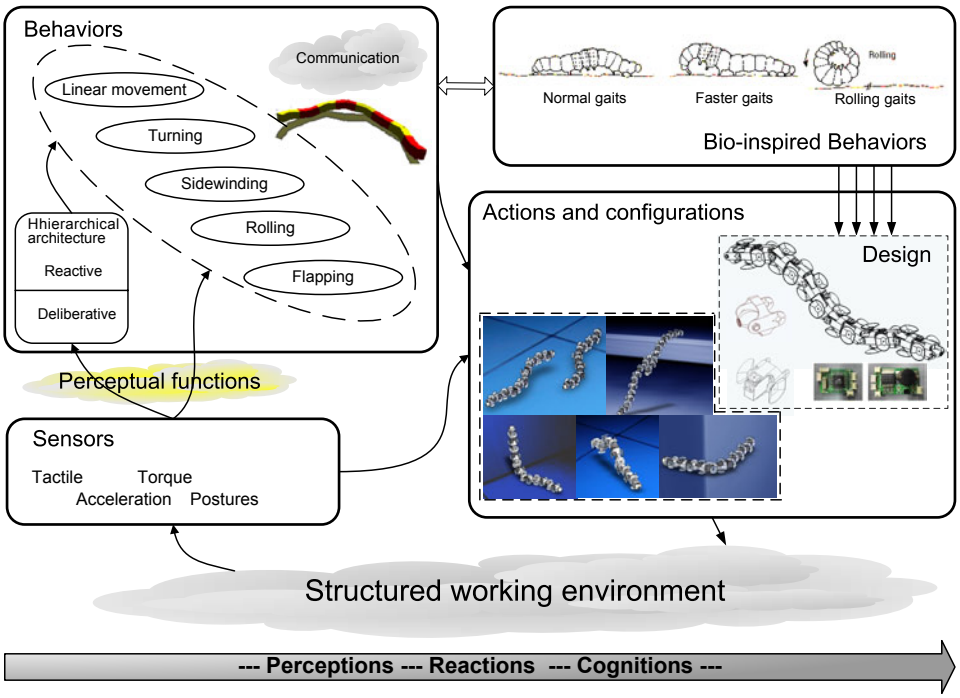


Figure 1.1: The fully functional integration of the limbless robots.

sensors. Figure 1.1 illustrates the idea.

A control architecture that integrates locomotion control, reactive control and deliberative control is considered to be sufficient for controlling the modular limbless robot. We have identified the following objectives for the development of the control architecture:

- The system should provide a reflex mechanism for internal sensor-motor planning that is integrated with the real sensor-motor pathways of the physical robot.

- The architecture should allow for the integration of perceptions and of control templates for motor skills (behaviors) based on CPGs.

- The system should integrate an elementary locomotion control which depends on bio-inspired control methods and higher-level interaction.

To meet these requirements, a hierarchical control architecture including a reflex level, CPGs with biological locomotion features and a learning algorithm for sensor-servo-based behavior control and active perception of the environment to complete the system is designed, as shown in Figure 1.2. The main task of this architecture is to guarantee fast reactive responses to environmental stimuli while providing a means for deliberative long/mid-term goal-oriented behavior.

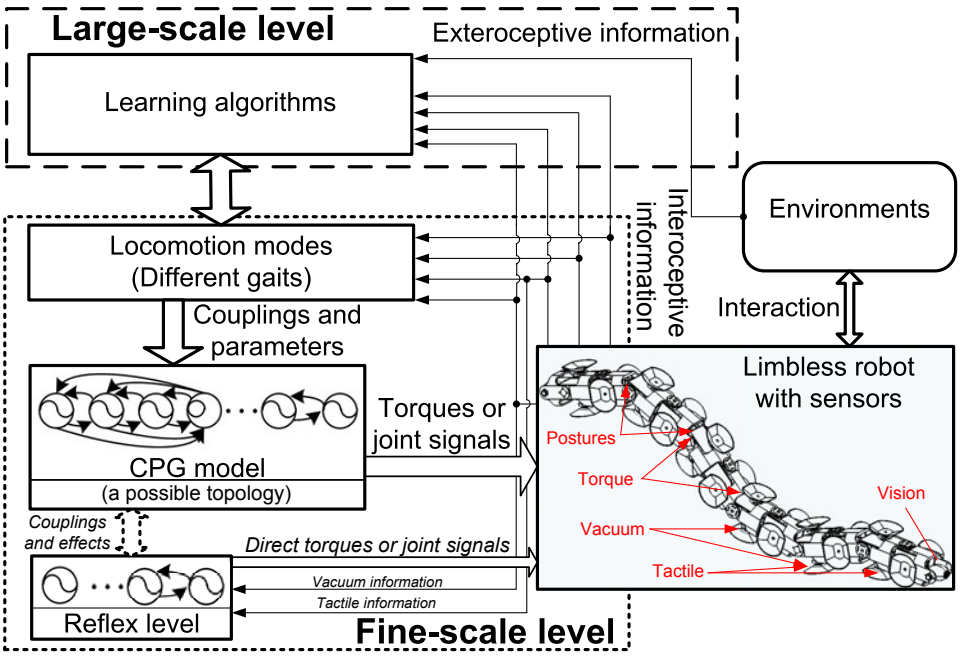


Figure 1.2: A hierarchical control architecture.

The modular limbless robot has to deal with the environment on two different levels. On the fine-scale level, a CPG controller is in charge of limbless gaits generation. The parameter space of the CPG model should be typically low, which is suitable to further implement learning algorithms to achieve new functionalities besides those intrinsically provided by the connectivity and the dynamics of the individual elements.

In addition to the CPG model, there exists a reflex level beneath the CPG model. The reflex mechanism is a special part of the control concept. The coupling between CPGs and the reflex level is strong. On the one hand, the reflex will guarantee the locomotion safety and reliability. The behaviors at this level have a high priority. On the other hand, some kinds of behaviors such as simple gait switching and emergency action can be decided directly at this level according to the stimuli of interoceptive information and the robot's working state. While other control functions are still decided by both the higher level and the reflex level to achieve an optimized solution.

External and internal sensors on the robot are responsible for collecting information about the operational environment and reflecting the self-status of the robot respectively. All sensor inputs will be sent to the large-scale and the fine-scale levels at the same time. Meanwhile, sensor information related to reaction control will generate the sensor-motor reflex in parallel.

On the large-scale level, the robot may travel from one location to the next by applying adaptive locomotion behaviors. To generate adaptive locomotion, some

learning algorithms, such as a genetic algorithm or reinforcement learning, can be utilized on the large-scale level. They can help the robot to gather sensory information, analyze the robotic internal status and surrounding stimuli, and learn the proper reactions. After the learning process, the robot will adapt locomotion modes or gaits to the environment while being automatically controlled by the fine-scale level. Thus, the control architecture is closed and complete.

1.3 Main contributions

The content of this thesis can be divided into three main parts: (i) a study of limbless robots through a review of the existing literature; (ii) the development of the fine-scale level for limbless gait generation and fast reactions based on CPGs; and (iii) the development of the large-scale level for adaptive limbless locomotion using genetic algorithms and reinforcement learning algorithms. The main novelties and contributions of this work are:

- A hierarchical control architecture including CPGs containing a reflex mechanism as well as biological features and learning algorithms for sensor-servo-based behavior control and active perception of the environment to complete the system;
- A novel CPG model inspired by the neuronal circuit diagram in the spinal cord of swimming lampreys that
  - provides simple but uncoupling parameters for output modulation;
  - provides a solution for realizing a reflex mechanism originating in physiological findings;
  - allows for the integration of sensory feedback;
- The design of four types of limbless gaits, as well as the reflex mechanism;
- The development of closed-loop control for adaptive locomotion on a limbless robot.

1.4 Structure of the document

In the following chapter we first give a brief overview of some relevant limbless robots, which assists in conveying an idea of limbless locomotion. Then, in Chapter 3, we propose our novel CPG model and discuss the CPG circuits as well as the property of parameters modulation in more detail. Following this, Chapter 4 presents the implementation of four types of limbless gaits based on the CPG circuits. Afterwards, we introduce how to add neural pathways into the CPG model that permit fast response to stimuli in Chapter 5. To show the ability of sensory feedback integration into the CPG model, in Chapter 6 and 7, we present a

closed-loop scheme of our hierarchical control architecturethe that employs a genetic algorithm and reinforcement learning method to achieve adaptive locomotion on a limbless robot. Finally, we conclude our work and discuss future work.

Chapter 2

Overview of Limbless Robots

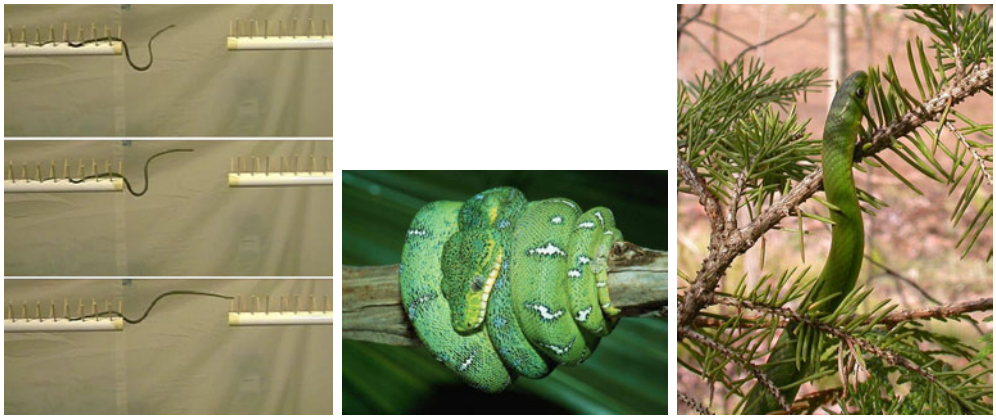
Contents

2.1 Introduction

Limbless animals have a wide range of locomotive capabilities, such as serpentine creeping, peristaltic crawling and anguilliform swimming. These animals, due to lack of legs, use their bodies to generate movements. They propagate flexural waves along the length of their bodies, so that the force generated between them and the surrounding environment can propel them forward. Compared to other forms of locomotion, limbless locmotion provides the following advantages in animals (Hopkins et al., 2009):

- Limbless animals have a linear structure with a compact cross-section that allows them to cross through thin holes and gaps.
- They can climb trees, rocks, and any other vertical surface. This is achieved by lifting the front one-third of their bodies up while setting the lower two-thirds of their bodies as a base.
- Their locomotion gaits are very stable. Since they keep most of their bodies in contact with the terrain during locomotion, they have a low center of mass and a large contact area that prevent them from falling over.
- They can act as manipulators when they are clenching prey or twining around tree branches.

Figure 2.1 is an illustrative example of limbless locomotion that other forms of locomotion can not accomplish.



Crossing gap Twining branch Climbing tree

Figure 2.1: Advantages of limbless locomotion.

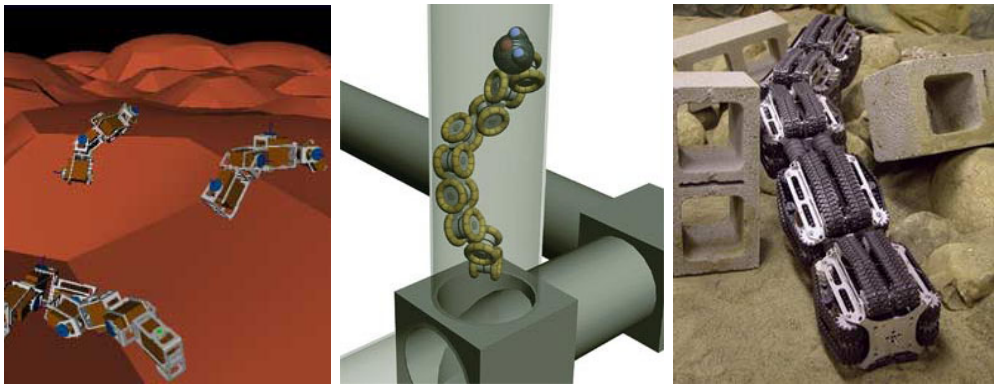
Motivated by the advantages of limbless locomotion, researchers have been interested in replicating these movements in mechanisms for centuries. The limbless design offers significant benefits for dealing with complex environments which traditional machines with appendages such as wheels or legs fail to traverse. Limbless robots have several potential advantages over wheeled and legged robots (Dowling, 1997):

- **Stability**: Copied from the morphology of limbless animals, limbless robots naturally inherit their configuration features, including distributed body mass, low center of gravity and multiple contact points. Thus there is no need to worry about stability in this kind of robots. In contrast, stability is of great concern to wheeled and legged robots. They suffer an impact with the ground and will fall over if the center of mass moves out of the bounds of contact points.

- **Terrainability**: Wheeled and legged robots are sometimes limited in the type and scale of terrains. While limbless robots are supposed to be able to traverse a wide variety of terrains since they can learn diverse locomotion modes from nature. This feature enables the use of limbless robots in more strict terrains, such as passing through pipes, climbing up and over obstacles, passing terrains with soft grounds.

- **Redundancy**: Limbless robots have redundant designs that repeat simple actuators in sequence many times. The modular approach enables the robotic system a robustness to the extent that even if one of the actuators fails, the robots are still able to move.

In spite of many advantages, limbless robots are also limited by their own configuration. Their shortcomings include poor payload capacity, slow speed of movement and a large number of degrees of freedom that need to be controlled, which result in



Exploration Inspection Search and rescue

Medical treatment Reconnaissance

Figure 2.2: Potential applications for limbless robots.

challenges in the design and control of limbless robots. The rest of this chapter will focus on the two problems and present the current status of research on limbless robots.

2.2 Applications

In the past, robots have only been applied in structured environments due to technical limitations. With the development of robotic technology, especially in the areas of sensing and control, the usage of robots has been extended to unstructured environments. Considering the characteristics of limbless robots, there would be several applications that limbless robots are suitable for. Figure 2.2 illustrates some possible applications, including exploration, inspection, search and rescue, medical treatment and reconnaissance.

- **Exploration**: Robots have the benefit of being able to explore many hazardous areas a human could not reach, such as deep sea areas and the planets in space (Yim et al., 2003). For exploration tasks, there is no prior knowledge about the environment. A robot may get stuck, fall over, and even get damaged resulting in loss of mobility. To avoid such situations, the robot must be good at terrain traversing. Limbless robots satisfy this requirement. As

highlighted in the last section, a limbless robot with redundant design can distribute body mass over a large area to support itself. Furthermore, its low center of gravity can prevent it from falling over. NASA has developed such a limbless robot for space exploration. The robot may help explore other planets and perform construction tasks in space.

- **Inspection**: In many cases, it is necessary for humans to inspect environments that are either too small or too dangerous for them. For instance, a human may need to detect the blockage of pipe networks, or to inspect a reactor in a nuclear plant. The inspection tasks usually take place in unstructured and tight environments. To accomplish inspection tasks, robots are required to be flexible and sensitive. Limbless robots carrying onboard sensors are considered to be very suitable for such tasks. On the one hand, the benefits of a small cross section enables the robot to climb through narrow and tight environment easily. On the other hand, the onboard sensors can help the robot to collect environmental information and carry out efficient inspection and accurate localization.

- **Search and rescue**: A search and rescue task requires rescuers to find trapped survivors in a collapsed structure after earthquakes or other disasters. Survivors may be buried amongst the debris that both human rescuers and trained dogs usually fail to find, much less to approach. Additionally, searching in partially-collapsed buildings is a dangerous task for rescuers due to potential risk from further collapse. Limbless robots would be a valuable aid to rescuers. As mentioned earlier, a limbless robot has redundant design and a small cross-sectional area which allow it to penetrate deep into the rubble. The limbless robot can be outfitted with a variety of sensors, such as cameras, microphones and infrared detectors so as to obtain infomation in the rubble. Sensing information can be sent back to rescuers in real time to help locating survivors. This would make the search and rescue task in the rubble faster and more effective.

- **Medical treatment**: Limbless robots could be potential medical devices for aiding in surgical procedures. The robots can help to overcome the limitations of traditional surgeries. For example, instead of opening large sections of skin and tissue during the surgery, a limbless robot with small cross section allows surgeons to operate with far less damage to the body. The robot could crawl into interior tissues along the small incision and make further diagnosis using onboard sensors. This would dramatically reduce the patients' suffering and healing time. Furthermore, with the help of limbless robots, surgeries will become faster and easier, resulting in reduction of medical costs.

- **Reconnaissance**: Limbless robots could also be applied to military reconnaissance. Because of their small size and moving low on the ground, a limbless robot is not easy to find, especially when it is disguised with protective

color. Moreover, by means of the diversity of limbless motion, the robot is able to move surreptitiously into hostile areas and further use onboard sensors for reconnaissance. Israel has developed such a reconnaissance robot with a camera and microphone installed in its head and plans to apply it for combat missions.

2.3 State of the art

Limbless robots have become a hot topic in the last few decades. A growing number of research groups has become actively involved in limbless robotics research. This section is not a complete review of all limbless robots. Instead, investigation is focused on those limbless robots that were developed in some important research labs.

2.3.1 Active cord mechanism

As one of the pioneers in this field, Hirose started the study of snake-like robots in the early 1970s (Hirose and Yamada, 2009). He first developed a snake robot called the Active Cord Mechanism (ACM) that could perform lateral undulation. The purpose of his work was to understand the mechanism of locomotion in real snakes. Latter, he developed a series of snake-like robots that followed from this work. The functionality of the robots has been promoted from 2D motion to 3D motion. Recently, he and his colleagues developed an amphibious snake robot that can move as easily on the ground as in water. Figure 2.3 shows the different prototypes in Hirose's lab developed from the beginning to the present.

The first one is the ACM III, which has a length of 2 meters and a weight of 28 kg. The robot is made up of 20 joints with a yaw-yaw connection. Since each joint consisting of a servo system can only bend to the left and right, the robot is constrained to 2D motion. Hirose used this configuration to study the serpentine gait. To generate the forward propulsion for the serpentine gait, he installed small wheels on the bottom of each joint along the direction of the body. The idea is to make it easy for the robot to slide in the tangential direction and difficult to slide in the normal direction. As a result, the ACM III achieved serpentine movement at a speed of approximately 40 cm/s (Hirose, 1993).

The ACM Revision 1 (ACM-R1) was built in 1995. Hirose and Endo (1997) redesigned the robot with 16 smaller modules and utilized wireless communication to control it. The robot can move with a velocity of about 50 cm/s, which becomes faster than the ACM III. Furthermore, it has higher mobility with serpentine locomotion as due to a skate blade placed under each joint, the robot succeeded to glide on ice in a skating rink (Endo et al., 1999).

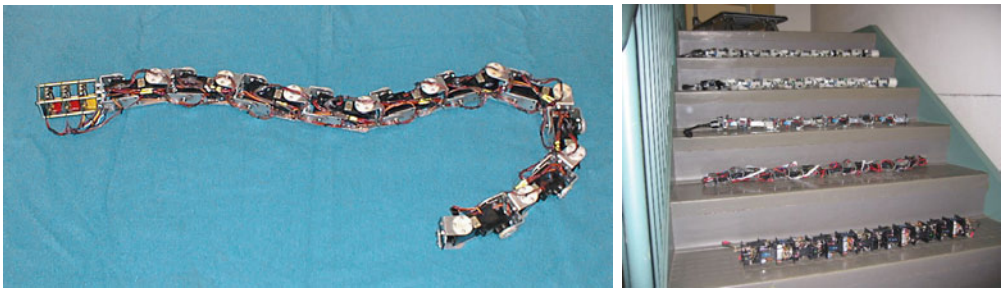
In the previous mechanical prototypes, the movement was limited to the 2D plane. In 2000, a three-dimensional type ACM called ACM-R2 was developed (Togawa et al., 2000). The ACM-R2 added a new degree of freedom in the pitch direction at each joint so as to realize three-dimensional mobility. To facilitate

Figure 2.3: The ACM family in the Hirose-Fukushima Robotics Lab.

the study of limbless locomotion, an improved design was used in the ACM-R3 (Mori and Hirose, 2002). The ACM-R3 is 1755 mm long and weighs 12.1 kg. It has a new configuration with 10 pitch modules and 10 yaw modules connected alternately. Each module is equipped with large passive wheels with diameters of 110 mm. These passive wheels wrap around the whole body so that the robot can make contact with the surface more easily. The ACM-R3 was utilized as a research platform by Hirose to investigate new types of locomotion, including serpentine, lateral rolling, sinus-lifting, lift rolling and pedal wave.

To ensure the robots' practicality, Yamada and Hirose (2006) developed a version of ACM known as the ACM-R4 in 2004. The ACM-R4 consists of 9 modules and weighs 9.5 kg. It has a length of 1100mm with a cross-section of 135 mm by 135 mm. Although the ACM-R4 has the same configuration as the ACM-R3, it has 3 new characteristics including active wheels, dust- and water-proofing and overload protection. The ACM-R4 uses motors to drive the wheels. The use of active wheels enable the robot to be used in extremely complex conditions like narrow pipelines or disaster sites.

The ACM-R5 is an amphibious robot developed in 2005 (Yamada et al., 2005). It can move both on land and in water, due to a dust- and waterproof packaging. The robot is composed of 9 joints with a length of 1750 mm and a weight of 7.5 kg. Each joint of the ACM-R5 consists of an universal joint and bellows. The universal joint works as bones and the bellows act as an integument. There are 6 swimming fins attached around each joint. Moreover, on the tip of each fin, passive wheels are installed to allow for creeping motion on the ground. The ACM-R5 can effect forward motion at a speed of about 40 cm/s on the ground as well as in water with the help of these fins and wheels.



(a) Dowling's snake robot (b) Modular snake robots

(c) Stair climbing (d) Pipe rolling (e) Rocks traversing

Figure 2.4: CMU's snake robots in the Biorobotics Lab at CMU.

2.3.2 CMU's snake robots

Kevin Dowling made a great contribution to snake robots in the middle of the 1990s at Carnegie Mellon University (Dowling, 1997). He developed a snake robot for studying the automatic gait generation using learning algorithms. The robot is 1020 mm in length and weighs 1.48 kg in total, as shown in Figure 2.4(a). It consists of 10 links for a total of 20 degrees-of-freedom. Each link contains 2 servos and has a diameter of 65 mm. The purpose of developing the robot is to support gait optimization studies. Dowling developed a framework for teaching the robot to move. He has selected specific metrics including energy, time and weight as the measure of performance for learning locomotion. As a result of his work, a number of interesting gaits including several snake-like gaits as well as some novel gaits not found in real snakes are developed.

Howie Choset has been researching and building snake-like robots at the robotics institute of CMU for years. He and his students have developed several modular snake robots (Wright et al., 2007), as shown in Figure 2.4(b). Initially, they built laser-cut plastic modules and employed them to construct the modular snake robots with parallel joint axes. Because the configuration restricted the robots to 2D motions, more complex tasks are impossible to accomplish. Therefore, they changed the original configuration into an orthogonal connection. Furthermore, they strengthened the module design by using aluminum modules and higher-torque servos. Based on the refinement of their previous snake robot designs, recently they developed a new prototype of modular snake robot called unified snake robot (Wright et al., 2012; Johnson et al., 2011). The robot consists of 16 modules and

AmphiBot I

AmphiBot II

Figure 2.5: AmphiBot in the Biorobotics Lab at EPFL.

has a length of 940 mm with a diameter of 51 mm. The mass of the robot is 2.9 kg. It utilizes a tether for power connection. The robot also has several built-in sensors, including wireless camera, accelerometer, gyroscope, as well as sensors for measuring temperature, voltage, motor current and joint angles.

This series of modular snake robots is used for the research on various snake-inspired gaits (Tesch et al., 2009; Hatton and Choset, 2010). Choset and his colleagues have developed a class of gaits called parameterized gaits. They used sinusoids as the basis for all of the basic gaits. They also developed scripted gaits for the robot to complete a desired task, such as stair climbing, pipe rolling and rocks traversing, as shown in Figure 2.4(c)-(e).

2.3.3 AmphiBot

Ijspeert et al. developed a biologically inspired snake-like robot called AmphiBot, as shown in Figure 2.5. It is designed as an amphibious robot capable of anguilliform swimming and snake-like undulation. Their research goals are taking the AmphiBot as the test-bed to (i) develop novel types of controllers based on some bio-inspired concepts, such as central pattern generators (CPGs), and (ii) investigate hypotheses of how central nervous systems implement locomotion in real animals.

The first prototype, AmphiBot I was developed around 2004 (Crespi et al., 2004, 2005a). AmphiBot I has some amphibious characteristics. On the one hand, the robot is equipped with a set of removable passive wheels, which allows the robot to perform serpentine locomotion on the ground. On the other hand, the robot is slightly buoyant so that it can float on the water when swimming. In addition, AmphiBot I is composed of several identical segments, named elements. This modular construction enables individual elements to be added or subtracted quickly. Each element has a length of 70 mm and a cross-section of 55 mm by 33 mm. The element is molded using polyurethane so that each individual element is made waterproof instead of having a covering over the entire robot. The element has a single degree of freedom. It contains four components: a body, two covers, and a connection piece. In order to have distributed actuation, power and control,

OT-8 OT-4

Figure 2.6: OmniTread in the Mobile Robotics Lab at UM.

the body part of each element is designed to carry its own dc motor, battery and microcontroller. However, AmphiBot I does not support wireless communication. It runs tethered to a power supply.

As a new version of AmphiBot I, AmphiBot II features significant improvement (Crespi and Ijspeert, 2006; Ijspeert and Crespi, 2007). First, AmphiBot II has a simplified construction and allows all the elements to be assembled without soldering. Second, more powerful motors are applied on the robot. The motor can produce 3.5 times as much maximum torque as the original motor. Third, the robot is capable of wireless communication through an internal transceiver. Finally, the robot can directly generate motor commands by a central pattern generator using a microcontroller, without any help from external computers. AmphiBot II has 7 actuated elements and a head element. It has a total length of 772 mm. Each element is 94 mm in length with a cross-section of 55 by 37 mm. AmphiBot II is used to test how the CPG parameters influence locomotion speed. The robot is reported to be able to crawl and swim at a maximum speed of 400 mm/s and 230 mm/s, respectively.

2.3.4 OmniTread

At the University of Michigan (UM) the mobile robotics lab has foucused on the development of serpentine robots in the last few decades. A class of serpentine robots, called OmniTread, are developed for the research of inspections and surveillance in hard-to-reach areas (Granosik et al., 2005). Figure 2.6 illustrates two prototypes of OmniTread robots. OmniTread robots are designed to use active means instead of body undulation to generate forward motion. This active propulsion can provide the robots with more powerful capabilities, such as climbing over high steps, ascending inside pipes and spanning wide gaps.

The first OmniTread robot is called the OT-8 (Borenstein et al., 2005). The OT-8 is 1270 mm in length and weighs about 13.2 kg. It has 5 box-shaped segments connected by 4 pneumatically actuated 2-DOF joints. Each segment has a dimension of 200×186×186 mm. The segment is equipped with tank treads on each

side. Since the OT-8 has active propulsion, covering the maximal possible surface area of the segment can support motion and make the robot indifferent to rolling over. The joint space between segments is 68 mm long. A universal joint is located in the center of the space. The OT-8 uses pneumatic bellows for joint actuation. The pneumatic bellows is able to adjust the stiffness of joints. This means the bellows can not only provide the robot with sufficient torques to lift body segments, but also allow the robot to conform to the terrain naturally. The robot uses 16 pneumatic bellows to control joint position and stiffness. In addition, the OT-8 has no onboard power systems. Electric energy is only provided through a tether.

The OT-4 is the next iteration of the OmniTread (Borenstein and Hansen, 2007; Borenstein et al., 2007). It comprises 7 segments. Compared to the OT-8, the OT-4 has smaller size. It has a total length of 940 mm with a diameter of 82 mm. Furthermore, it has a much lighter weight of 4 kg. Besides the size and weight, the OT-4 has other advantages over the OT-8. First, the OT-4 has an onboard miniature air-compressor and batteries. Therefore, there is no need to provide electric and pneumatic energy through a tether. Second, the robot contains micro-clutches that can engage or disengage individual tracks. It would be helpful to save onboard electric power by stopping driving idle tracks. Finally, the OT-4 can carry sensor equipment, such as cameras, microphones, and speakers.Through wireless communication, sensor data can be sent from the robot to the off-board laptop.

2.3.5 Others

Figure 2.7 gives an overview of the remaining famous limbless robots developed in recent years.

A wheel-less snake robot, called Aiko, was developed at the advanced robotics lab at the Norwegian University of Science and Technology (NTNU). The robot has 10 cylindrical links with 20 DOFs. It is 1.5 m long and weighs 7 kg. Aiko is used to support the research on obstacle-aided locomotion (Transeth et al., 2008b; Liljeback et al., 2011, 2012a). The robot is reported capable of generating forward propulsion by pushing against walls or other external obstacles.

As a self-funded project, Dr. Gavin Miller has developed a series of snake robots to investigate snake locomotion (Miller, 2002). Miller's robots are inspired by the work of Hirose. All his robots are equipped with passive wheels to assist in the serpentine locomotion. The most realistic snake robot is the prototype S5. The S5 is made up of a head and 32 actuated links with 64 servos. The robot has 42 onboard batteries and is controlled by one basic stamp II microprocessor, one scenix microprocessor and 8 servo control units.

There are some reconfigurable modular robots that can configure to chain-type. For example, Yim et al. (2001) used PolyBot G1v4 to implement a caterpillar robot that can climb up stairs, ramps, and even vertical porous materials. Another example is M-TRAN. Kamimura built thread configuration with M-TRAN II modules. He applied a CPG model and genetic algorithms to the robot so that it can make caterpillar-like locomotion along uneven terrain (Kamimura et al., 2003, 2005).

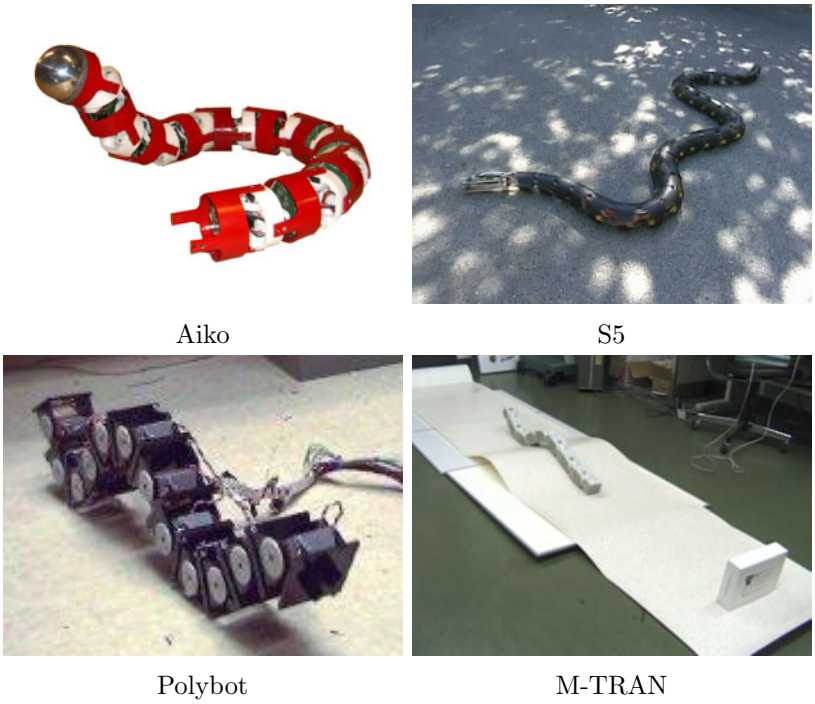


Figure 2.7: Other limbless robots: Aiko, S5, Polybot and M-TRAN.

2.4 Limbless locomotion

From the literature, it is evident that the movement of limbless robots depends on their configurations, as well as auxiliary equipment. On the one hand, the connection between modules determines the motion space. A limbless robot with modules that have parallel joint axes can only effect planar motion. If these modules that compose the robot change into an orthogonal connection, the robot's motion space will extend to 3D. On the other hand, auxiliary equipment also determines the styles of limbless motion. Wheels, treads and even the absence of auxiliary equipment are three possible ways that may help the robot to generate motion. Apparently, different auxiliary equipment leads to different locomotive styles.

To study the movement of limbless robots, they can be classified according to their configurations, or auxiliary equipment. Because the configuration based classification is too general (2D and 3D), it does not help to understand what category of limbless robots I am studying among all the existing robots. Therefore, auxiliary equipment based classification is considered to be a preferred choice. The following divides limbless robots into five categories, including lateral undulation with passive wheels, self-propulsion with active wheels, self-propulsion with active treads, pure body undulation and rectilinear with body contraction and expansion.

2.4.1 Lateral undulation with passive wheels

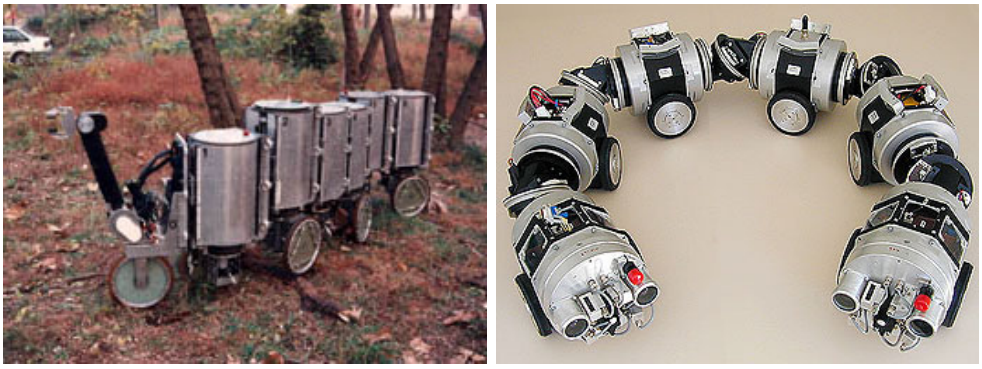
Lateral undulation is the most frequently used form of snake locomotion (Gray, 1946). In this motion, all parts of the snake move simultaneously and bend to the left and right alternately, resulting in a series of sinusoidal-like curves. The body of the snake follows the path the head has traveled during locomotion, which makes the snake looks as if it were sliding along a single narrow winding track. The forward propulsion is a result of all frictional forces generated between the snake's body and the ground. When the snake pushes against multiple contact points simultaneously, a resultant force with lateral frictional force that cancele each other propels the snake forward.

To mimic this kind of motion, limbless robots are usually developed with extra consideration. First, the robot is yaw-yaw connected so as to enable its body to rotate horizontally. Second, the robot is able to perform lateral undulatory gait. The key to lateral undulation is to continuously change the S shape of the robot by coordinating all the modules in special temporal sequence. And third, the robot has passive wheels mounted on the underside of each module, placed in a the tangential direction along the length of the robot. The purpose is to produce asymmetric friction with the ground. The passive wheels can increase frictional force in the lateral direction, while reducing the resistance in the forward direction.

This locomotive style has been verified and implemented on different robot prototypes including ACM III (Hirose, 1993), ACM R1 (Endo et al., 1999), Ma's snake robot (Ma et al., 2001), AmphiBot I and II (Crespi et al., 2005b; Ijspeert and Crespi, 2007) and Gavin Miller's snake robot family. In spite of different control mechanisms that are applied to these prototypes, the behavior of lateral undulation is vividly produced by means of passive wheels. Nevertheless, limbless robots with passive wheels on the bottom have the limitation that they can only be applied on flat and smooth terrains. Some robot prototypes, such as ACM R5 (Yamada et al., 2005) and Wheeko (Liljeback et al., 2012b), break the restriction by switching to the use of universal joints and attaching passive wheels around the body of the robot. Therefore these robots can not only perform lateral undulation, but also achieve a variety of body undulations, such as sidewinding, rotating and rolling.

2.4.2 Self-propulsion with active wheels

Limbless robots are also able to be equipped with active wheels. In contrast to the robots with passive wheels, this kind of self-propelled robots has the main advantage of performing limbless motion with a few modules. Therefore, from the control point of view, it is easier to generate motion by the coordination of fewer degrees of freedom. Powered wheels also provide these robots with higher flexibility in complex environments. Robots do not have to use body undulation but utilize the propulsion of active wheels to traverse obstacles. Taking advantages of the chained shape, these robots are suitable to deal with some types of rough terrain, such as ditches, obstacles, and even narrow and meandering paths.



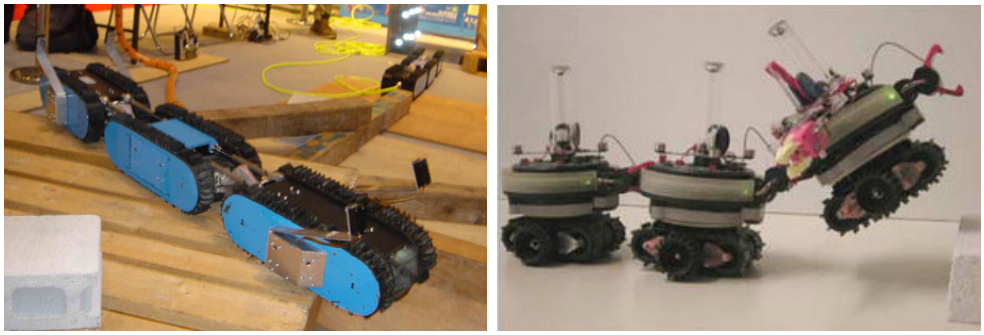
Koryu II Makro

Figure 2.8: Self-propulsion with active wheels.

Figure 2.8 shows two examples of limbless robots with active wheels. One of them is Koryu II (Hirose et al., 1991). Koryu II is developed with high terrain adaptability. It can be used in situations where large body robots are not maneuverable enough to negotiate and small robots cannot transport any needed equipment in a cramped environment. The robot has a train-like appearance. It weighs approximately 320 kg and has a size of 3300 mm in length, 1080 mm in height and 460 mm in width. It consists of a head and 6 cylindrically shaped units. Each unit is attached to an independently powered wheel driven by a dc motor. The unit has three DOF: one in the rotational movement axis (q axis) for the swing to the left and right, one in the perpendicular movement axis (z axis) for sliding up and down and one in the wheel axis (s axis) for the forward movement. The configuration with the powered z axis enables the robot to adapt to rough ground. The robot has been proven capable of autonomous running on stairs, outdoors and in city streets.

The other example is Makro that is developed at the Frauenhofer Institute of artificial intelligence (Rome et al., 1999; Streich and Adria, 2004). The robot has a total length of approximately 1500 mm with a diameter of 160 mm. It has 6 aluminum modules connected by 5 motor-driven active joints. Each module is equipped with a pair of wheels and propelled by an inside motor. Each joint contains 3 motors that allow for rotations along the x, y and z axis, respectively. At the head it contains a stereo camera, a lamp, two laser projectors and four ranging sensors. Makro is developed for providing sewer inspection and maintenance. It is able to navigate inside the sewer system with a diameter of 30 to 60 cm and permits maneuvering around sharp bends up to 90°. Since the robot needs to move backward when it detects blockage, it is designed symmetrically with identical head modules on both ends.

In addition to these two examples, there are some other limbless robots using active wheels, such as GMD-SNAKE2 (Klaassen and Paap, 1999), Genbu (Kimura and Hirose, 2002), ACM R4 (Yamada and Hirose, 2006) and MoMo (Khunnithiwarawat and Maneewarn, 2011).



Soryu IV Swarm-Bot

Figure 2.9: Self-propulsion with active treads.

2.4.3 Self-propulsion with active treads

Limbless robots with active treads are usually developed for search and rescue tasks. Although the robots with powered treads or crawlers are not as energy efficient as those with active wheels, treads are much less likely to cause the robot to get stuck in extremely rough terrain. Due to a higher traction and a larger contact area, they can provide the robot with better mobility, such as rolling over bumps and crossing trenches or breaks. Figure 2.9 illustrates two prototypes of this kind of robots.

The first example is Souryu robots. Actually, a series of Souryu robots has been developed for the purpose of finding survivors trapped in post-disaster environments. The robots are made up of three parts: the front body, the center body and the rear body, each with caterpillar treads, linked by active joints that move in vertical and lateral directions. For Souryu I and II (Takayama and Hirose, 2000, 2003), the front and rear bodies are designed in wedge shape in order to facilitate entry into the rubble. They can carry out pitch and yaw motion symmetrically by coupled drive of two active joints. Furthermore, Souryu I and II only use one motor to generate propulsion and drive the three bodies concurrently via a power transmission axis. For Souryu III (Masayuki et al., 2004), the propulsion mechanism is improved. Each body of the robot is outfitted with an independent driving motor. Therefore, the power transmission axis is eliminated. Souryu IV and V are two improved prototypes (Arai et al., 2008). For Souryu IV, it is equipped with double-sided treads. Each body of the robot contains two motors for driving the two sides of the treads independently. The robot also contains a joint-driving unit and a blade-spring joint for posture change. For Souryu V, tread placement is taken into account. Double-sided treads have space between the two treads where no driving force can be generated. Therefore they are replaced by a mono tread so as to prevent the robot from getting stuck on rubble.

The other example is Swarm-Bot (Mondada et al., 2003, 2004). Swarm-Bot is a self-assembling robot that consists of a number of small identical robots, called s-bot. Each s-bot has a cylindrical body with a diameter of 116 mm. The s-bot is outfitted with double-sided treads. To improve mobility, two additional teethed wheels with

PolyBot M-TRAN

Figure 2.10: Pure body undulation.

slightly larger diameter than the tread height are attached to the outside of the treads. The s-bot can utilize a rigid gripper or a flexible gripper for connecting other s-bots and grasping objects. Besides, the s-bot is also equipped with an omnidirectional camera, accelerometer, encoders, as well as these sensors which are used for measuring distance, light, torque and humidity. Each s-bot weighs 0.66 kg in total and has fully autonomous mobility. The Swarm-Bot is developed for tasks in hard environment such as exploration, navigation and transportation in rough terrain.

There are also some other robots belonging to this category, such as Koryu I (Hirose and Morishima, 1990), MOIRA (Osuka, 2003), OmniTread OT-8 (Granosik et al., 2005) and OT-4 (Borenstein et al., 2007), and JL-I (Zhang et al., 2006).

2.4.4 Pure body undulation

Although auxiliary equipment is helpful for limbless robots to generate locomotion, it constrains the movement patterns of the robots at the same time. This causes these robots not to be versatile enough to deal with different situations. In contrast, limbless robots without any auxiliary equipment have the ability to perform various limbless motions, such as lateral undulation, sidewinding and rolling. This kind of robots generates locomotion by means of pure body undulation. To achieve a locomotive pattern, a robot should keep its body shape changing over time by coordinating all its segments. The purpose is to employ traveling body waves for propulsion. Each segment of the robot is required to follow the motion of its adjacent segment from the rear side. Thus a traveling wave can be generated and propagated along the body of the robot from the tail to the head, resulting in an effective motion.

Some reconfigurable modular robots belong to this category, as shown in Figure 2.10. The PolyBot is an early type of modular robot developed for investigating versatility, reliability and low cost of modular robots (Yim et al., 2000). The design concept of PolyBot is to combine simple structured modules to form a complex system that could achieve complex tasks. PolyBot has two types of modules, called

segments and nodes. Each segment has a cube shape and consists of two connection plates, the actuator and the electronics. The node is a rigid cube with 6 connection plates. It has no degree of freedom but endows the PolyBot with non-serial chain structures. There are mainly three versions of PolyBot: G1, G2 and G3 (Duff et al., 2001). The first generation G1 is a simple prototype made of plastic. It is not self-configurable and uses off board power and computation. The generation G2 is an improved prototype that it adds onboard computing as well as the ability of self-configuring. The most advanced generation G3 adds a brake to the main actuation and features further improvements in the form of more sensors, lower power consumption and less weight and volume. The PolyBot utilizes precomputed gait control tables method to control large numbers of modules (Yim, 1994). Experiments have demonstrated that the PolyBot is very versatile for multiple modes of locomotion.

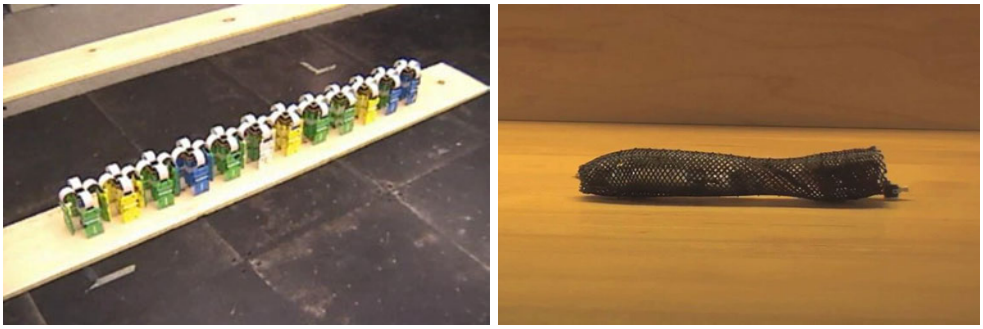
The modular transformer (M-TRAN) is one of the most advanced modular robots as it can change its structrure and motion to adapt to the environment. The M-TRAN module is a kind of double-cube module. It consists of two semi-cylindrical parts and a link part. The link part contains two geared motors which can rotate each of the semi-cylindrical parts from -90 to 90 degrees independently. In addition, each module is equipped with a microprocessor, transmission devices, li-ion battery and connection mechanisms. To achieve more reliable operation, three generations including M-TRAN I, II & III (Murata et al., 2002; Kurokawa et al., 2003, 2008) have been developed, with improvements to downsizing, power consumption and connection mechanism. As for locomotion, an automatic locomotion pattern generation method is used to control the M-TRAN (Kamimura et al., 2005). A decentralized locomotion controller based on a neural oscillator model is employed on the robot for basic gait generation. A genetic algorithms is further utilized on the robot to improve the performance of the locomotion pattern. Through software simulations and hardware experiments, the robot is reported capable of performing stable and effective locomotion by taking advantage of body undulation.

Other noteworthy examples of this category include CONRO (Castano et al., 2000), CMU's snake robots (Johnson et al., 2011) and GZ-I (Zhang et al., 2008). Our study is also related to this category.

2.4.5 Rectilinear with body expansion and contraction

Rectilinear is a special mode of locomotion typically used by large snakes like pythons. In this locomotion, the snake moves in a straight line by using the muscles between the skeleton and the skin. First, the snake lifts a section of its belly slightly up from the ground and anchors one step forward, which leads to the stretching of the belly scales and the corresponding muscles. Then, the snake pulls these muscles back to normal state. Since the snake's belly sticks against the ground, muscular contractions thus propel the snake forward against frictional resistance.

For limbless robots, rectilinear motion can also be obtained. Unlike the robots aforementioned, this kind of robots makes use of body expansion and contraction to



Crystalline Meshworm

Figure 2.11: Rectilinear with body expansion and contraction.

imitate the movement. Figure 2.11 illustrates two examples including the crystalline robot and the Meshworm robot.

The crystalline robot (Rus and Vona, 2000, 2001) is a configurable robot developed at the distributed robotics laboratory of MIT. The robot is composed of a number of shrinkable modules, called atoms. Each atom has a two-dimensional structure with one face on each side. The atom occupies a 2 inch square in the normal state. When expanded, it can extend to twice the length of its original length. The atom weighs about 340 g and contains an on-board processor, batteries and support circuitry. It has two versions. For version 1, the atom ties the 4 faces together and uses a single motor for actuation. This means it has to extend or contract the 4 faces simultaneously. In version 2, the atom is more versatile in that it can control each face independently when extending or contracting. It has been demonstrated that the crystalline robot could be arranged in a form of chain type and achieve the rectilinear motion by extenting and compressing each atom in turn from the tail to the head.

The Meshworm (Seok et al., 2012) is a soft-bodied earthworm robot that can mimic peristaltic movement. The robot has a body structure that looks like a flexible meshlike tube. Wired muscles are wound around the body like belts. An artificial muscle made of a nickel titanium shape memory alloy serves as a soft actuator. When heating up the nickel titanium coils on the muscle in the form of an electrical current, the muscle contracts the corresponding segment radially. Thus the length of the segment expands. Furthermore, longitudinal tendons are attached along the body of the robot to keep the body length constant. Therefore the contraction of a segment will lead to longitudinal contraction of the remaining segments. Besides a straight movement, the robot has steering capabilities implemented by the use of two additional longitudinal muscles. The Meshworm is proved robust enough that even if hit by a hammer or stepped on, the robot can continue crawling after these perturbations.

Within this category, robots using body expansion and contraction also include Slim Slime (Ohno and Hirose, 2001), Telecubes (Suh et al., 2002), and some earthworm robots (Menciassi et al., 2004; Kim et al., 2006).

	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}	α_{11}	α_{12}
Step 1	[10	-10	-10	10	10	-10	-10	10	10	-10	-10	10]
Step 2	[10	-10	-10	10	10	-10	-10	10	90	-60	-90	60]
Step 3	[10	-10	-10	10	10	-10	-10	10	85	-85	-85	85]
Step 4	[10	-10	-10	90	-60	-90	60	10	0	0	-35	35]
Step 5	[10	-10	-10	85	-85	-85	85	10	10	-10	-10	10]
Step 6	[90	-60	-90	60	0	0	-35	35	10	-10	-10	10]
Step 7	[85	-85	-85	85	10	-10	-10	10	10	-10	-10	10]

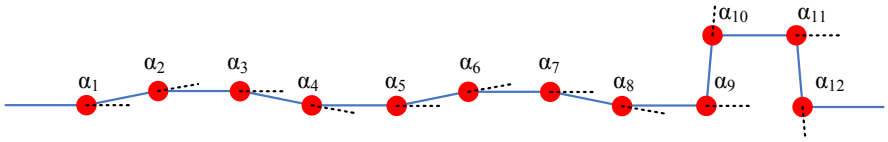


Figure 2.12: Gait control table for the caterpillar gait, adopted from Yim et al. (2001).

2.5 Control methods

So far a number of limbless robots as well as their locomotive styles have been presented. However, how to control such a limbless robot has not been discussed yet. Actually, as limbless robots have redundant design, their cotrol problems lie in the coordination of many degrees of freedom. The coordination is achieved only when the robot combines all the body segments in a manner that is well timed, smooth, and efficient with respect to an intended movement. Therefore, the core issue of coordination is to find proper functions that can determine how the angle of each segment varies over time, so that the robot manages to move. The following presents some control methods that resolve the coordination problem.

2.5.1 Gait control table

The gait control table is a manual method that Mark Yim proposed in his doctoral thesis for generating gaits in modular robots (Yim, 1994). In this method, a gait is represented by a table which contains a sequence of relevant actions running on the robot's joints. The gait table is a matrix. Each column of the table represents a corresponding joint in the robot and contains the discrete position of that joint over time. Each row of the table stands for the body shape of the robot at a certain time instant. Thus a gait table corresponds to a designated gait.

Figure 2.12 shows an example of controlling a limbless robot using a gait control table. The content of the table is described in the form of angle values. At each time step, a controller reads one row of the table and sends the position to the corresponding joints. This leads to the change of the robot's body shape frame by frame. The procedure is continued until the controller reaches the last line of the table. Then the controller again starts to read row by row from the beginning, resulting in a repetitive gait.

It has been shown that the gait control table is an effective way to control

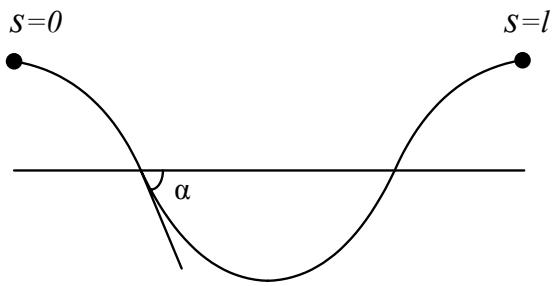


Figure 2.13: Serpenoid curve, adopted from Hirose and Yamada (2009).

limbless robots. Nevertheless, it lacks flexibility. Even a minor change in the configuration of a robot will make the gait table useless. The gait table needs to be recalculated thoroughly before downloading into the robot. Also, this method does not scale well. It would be tedious and complex to create a proper gait table when the number of the robot's joints increases.

2.5.2 Analytical method

As a classic control method, the analytical method uses kinematic or dynamic models of limbless robots to analyze the locomotion pattern and design control strategies. The purpose is to find an expression of the equations of motion for a given robot with known kinematic constraints, which offers a way for gait generation.

This method has been studied by Hirose on the ACM III (Hirose, 1993). Hirose first analyzed the serpentine movement of real snakes. He found the key property in snakes for achieving serpentine locomotion, namely the different friction coefficients in the tangential and the normal directions. He also discovered that the shape of the snake during this kind of motion is a continuous curve which changes the curvature along the length of the body sinusoidally. To approximate the shape of the snake, he proposed a curve with a name serpenoid curve that is different from sinusoidal or even clothoid curves. The serpenoid curve is illustrated in Figure 2.13 and described by the following equations:

$$x(s) = sJ_0(\alpha) + \frac{4l}{\pi}\sum_{m=1}^{\infty}\frac{(-1)^m}{2m}J_{2m}(\alpha)sin(m\pi\frac{s}{l}) \tag{2.1}$$

$$y(s) = \frac{4l}{\pi}\sum_{m=1}^{\infty}(-1)^{m-1}\frac{J_{2m-1}(\alpha)}{2m-1}sin(\frac{2m-1}{2}\pi\frac{s}{l}) \tag{2.2}$$

where x and y are the coordinate point along the curve; s is the distance from the starting point along the curve; l is the length of one period of the curve; J_m is the mth order Bessel function; and α is the winding angle that is defined as the angle of the body with respect to the horizontal axes. The serpenoid curve is proved to be in close agreement with the measured data from real snakes.

Hirose then simplified the ACM III as a skeleton model and went on to study the force and power the robot needed. By analyzing the forces and torques along the

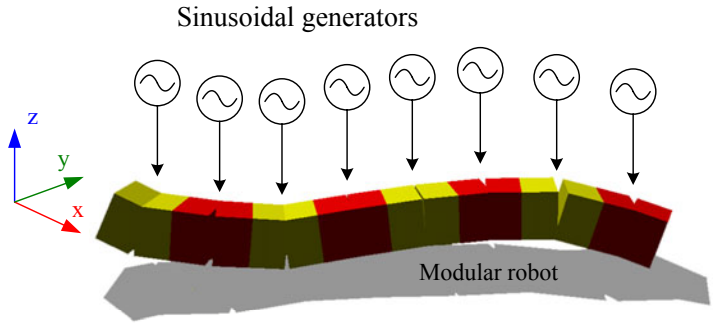


Figure 2.14: Sinusoidal generators, adopted from Gonzalez-Gomez et al. (2007).

body of the robot, he derived the forces in the tangential direction and the normal direction separately. The forces are expressed as density functions with respects to continuous torque and curvature:

$$f_t(s) = \frac{dT(s)}{ds}k(s) \tag{2.3}$$

$$f_n(s) = \frac{d^2T(s)}{ds^2} \tag{2.4}$$

where T is the bending torque; k is the curvature of the serpenoid curve; f is the force per unit of length along the body. These functions show the principle of the serpentine motion. Again, the generated forces are verified very close to the force data measured from snake movenments.

There has been a lot of related work that describes models of the kinematics and dynamics of limbless robots, such as the work by Chirikjian and Burdick (1995), Ma (1999); Ma et al. (2003), Saito et al. (2002), McIsaac and Ostrowski (2003) and Transeth et al. (2008a). However, this method excessively relies on the model of the robot. An inaccurate model may cause a sharp decline in the robot's performance. Moreover, this method is not flexible engough to change the behavior of the robot especially when dealing with a complex environment.

2.5.3 Sine-based method

The sine-based method is another alternative to solve the control problem of limbless robots. This method takes simple sine-based functions as the generator of rhythmic movement. It usually contains explicit parameters that are used for the modulation of frequency, amplitude, phase difference and offset, respectively. To generate an effective gait, all sine-based generators should have a unified amplitude and frequency, as well as a fixed phase difference. In this way, they can oscillate synchronously and produce traveling waves along the body of the robot.

Gonzalez-Gomez et al. (2007) studied the locomotion of 1D pitch-pitch and pitch-yaw modular robots. He took pure sinusoidal generators as the control model of the robot since they are simple to implement and require low computing resources. More importantly, they can be implemented in inexpensive microcontrollers. As

Table 2.1: Parameters of the sinusoidal generators

Symbols	Description	Range
φ_i	Bending angle of the module i	[-90, 90] degrees
A_i	Amplitude of generator i	[0, 90] degrees
T_i	Period of generator i	Time units
ϕ_i	Phase of generator i	(-180, 90] degrees
O_i	Offset of generator i	[-90, 90] degrees
M	Number of modules of the robot	$M >= 2$

shown in Figure 2.14, each sinusoidal generator acts on one corresponding joint on the robot and sets the joint angles as functions of joint number and time:

$$\varphi_i(t) = A_i sin(\frac{2\pi}{T_i}t + \phi_i) + O_i \quad i \in \{1...M\} \tag{2.5}$$

The meanings of the related parameters are listed in Table 2.1. Juan investigated the solution space of these parameters and obtained locomotion principles for the groups of pitch-pitch and pitch-yaw connecting modular robots. Based on sinusoidal generators, five different gaits including 1D sinusoidal, rolling, rotating, turning and sidewinding have been simulated and finally tested on real robots. Furthermore, Juan also studied the minimal configurations for limbless locomotion. He found that with a minimum number of modules the robot is still able to perform locomotion in 1D and 2D.

Tesch et al. (2009) proposed a similar control model that applied to pitch-yaw connected limbless robots. He grouped the pitch and yaw modules and developed some basic gaits called parameterized gaits. All the parameterized gaits are based upon sinusoidal waves propagating in two mutually perpendicular planes: one along the pitch modules and the other along the yaw modules. The parameterized gaits can be described as:

$$\alpha(n,t) = \begin{cases} \beta_{even} + A_{even} sin(\theta), & n = even \\ \beta_{odd} + A_{odd} sin(\theta + \delta), & n = odd \end{cases} \tag{2.6}$$

where A, θ, δ and β are control parameters of amplitude, frequency, phase shift and offset, respectively; *even* represents the joints in the pitch modules while *odd* represents the joints in the yaw modules. Several limbless gaits have been implemented by using this model, such as lateral undulation, linear progression, slithering and rolling.

Although this method features simplicity and the ability of modulation, it has two disadvantages. First, sine-based models cannot produce smooth modulation. This may cause the robot to generate jerky behaviors or even get damaged in motors and gearboxes, especially when the control parameters are abruptly changed.

Second, there is no simple way to integrate a feedback system into sine-based models. Even if this problem can be solved to some extent, the model itself loses simplicity.

2.5.4 CPG-based method

As described in Chapter 1, the CPG-based method is an elegant solution for gait generation. CPG models used for the control of locomotion in robots are found to have interesting properties (Ijspeert, 2008). First, CPG models exhibit limit cycle behavior, which means CPG models can not only produce stable rhythmic patterns, but also can recover from external perturbations. Second, CPG models are well suited for modular robots, since they provide distributed implementation. Third, CPG models feature the ability of smooth modulation, i.e. changing the locomotion with respect to the speed or the direction, with only a few control parameters. Fourth, some CPG models are able to integrate sensory feedback. Thus the resulting controller is strongly coupled with the mechanical robot it controls. Furthermore, sensory feedback integration allows the controller to further search optimal control parameters and develop adaptive locomotion. Fifth, CPG models are ideally suited for hierarchical control. They can provide learning and optimization algorithms on the higher level with a good basis as mentioned above.

Here we take AmphiBot I as an example of this method (Crespi et al., 2005a). A CPG model based on coupled nonlinear oscillators is constructed as the controller of the robot. In this work, the structure of the CPG forms a single chain of oscillators with nearest neighbor connected. Each oscillator is governed by the following differential equations:

$$\tau \dot{v}_i = -\alpha \frac{{x_i}^2 + {v_i}^2 - E_i}{E_i} v_i - x_i + \sum_j (a_{ij} x_j + b_{ij} v_j) \tag{2.7}$$

$$\tau \dot{x}_i = v_i \tag{2.8}$$

where x is the desired angle of the corresponding joint; v is an internal state variable; α is the convergence rate; τ and E are positive constants that control the frequency and amplitude of the oscillator, respectively; a_{ij} and b_{ij} are the coupling variables that determine the phase difference between adjacent oscillators.

Besides these tunable variables, the nonlinear oscillator also exhibits limit cycle behaviors. Figure 2.15 illustrates the phase plot of the nonlinear oscillator with different random starting points. It is observed that the oscillator is always attracted into the limit cycle except the origin point. Indeed the oscillator converges to:

$$\tilde{x}(t) = \sqrt{E} sin(t/\tau + \phi) \tag{2.9}$$

where ϕ is the phase of the oscillator that depends on the starting point. To achieve position control, a PD controller is used to calculate the torques needed for driving the joint to the desired angle. Through simulation and hardware experiments, the CPG model is proven capable of generating travelling waves on the robot during lateral undulatory locomotion on the ground. Furthermore, since the CPG model

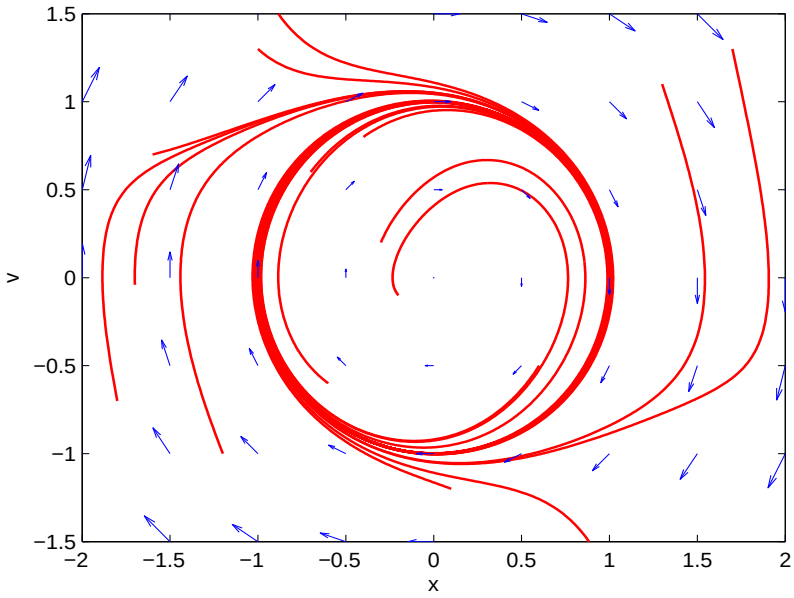


Figure 2.15: Limit cycle behavior, adopted from Crespi et al. (2005a).

provides tunable parameters for online modulation,the optimal frequency, amplitude and wavelength are also identified for producing the fastest locomotion gait.

The CPG-based method has already been successfully applied to control different kinds of robotic systems, not only for limbless configuration, but also for multi-legged robots. However, this method poses challenges that need to be overcome. One is the methodology for designing CPG models. The other one is the integration of feedback system for higher level control. Our study focuses on solving these two problems. The following chapter will discuss more CPG models in detail and present our novel CPG model as well.

2.6 Summary

In this chapter we have seen a variety of limbless robots through the published literature. From a general perspective, we investigate recently created prototypes and look at prior efforts in understanding limbless robots from different aspects, including their advantages, applications, locomotive styles and control methods. The following briefly sums up each aspect of limbless robots.

Since limbless animals can exhibit various highly efficient methods of locomotion to deal with different environments, people have been inspired by their linear structures and duplicated them in mechanisms. Thus, limbless robots inherit the configuration and feature the stability, terrainability and redundancy. That is also why limbless robots are more reliable than traditional wheeled or legged robots in complex environments.

Taking the limbless structure into account, limbless robots have a number of applications. For the task of exploration, inspection and search and rescue, these

robots can cross extreme environments or those areas that are too small for humans to enter. For medical treatment, these robots are beneficial for invasive surgery, which can shorten patients' suffering and healing time. For reconnaissance tasks, these robots are good at hiding their locations and surveils in hostile areas. Of course, limbless robots are not limited to these areas, they can also be applied in other fields, such as education, entertainment and home surveillance.

By surveying existing limbless robots, we see that there are several styles of limbless locomotion. Passive wheels are considered as helpful auxiliary equipment that conduce to generating lateral undulation. Limbless robots can use passive wheels to produce asymmetric friction on the ground and use lateral undulation for propulsion. Active wheels are also helpful for limbless locomotion. The robots with active wheels can produce self-propulsion. This locomotive style involves fewer segments and allows the robots to deal with rough terrains. A similar limbless locomotion is implemented by using active treads. Active treads have a higher traction and a larger contact area compared to powerd wheels. Therefore, limbless robots with active treads have a better mobility and are well suited for search and rescue tasks. Pure body undulation is another way for gait generation. The robots coordinate all the segments they have in proper temporal sequence so as to generate and propagete traveling waves for propulsion. In addition, limbless robots are also able to perform rectilinear motion by body expansion and contraction from the tail to the head.

Since limbless robots have a high number of degrees of freedom, how to coordinate them to form effective gaits is the fundamental issue in control problems. Four control approaches are introduced: a gait control table is a discrete-time position control method that permits to define a sequence of limbless movement "frame by frame"; the analytical method uses known constraints from kinematic or dynamic models to describe limbless motion and design control laws; the sine-based method solves the control problem by employing simple sine-based functions as the controller of individual segments of limbless robots; the CPG-based method takes CPG models as control functions leading to a good solution for gait generation.

Even though limbless robots have been studied in the last few decades, there are still numerous challenges regarding modelling and control issues. How to develop a versatile limbless robot to deal with complicated terrains is one of these challenges. From our review, we find that the CPG-based method is a better way for developing control system on limbless robots. Therefore, we decide to develop a hierarchical control architecture with a CPG model and sensory feedback system well integrated, so as to enable limbless robots to move intelligently through complex environments.

Chapter 3

Design of a Lamprey Spinal Generator

Contents

3.1 Introduction

Early in the 20th century, neuroscientists debated two hypotheses for the generation of rhythmic movements (Marder and Bucher, 2001). One of the hypothesis states that rhythmic motor behavior is produced by chains of reflexes. In such a reflex chain model, sensory neurons are the main trigger to excite interneurons and further activate motor neurons to muscles to generate movements. The other hypothesis states that there are central neural circuits that can generate rhythmic patterns of activity in the spinal cord without any sensory input. The evidence that rhythmic patterns are centrally generated was first demonstrated by an experiment on locust. Indeed, the experiment has shown that the locust nervous system when isolated from the body could still generate fictive flight motor patterns. Thus, the existence of central neural circuits is indisputable any more. These central neural circuits, now called central pattern generators (CPGs), are defined as follows:

CPGs are neural circuits resident in relevant ganglia of invertebrates or the spinal cord of vertebrates, which can produce coordinated patterns of rhythmic activity without any rhythmic inputs from sensory feedback or from higher control centers (Hooper, 2000).

In robotics, based on the level of abstraction from neurobiology, most existing CPG models can be categorized into three levels (Ijspeert, 2008), i.e. the biophysical level, the connectionist level, and the abstract level of coupled oscillators. For most biophysical models, ion channels and membrane potentials are computed to generate rhythmic activities following biological mechanisms, i.e., to simulate the

Hodgkin-Huxley type of neuron model. For connectionist models, rhythmic signals are generated from the interneurons network reactions, i.e. excitatory and inhibitory synapses. Most oscillator models utilize mathematical models, e.g. Van der Pol oscillator (Bay and Hemami, 1987; Collins and Richmond, 1994), phase oscillator (Ijspeert et al., 2007) and Hopf oscillator (Righetti and Ijspeert, 2008; Heliot and Espiau, 2008), in the generation of rhythmic signals, and investigate the population dynamics of networked oscillators. Here we present a few representative CPG models based on different principles and different levels of detail modeling.

3.1.1 Ijspeert's model

Ijspeert and Crespi (2007) developed a CPG model to control the AmphiBot robot. The goal is to demonstrate that the CPG model is suitable for generating online trajectories in a redundant robotic system. The CPG model is based on a system of amplitude-controlled phase oscillators. In the model, neurons are mutually connected to their nearest neighbors, forming a double chain of network, as shown in Figure 3.1. For each neuron, it is simplified and modeled as a phase oscillator, which is described as:

$$\dot{\theta}_i = 2\pi v_i + \sum_j w_{ij} sin(\theta_j - \theta_i - \phi_{ij}) \tag{3.1}$$

$$\ddot{r}_i = a_i(\frac{a_i}{4}(R_i - r_i) - \dot{r}_i) \tag{3.2}$$

$$x_i = r_i(1 + cos(\theta_i)) \tag{3.3}$$

where θ and r are the phase and amplitude of the oscillator; v and R are the intrinsic frequency and amplitude of the oscillator; w is the coupling weight; ϕ is the phase bias between connected neurons; a is a positive constant; x is the output of the oscillator. The first differential equation describes the variation of the phase θ. The last term of the equation indicates the coupling between neurons, which determines the phase lag between neurons. The second differential equation describes the variation of the amplitude r. Due to the second order linear differential equation, r is guaranteed to smoothly converge to R in a dampened fashion.

To generate a travelling wave, the corresponding parameters in all oscillators need to be set to the same values. The purpose is to synchronize these coupled oscillators. Furthermore, the neurons between the left and the right sides of the double chain have a fixed phase lag that is equal to π, which makes them oscillate in anti-phase. Note that the neuron's output x has only positive values, it is not suitable to directly apply the output on a robot's joint. Instead, the desired angle φ corresponding to the joint of the robot is further defined as the difference between the output of the left oscillator and the right oscillator:

$$\varphi_i = x_{i\_left} - x_{i\_right} \tag{3.4}$$

Ijspeert's model belongs to the abstract level. It not only produces stable rhythmic patterns through the limit cycle behavior, but also has explicit control parameters for modulating the CPG model such as frequeny, amplitude and phase lag.

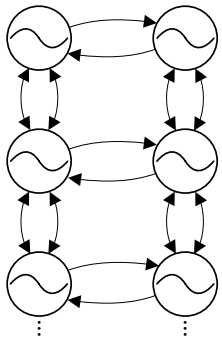


Figure 3.1: Ijspeert's CPG model, adopted from Ijspeert and Crespi (2007).

This kind of CPG models are usually designed artificially by using mathematical properties. However, a systematic way of desining such CPG models with reflex mechanism and sensory feedback inclusion is seldom discussed.

3.1.2 Matsuoka's model

Matsuoka (1985) proposed a sustained oscillatory CPG model using mutually inhibiting neurons. In the CPG model, neurons are represented by a continuous-variable neuron model with a kind of fatigue or adaptation effect. The neuron's activity is simulated as the variation of a continuous variable over time, which is described by the following equations:

$$\begin{cases} \tau \dot{x}_i + x_i = \sum_{j=1}^{n} a_{ij} y_j + s_i - b x'_i \\ T \dot{x}'_i + x'_i = y_i \\ y_i = max(x_i, 0), \quad (i = 1, ...n) \end{cases} \tag{3.5}$$

where x is the membrane potential of the neuron; x' is an internal state variable; τ and T are time constants; a is the coupling weight between neurons; s is the total input from the outside; b is the gain for self inhibition; n is the number of neurons involved; y is the output of the neuron. Matsuoka further analyzed the oscillations generated by mutual inhibition between n neurons, and proved that the model can generate self-sustained oscillations. However, Matsuoka's model has a limited applicability since it can only produce positive values.

Fukuoka and Kimura (2003) extended the CPG model to realize dynamic walking of a quadruped robot on irregular terrains. They combined two mutually inhibiting neurons instead of one as a single neuron, so as to eliminate the disadvantages of Matsuoka's model. Figure 3.2 illustrates the detail connection in one neuron. The

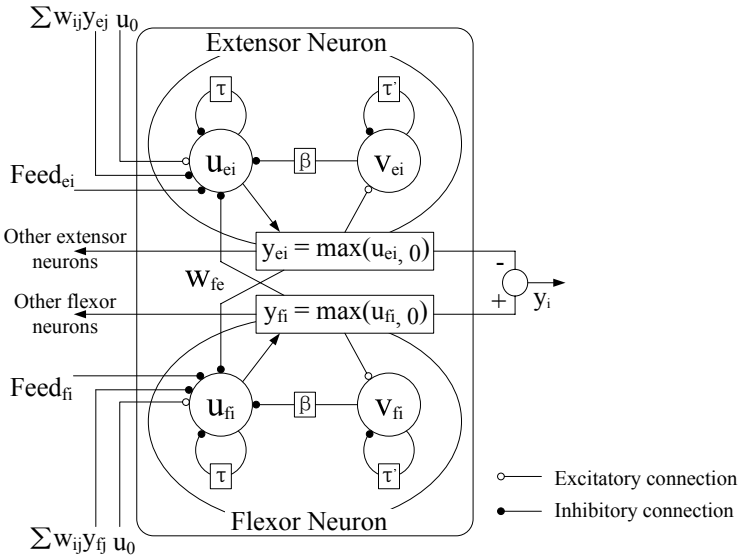


Figure 3.2: Kimura's extended CPG model, adopted from Fukuoka and Kimura (2003).

two inner neurons that correspond to extensor and flexor are described as follows:

$$\begin{cases} \tau \dot{u}_{\{e,f\}i} = -u_{\{e,f\}i} + w_{fe} y_{\{f,e\}i} - \beta v_{\{e,f\}i} \\ \qquad\qquad + u_0 + Feed_{\{e,f\}i} + \sum_{j=1}^{n} w_{ij} y_{\{e,f\}i} \\ \tau' \dot{v}_{\{e,f\}i} = -v_{\{e,f\}i} + y_{\{e,f\}i} \\ y_{\{e,f\}i} = max(u_{\{e,f\}i}, 0) \\ y_i = -y_{ei} + y_{fi} \end{cases} \tag{3.6}$$

where u and v represents the firing rate and fatigue of neurons, respectively; τ and τ' are time constants; w is the connection weight between neurons; β is the coefficient for the effect of the fatigue; u_0 is a constant that represents the tonic input to the neuron; $Feed$ is a feedback signal of joint angle from the robot; y is the output of a neuron.

There are many tunable parameters in the extended CPG model. As discussed in Matsuoka (1987), Williamson (1998) and Bailey (2004), the parameters u_0, τ, τ', β and w_{fe} dictate the dynamic of the model. u_0 adjusts the amplitude of oscillaton. Changing τ or τ' alters the shape of oscillation and affects the frequency of oscillation. Only if the ratio between τ and τ' is constant, τ can change the frequency in a linear fashion. Changing β or w_{fe} modifies not only the amplitude but also the frequency of oscillation. Due to the coupling effect on oscillation when tuning these parameters, it is difficult to modulate the model in an independent way. Therefore, to avoid complicating the design process, τ', β and w_{fe} are usually fixed, and only u_0 and τ are used for modulation.

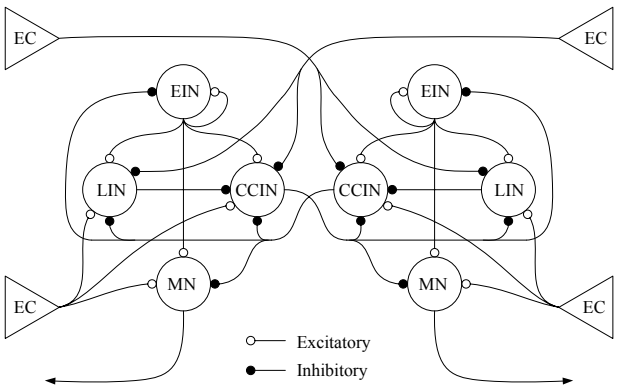


Figure 3.3: Ekeberg's CPG model, adopted from Ekeberg (1993).

3.1.3 Ekeberg's model

Ekeberg (1993) reduced the lamprey's CPG network to a simplified connectionist model. The designed CPG network is a non-spiking model, since the behavior of the CPG network is mainly dependent on the pattern of synaptic connectivity, rather than neuronal properties such as action potentials. In the CPG model, each segment is represented as a population of functionally similar neurons. As shown in Figure 3.3, five types of interneurons are mutually connected at the segmental level. For each type of interneurons, it acts primarily as a leaky integrator, where delayed excitatory and inhibitory inputs are separately integrated. The output of an interneuron is calculated according to the following formula:

$$\dot{\xi}_+ = \frac{1}{\tau_D}(\sum_{i\in\Psi_+} u_i w_i - \xi_+) \tag{3.7}$$

$$\dot{\xi}_- = \frac{1}{\tau_D}(\sum_{i\in\Psi_-} u_i w_i - \xi_-) \tag{3.8}$$

$$\dot{\vartheta} = \frac{1}{\tau_A}(u - \vartheta) \tag{3.9}$$

$$u = \begin{cases} 1 - exp\{(\Theta - \xi_+)\Gamma\} - \xi_- - \mu\vartheta & if \quad u > 0 \\ 0 & otherwise \end{cases} \tag{3.10}$$

where ξ_+ and ξ_- are delayed value for excitatory and inhibitory synaptic inputs, respectively; τ_D and τ_A are time constants; Ψ_+ and Ψ_- represent the incoming excitatory and inhibitory synapses from other neurons, respectively; u is the output of the neuron; w is the synaptic weight; ϑ is the neuron's internal state variable; Θ, Γ and μ are parameters that control the threshold, gain and adaptation rate of the neuron's output.

Besides the segmental connectivity, there are also synapses between neighboring segments. Neurons in one segment can distribute synaptic output to nearby segments both caudally and rostrally. Ekeberg used 100 replicas of such segments with

a complicate connection, and succeeded to simulate swimming behaviors by manually selecting synaptic weights. To improve the performance of the CPG model, genetic algorithms were used to optimize the the connection weights and the parameters of neurons (Ijspeert et al., 1997; Patel et al., 2006b). The evolved models were reported more efficient and can reach a wider range of frequencies.

Ekeberg's model is realistic to some extent. It provides a feasible way to simplify biological models for robotic applications and offers the potential for feedback integration. However, the model has a high computational complexity. Every neuron in the model needs to use several one order differential equations to describe the activity of the neuron. Furthermore, the model is also structurally complicated and difficult to analyze numerically (Wu and Ma, 2010).

3.1.4 Herrero-Carrón's model

Herrero-Carrón et al. (2011) developed a CPG model for modular robots based on recently revealed strategies that living CPGs follow. The CPG model consists of neurons and synapses. For neurons, a Rulkov's model is utilized to mimic the activity of real multiple time-scale neurons. It describes the dynamic of ion conductances and membrane potential in neurons. The mathematical description is as follows:

$$f(x,y) = \begin{cases} \frac{\alpha}{1-x} + y & if \quad x \leq 0 \\ \alpha + y & if \quad 0 \leq x < \alpha + y \\ -1 & otherwise \end{cases} \tag{3.11}$$

$$x_{n+1} = f(x_n, y_n); \tag{3.12}$$

$$y_{n+1} = y_n - \mu(x_n + 1) + \mu\sigma + \mu I_n; \tag{3.13}$$

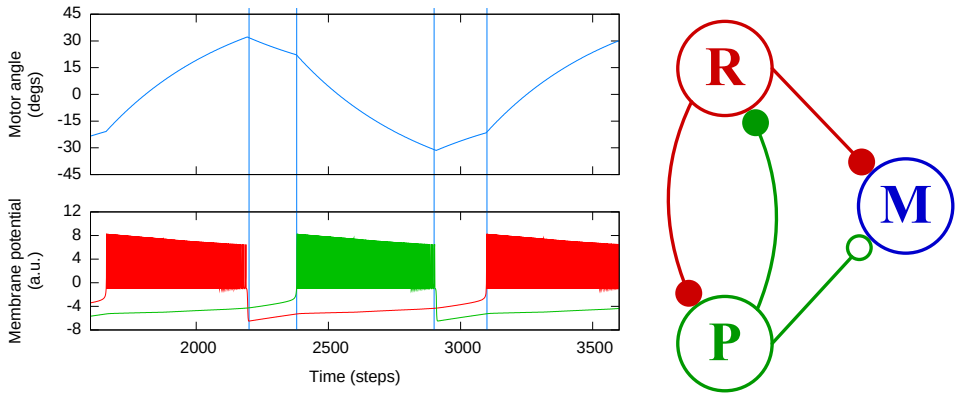
where x_n represents a neuron's membrane voltage; y_n is an internal state variable; μ is a constant; α and σ are variables that control burst duration and period; I_n is the the current flowing from other neurons. The bottom of Figure 3.4(a) shows that neurons modeled with rich-dynamic iterated map can generate rhythmic activity in the form of bursts of spikes.

For synapses, a Destexhe's model is used as the communication channel of neurons. When a presynaptic neuron begins to fire and its membrane potential crosses a given threshold value, the synapse starts to release neurotransmitter to the presynaptic neuron over a certain amount of time. The ratio of bound chemical receptors r and synaptic current I are defined as:

$$\dot{r} = \begin{cases} \lambda[T](1-r) - \beta r & if \quad t_f < t < t_f + t_r \\ -\beta r & otherwise \end{cases} \tag{3.14}$$

$$I(t) = g \cdot r(t) \cdot (X_{post}(t) - E_{syn}) \tag{3.15}$$

where λ, β and g are synaptic constants; $[T]$ is neurotransmitter concentration; t_f and t_r are firing time and lasting time for the presynaptic neuron; X_{post} and E_{syn} are the membrane potential and reversal potential for the postsynaptic neuron.



(a) Neural activity (b) Neural organization

Figure 3.4: Herrero-Carrón's CPG model, adopted from Herrero-Carrón et al. (2011).

To generate an regular oscillatory activity on an isolated module, two neurons (R and P) are mutually connected with inhibitory synapses, as shown in Figure 3.4(b). The two neurons are further connected to a motoneuron (M) where its output signal are directly sent to the servo controller. The following equations describs the synapses connected to the motoneuron s and the output of the motomeuron m:

$$s(x, v) = \begin{cases} 1 & if \quad x > v \\ 0 & otherwise \end{cases} \tag{3.16}$$

$$\tau \dot{m} = -m(t) + \gamma s(x_P(t), v) - \gamma s(x_R(t), v) \tag{3.17}$$

where v is the threshold that detects spikes of neurons; τ is a time constant that controls the chage rate of m; γ determines the maximum amplitude. Since mutual inhibition results in the two neurons R and P firing in anti-phase, the output of the motoneuron M oscillates between $-\gamma$ and γ, as shown at the top of Figure 3.4(a).

Herrero-Carrón's model has rich intrinsic dynamic for neurons and synapses, and retains some characteristic of living CPGs. However, the complexity of the model makes it unsuitable for robotic applications. Moreover, the model has a poor ability for modulation, which also offsets its effectiveness.

3.1.5 Comparison

In addition to the four CPG models introduced above, there are still many other CPG models that are used for robotic applications, such as cyclic inhibitory CPG model (Lu et al., 2005), CNN-based CPG model (Arena et al., 2005a) and hybrid CPG model (Shan and Nagashima, 2002). From these existing CPG models, we sum up the advantages and disadvantages of CPG models in three different aspects that we are particularly interested in, including the computational complexity, the ability of linear modulation and the ability of sensory integration. Table 3.1 illustrates the comparison between the three levels of CPG models. For the biophysical

Table 3.1: Comparison between three levels of existing CPG models

Category	Computational complexity	Parameters for modulation	Sensory integration	Example
Biophysical level	High	Coupled	Simple	Herrero-Carrón's model
Connectionist level	Middle	Coupled	Simple	Kimura's model Ekeberg's model
Abstract level	Low	Explicit	Complicate	Ijspeert's model

CPG models, they retain most properties of living CPGs and have rich dynamic of neural activities. The disadvantages of these models are their high computational complexity and the poor ability for modulation. For the CPG models at the connectionist level, they benefit from biological findings and possible to add reflex or sensory feedback in a natural way. Nevertheless, these models are usually lack of uncoupled control parameters over relevant characteristics. For the CPG models belonging to the abstract level, they feature low computational complexity and explicit parameter modulation, but suffer the inconvenience of sensory integration.

From the comparison, we find that there is no existing CPG model that combines the advatanges of each level of CPG models, namely not only to have low computational complexity and explicit control parameters, but also provide a simple way for sensory integration. Actually, the three aspects for comparison are correlated to some extent. For example, the simpler the CPG model is, the lower computational complexity and the more explicit control parameters it has. But the simpler CPG model also increases the burden on designing sensory feedback mechanism. To develop a more powerful CPG model, one needs to consider which level of CPG model is suitable for the application. At the same time, one needs to take into account the trade-off between the three aspects as well, so as to increase the availability of the CPG model.

3.2 Design goals

In our hierachical control architecture, the CPG model is the kernel of the fine-scale level. As illustrated in Figure 1.2, the newly designed CPG model should meet the following conditions. First, the CPG model should have the ability to generate several kind of gaits, so that the limbless robot can switch to an appropriate gait in different environments. This is the key function of the CPG model. The CPG model is therefore required to offer rich dynamic properties for various gaits generation. Second, the CPG model should provide the interactive interface to the reflex level. This means the CPG model needs to deal with sensory reflex signals and makes proper response to the external stimuli. Third, the CPG model

should be able to receive control signals from the large-scale level by using a set of tunable parameters. If the CPG model has a low control parameter space and provides explicit and uncoupled parameters for online modulation, it is much easier to implement learning algorithms in the large-scale level without considering the detailed control of the CPG model. Fourth, the CPG model should be able to integrate sensory information from onboard sensors or from the large-scale level. Thus the CPG model can be shaped by afferent sensory information and help the limbless robot to achieve adaptive locomotion on uneven terrains.

Considering the functional requirement of the CPG model, we decide to design a new CPG model at the connectionist level. The reason is threefold. First, since the connectionist model is usually derived from simplified biological prototype, it retains the rich dynamic of neural activities. Second, the connectionist model is a compromise between the biophysical model and the abstract model. On the one hand, it has a lower computational complexity than the biophysical model. On the other hand, compared to the abstract model, it has an easier way to add reflex mechanism or sensory feedback originating from biological studies. Third, the connectionist model is also possible to obtain simple and uncoupled control parameters by simply modifying the topology of the neural circuits and redescribing the dynamic of internal neurons, as long as the modification does not offset the above-mentioned advantages.

3.3 Single oscillator design

In biology, although the underlying mechanisms of the CPG circuits are not yet fully understood, several simple creatures, such as the lamprey, have been extensively studied to learn the neural circuits in the spinal cord (Buchanan and Grillner, 1987; Buchanan, 1992; Matthews, 2001). In this section, biologically inspired from the neuronal circuit diagram in the spinal cord of swimming lampreys, we propose a novel CPG model for rhythmic signal generation. The CPG model features the following characteristics:

- The CPG model is based on the spinal cord of lampreys. It has rich dynamic of osillatory activities. Besides the normal osillatory activity, it also exhibits other oscillatory phenomena, such as synchronisation and maintenance activities (see Section 3.5).
- It provides explicit control parameters for output modulation, including the modulation of amplitude, period, phase difference and offset (see Section 3.4).
- It is able to integrate sensory reflex mechanisms. By employing the concept of reflex arc in physiology, the model can make quickly response to external stimuli (see Chapter 5).
- It allows sensory feedback integration. Sensory information can be directly transmitted into the model through additional sensory neurons and further shapes the output of the model (see Chapter 6).

Here we first introduce the design of single oscillator. As shown in Figure 3.5, the CPG circuit consists of two parts: one motor control center and a set of oscillators. The motor control center behaves as the brainstem of the CPG circuit, and descends motor commands through command neurons to modulate the output signal of the oscillators. The oscillator is responsible for rhythmic output generation. One oscillator is composed of two symmetric parts: the left part and the right part. To generate rhythmic activity, four types of interneurons, including Crossed InterNeurons (CIN), Lateral InterNeurons (LIN), Excitatory InterNeurons (EIN) and MotoNeurons (MN), are synaptically connected.

Internal coupling synapses are emitted and received among the interneurons in one oscillator: each CIN emits inhibitory synapses to all the other interneurons except the EIN at the contra-lateral side; each EIN emits excitatory synapses to all the other three types of interneurons on the same side; and each LIN sends an inhibitory synapse to the CIN on the same side. The two MNs from both sides are combined together after signal filtering, which finally generate the output signal of the oscillator.

Besides internal coupling synapses in one oscillator, there are coupling synapses between oscillators. One oscillator emits inhibitory synapses to other oscillators through the EINs. Also, it uses the LINs to receive inhibitory synapses from other oscillators. In most cases, the presynaptic interneuron in one oscillator and the postsynaptic interneuron in another oscillator are both on ipsilateral side, i.e., both belonging to the left side or to right side. Furthermore, the connection between oscillators are symmetric. That means if one oscillator connects to another oscillator, there are two coupling synapses between them: one is on the left side and the other is on the right side.

In the oscillator model, a set of leaky integrator type interneurons is incorporated into the design of the neural diagram. A leaky integrator is often utilized to model biological CPG due to its simplicity and facility for use. As a manner of one order differential equation, the leaky integrator describes the behavior of neuron that gradually leaks a small amount of neurotransmitters over time.

$$\tau \dot{x}_i = -x_i + \sum w_{ij} x_j \tag{3.18}$$

As seen in equation 3.18, the output of leaky integrator x represents the membrane potential of a neuron, and the second term of the equation stands for the neuron's short-term average firing frequency. Similar to the output of a neuron, the leaky integrator's output is monotone ascending or descending and will reach a steady state.

Based on the leaky integrator, the dynamics in the oscillator can be described by the following equations:

$$\tau \dot{x}_{\{CIN\}i} = -x_{\{CIN\}i} + \sum \omega_s s_{\{CIN\}i} \tag{3.19}$$

$$\begin{aligned} \tau \dot{x}_{\{LIN\}i} = &-x_{\{LIN\}i} + \sum \omega_s s_{\{LIN\}i} \\ &+ \sum \omega_c c_{\{LIN\}j} \end{aligned} \tag{3.20}$$

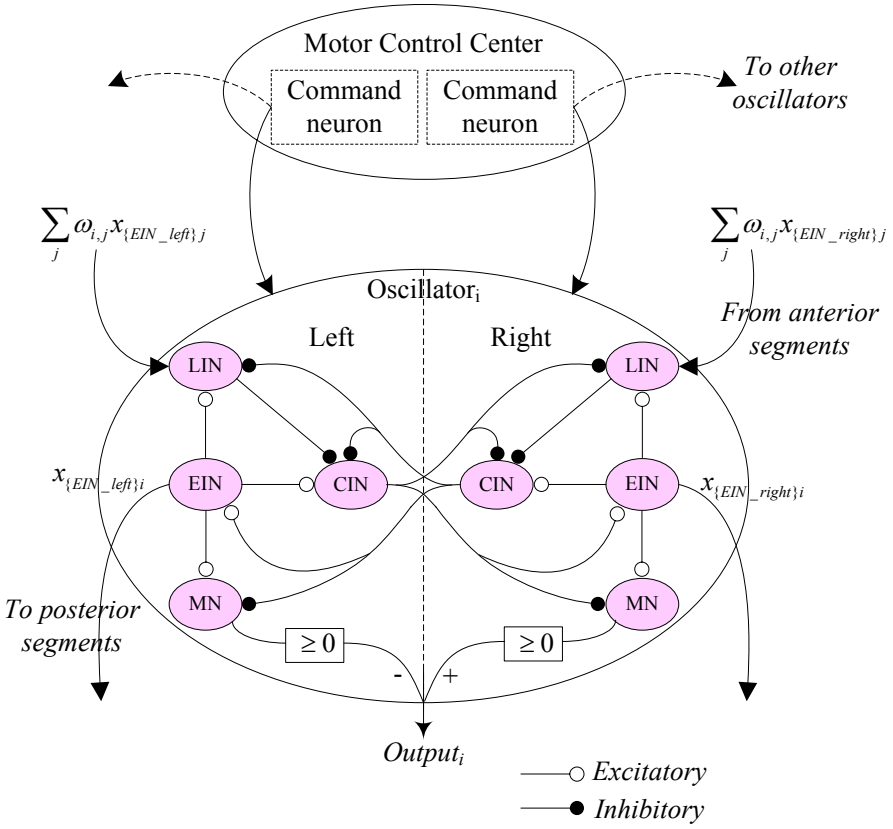


Figure 3.5: The oscillator model. The circuit consists of the motor control center and the oscillator. The role of the motor control center is neuromodulation. It transfers motor commands from the large-scale level and adjusts related control parameters for desired modulation. The role of the oscillator is producing rhythmic activity through the interaction between these interneurons.

$$\tau \dot{x}_{\{MN\}i} = -x_{\{MN\}i} + \sum \omega_s s_{\{MN\}i} + \beta A \tag{3.21}$$

$$x_{\{EIN\}i} = \frac{A}{1 + e^{x_{\{\overline{CIN}\}i}}} - \frac{1}{2}A \tag{3.22}$$

$$output_i = max(x_{\{MN\}i}, 0) - max(x_{\{\overline{MN}\}i}, 0) \tag{3.23}$$

where x_i is the state of each interneuron; the subscript of x represents the types of the corresponding interneuron; the overline on the subscript of x represents the state of the interneuron on the opposite side of the same oscillator; the parameters τ, A and β are tunable parameters that modulate the oscillator's period, amplitude and offset, respectively; ω represents the synaptic weight parameter, where ω is positive for excitatory synapses, and negative for inhibitory synapses; s_i is the presynaptic neuron in the same oscillator; s_i together with the synaptic weight ω_s represents the amount of neurotransmitter transmitted to the postsynaptic neuron; c_j is the presynaptic neuron in other oscillators; c_j together with the synaptic weight ω_c represents the amount of neurotransmitter transimitted to the postsynaptic neuron from other oscillators; the variable $output_i$ stands for the oscillator's output.

Note that the dynamics of the EIN type of interneurons is different from other types of interneurons. They are described by sigmoid function instead of leaky integrator. The sigmoid function is indeed another key element for oscillatory generation. Since the EINs activate the other types of interneurons on the same side, the sigmoid function helps these interneurons to switch internal state alternately. Meanwhile, the leaky integrator helps them to achieve the desired state monotonically. Thus, the sigmoid function together with the leaky integrator forms the self-sustaining mechanism.

As for the computational complexity, for each interneuron, it only needs a first-order diffential equation or a sigmoidal equation to describe a neural behavior. In other words, our CPG model requires 9 equations to calculate the output of one oscillator. Although Ekeberg's CPG model is also derived from lamprey's neuronal circuit in the spinal cord, its interneuron is represented by 4 equations and the output of one oscillator therefore needs 32 equations in total. Compared to that, our model saves a lot computing resources. Actually, compared to other famous CPG models, such as Ijspeert's model (needs 7 equations for an oscillatory output), Kimura's model (needs 7 equations) and Herrero-Carrón's model (needs 12 equations), our CPG model has an average level of computational complexity.

Synaptic weights in the oscillator are fixed and form a symmetric matrix, as listed in Table 3.2. To create a rhythmic and smooth output, all the synaptic weight parameters ω in the oscillator have the absolute value of 1.0, except for the synapse from EIN to MN, whose weight parameter's value is 0.1. The purpose for the decrease of the synaptic weight from EIN to MN is to guarantee the oscillation in the range of ± 90 degrees.

Besides the synaptic weights, the initial values of these interneurons also play important roles in the start of rhythmic oscillation. It has been found that a slight initial asymmetry between the two CINs on both sides can give rise to self-sustained oscillations. Table 3.3 shows a feasible set of initial values for the interneurons

Table 3.2: Synapse weights in the oscillator model

Presynaptic neuron	Postsynaptic neuron	Type	Value
EIN	CIN	Excitatory	1
EIN	LIN	Excitatory	1
EIN	MN	Excitatory	0.1
CIN	CIN	Inhibitory	-1
CIN	EIN	Excitatory	1
CIN	LIN	Inhibitory	-1
CIN	MN	Inhibitory	-1
LIN	CIN	Inhibitory	-1

Table 3.3: Initial values in the oscillator

Interneurons	Value	
	Left side	Right side
$x_{\{CIN\}}$	0.01	0.010001
$x_{\{EIN\}}$	0	0
$x_{\{LIN\}}$	0	0
$x_{\{MN\}}$	0	0

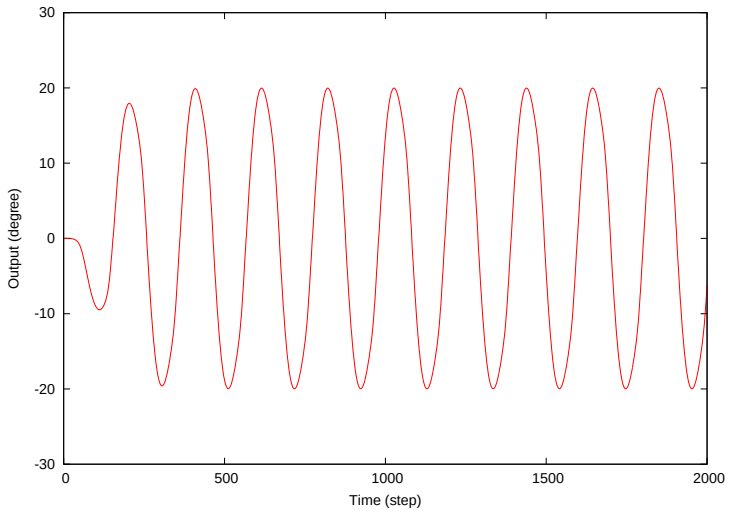


Figure 3.6: The output of a single oscillator. The oscillator succeeds to oscillate by using the synaptic weights in Table 3.2 and the initial values in Table 3.3, with parameters $\tau = 0.2$, $A = 20$ and $\beta = 0$.

in the oscillator. By using these initial values, the oscillator is able to achieve rhythmic pattern. Figure 3.6 illustrates the corresponding oscillatory behavior from the beginning to a steady oscillatory state.

3.4 Chained inhibitory CPG circuit

Besed on the single oscillator design, CPG circuits can be further constructed by means of the connection between oscillators. Oscillators in CPG circuits are no longer isolated, but interconnected with one another. Therefore, to some extent, the connectivity among oscillators determines the behaviors of CPG circuits. This section investigates a chained inhibitory CPG circuit, as well as its control parameters for output modulation.

Before the introduction, in order to make the appearance of CPG circuit more concise, interneurons and synapses in the oscillator are simplified, as shown in the right side of Figure 3.7. In the simplified version of oscillator, 'L' and 'R' represent four interneurons together with their synaptic connection on the left and right part, respectively. The dashed lines between the 'L' and the 'R' in the oscillator indicate the mutually connection between the left and right part. The rest of the thesis uses this simplified version of oscillator to replace the standard one in all the constructed CPG circuits.

3.4.1 CPG circuit construction

As discussed in the last section, an oscillator has the ability to establish connection with other oscillators through the EIN and LIN interneurons. Two identical oscillators therefore can be connected by inhibitory synapses from a pair of EINs interneurons in one oscillator to a pair of LINs interneurons in a second oscillator, as shown in Figure 3.7. If oscillators are connected one afer another in this way, a chained ininhibitory CPG circuit can be constructed.

Figure 3.8 shows the construction of the chained inhibitory CPG circuit. For each oscillator, unidirectional inhibitory synapses are emitted to its adjacent oscillator. The inhibitory synaptic weights ω_c between oscillators are assigned to the same value of -1. The purpose is to maintain a fixed phase difference between these oscillators.

In addtion to the normal oscillators, there is a special oscillator in the circuit. A command oscillator belonging to the motor control center emits inhibitory synapses to the first oscillator of the chained topology. Two additional synapses, one for excitatory synapse and the other for inhibitory synapse, are self projected to the command oscillator, which are used to modulate the phase difference among these oscillators. It has been tested that if the two synapses have relative small values of synaptic weights, such as in the range of (0, 1) and (-1, 0), respectively, they can shift the phase difference sensitively. Moreover, if the two synaptic weights are interrelated, such as by artificially setting the sum of their absolute values to a

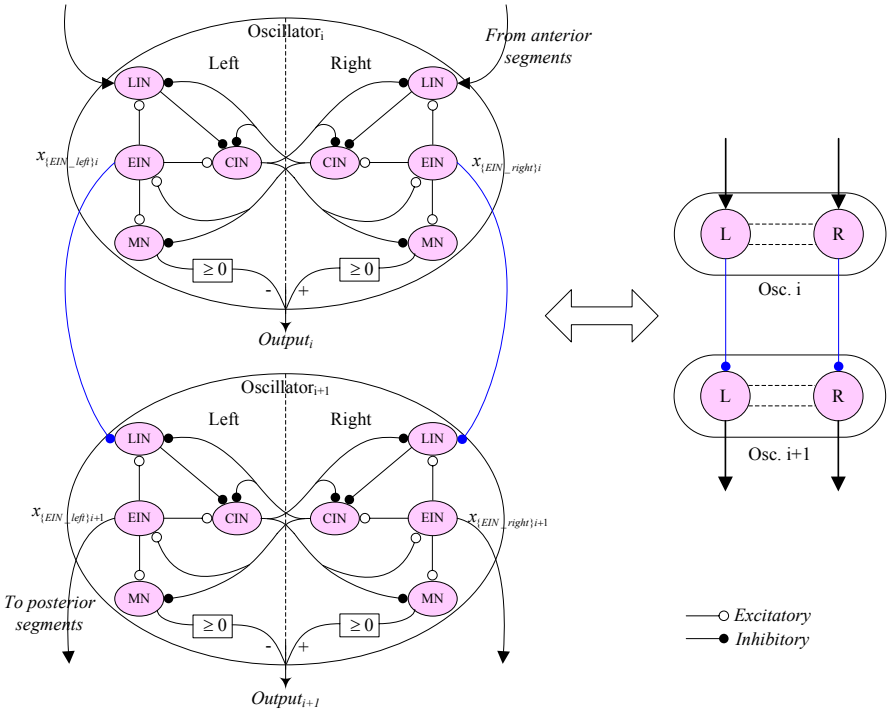


Figure 3.7: The unidirectional connection between two oscillators and their simplification. Both the EINs in the first oscillator emit inhibitory synapses to the ipsilateral LINs in the second oscillator. The simplified version of oscillators uses the 'L' and the 'R' to represent the corresponding interneurons and connection.

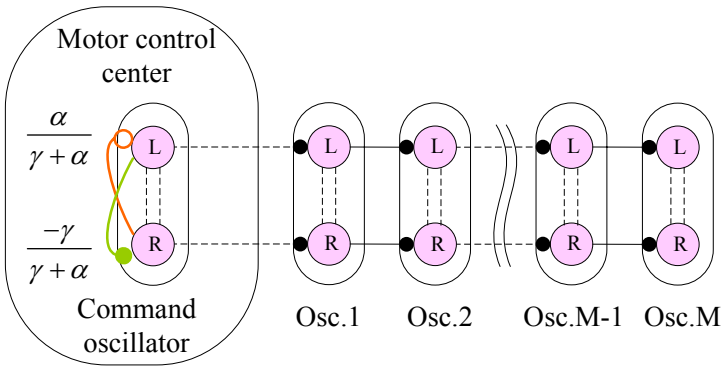


Figure 3.8: The chained topology of oscillators. When oscillators are unidirectional connected with the same inhibitory synaptic weights, a phase difference between them develops. By altering the weight of additional synapses of the command oscillator, the other oscillators can be modulated to maintain the desired phase difference. Parameters α and γ, with a range of (0, 1], are used to modulate the phase difference.

constant, the variation of phase difference becomes monotonous and smooth with respect to the two synaptic weights.

Based on the above investigation, the two synapses are defined as follows:

$$\omega_{excitatory} = \frac{\alpha}{\gamma + \alpha} \tag{3.24}$$

$$\omega_{inhibitory} = \frac{-\gamma}{\gamma + \alpha} \tag{3.25}$$

where $\omega_{excitatory}$ and $\omega_{inhibitory}$ are the weights of the two synapses; α and γ, with a range of [0, 1], are control parameters that play a role in phase difference modulation.

To find out how α and γ affect the phase difference, other control parameters in equations 3.19–3.23 namely parameters τ, A and β, are fixed with a value of 0.4, 20 and 0, respectively. Figure 3.9 illustrates how the phase difference between oscillators varies with respect to parameters α and γ. The result is a monotone smooth curved surface. It is noted that the phase difference decreases with the growth of α, while it increases with the growth of γ. Both of them can monotonously modulate the phase difference.

Since parameters α and γ have a similar functionality on phase difference modulation, there is no need to take both of them as phase difference control parameters. In this case, γ is considered as a constant rather than a variable, while α is considered as the only control parameter for phase difference modulation. γ is set to a value of 0.2 throughout the thesis. Of course, it is possible to set γ to any value as long as they lie in the range of (0,1]. However, considering that a wider range of phase difference would be more useful in developing limbless locomotion, a small value of γ is preferred according to the variation of phase difference in Figure 3.9. Figure 3.10 shows that when γ is equal to 0.2, the phase difference decreases with the growth of α, and the available phase difference lies in the range $[45°, 145°]$.

3.4.2 Parameters adjustment

Numerical simulations are performed to study the output of the chained inhibitory CPG circuit. Dozens of oscillators are employed in the simulation. For simplicity, only the output of the first five oscillators is shown when explaining the effect of parameter adjustment.

As introduced before, there is a number of tunable paramters, namely A, τ, α and β, that determine the dynamics of the circuit. Each of them takes charge of changing the shape of the output between oscillators in a different way. To quantitatively analyze the impact of each control parameter on the output, in each simulation, only one parameter is tuned within an acceptable range while the other parameters are fixed. Taking advantage of this method, it is simple to examine whether a control parameter has coupling impacts on the circuit's output. The following introduces each of these control parameters and shows examples of their effects on the output.

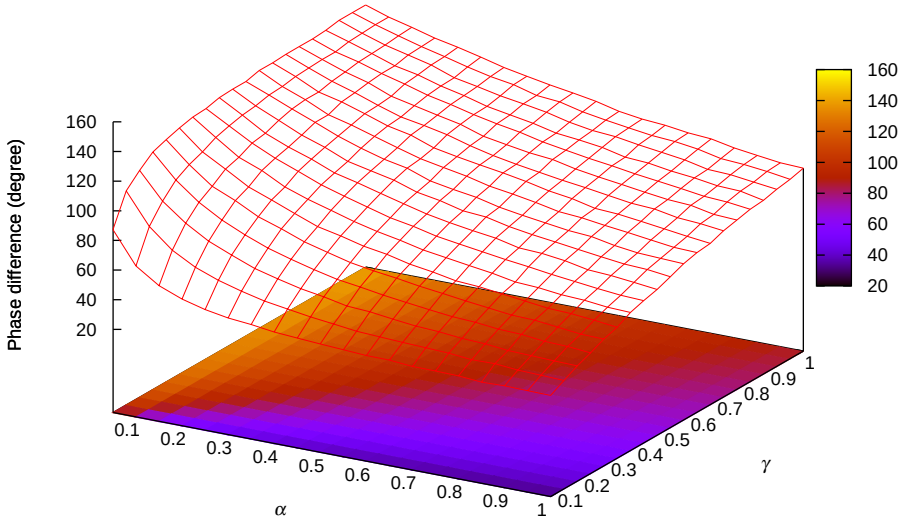


Figure 3.9: The variation of phase difference between oscillators with respect to parameters α and γ, where other control parameters are fixed as $\tau = 0.4$, $A = 20$ and $\beta = 0$. α and γ have the same functionality that both of them can shift the phase difference between oscillators monotonously.

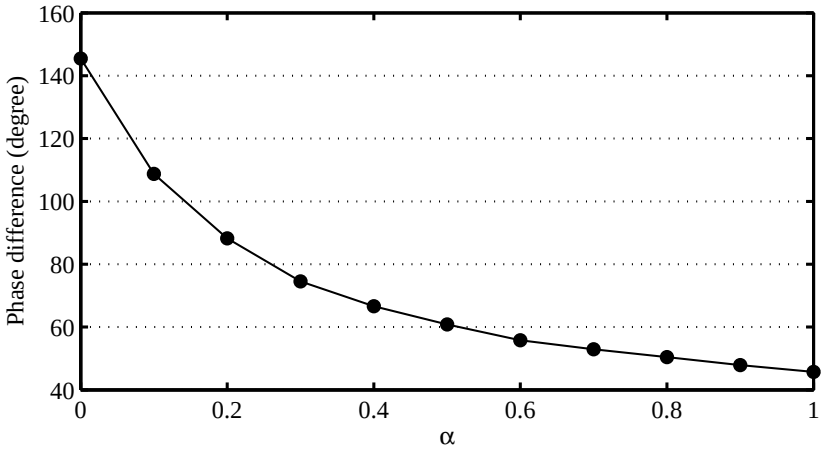


Figure 3.10: Phase difference modulation with respect to parameter α, where $\gamma = 0.2$. The phase difference can be modulated in the range from 45° to 145°.

Changing A

A is the control parameter that is pertinent to the amplitude of the oscillator. To examine how it impacts on the amplitude, τ, α and β are set to 0.4, 0.18 and 0, respectively. From the numerical results in Figure 3.11(a), it has been found that A and the ampitude of the oscillator are linearly dependent. They have a one-to-one mapping with a coefficient of 1. That is to say, A is a direct representation of the amplitude. A is restricted in the scope of (0, 90]. From Figure 3.11(a), it is also noted that modifying the amplitude of the oscillator by changing A would not affect the oscillators' period, as well as the phase difference.

Figure 3.15(a) is a specific example of amplitude modulation. At the beginning, A, τ, α and β are set to 60, 0.4, 0.7 and 0, respectively. Altering A from 60 to 30 at the 4000th time step results in the shrink of the amplitude by half.

Changing τ

τ is a time variable that controls the period of oscillation. Figure 3.11(b) shows the variation of oscillator's amplitude, period and phase difference with respect to τ, where $A = 40$, $\alpha = 0.18$ and $\beta = 0$. The result reveals that τ is linearly proportional to the oscillation period. The larger value of τ the oscillator has, the larger period it will get. Furthermore, the result also verifies that τ only affects the oscillators' period without influencing their amplitude and phase difference.

τ is designed in the range of [0.2, 0.8]. On the one hand, although τ can modulate the period more sensitively if it is less than 0.2, it loses the linear relationship with the period. On the other hand, too large oscillation period resulted from too large value of τ is meaningless for real applications. Therefore, τ is set empirically to a maximum value of 0.8. From the simulation, τ is tested capable of altering the oscillation peroid from 136 to 534 time steps. This means the period can be modulated as much as 4 times in the available range of τ. The period can be approximately described as a function of τ:

$$period \approx 670 \cdot \tau \tag{3.26}$$

An example of period modulation is shown in Figure 3.15(b), where A, τ, α and β are set to 60, 0.3, 0.7 and 0, respectively. Parameter adjustment occurs at the 4000th time step, where τ changes its value from 0.3 to 0.6. Thereafter, all oscillators achieve a steady state within one period. Their amplitude and phase difference are not affected at all. Only the period is doubled after the adjustment.

Changing α

α is responsible for phase difference modulation. As introduced in Figure 3.10, α lies in the range of [0, 1], and can modulate the phase difference monotonically decreasing from 145° to 45°. Figure 3.11(c) further illustrates how α affects oscillators in the circuit not only on the phase difference, but also on the amplitude and the period. The corresponding parameters are $A = 40$, $\tau = 0.4$ and $\beta = 0$.

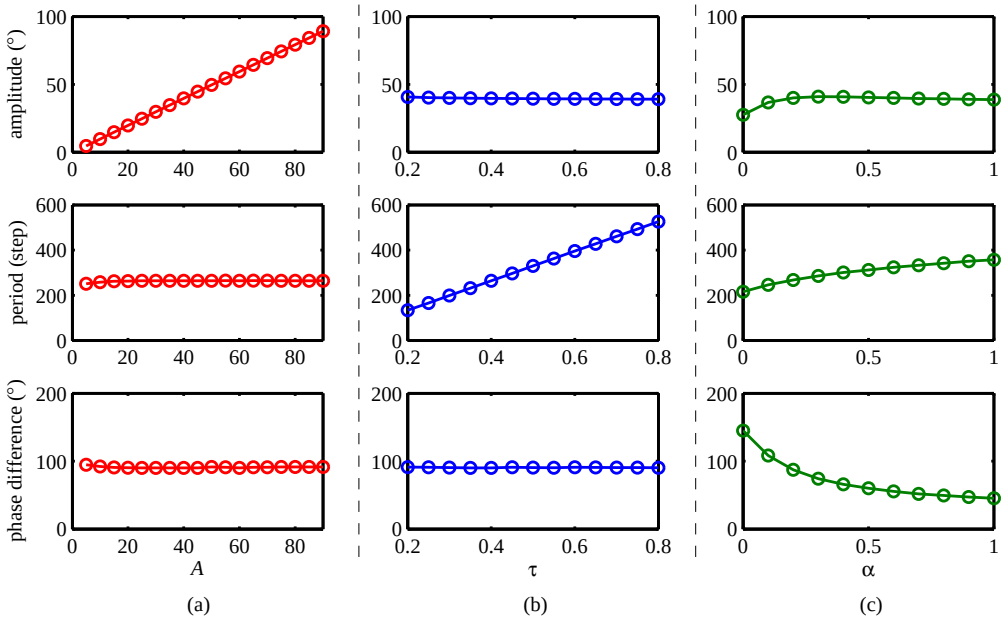


Figure 3.11: Relationship between tuning parameters and oscillatory characteristics of the chained inhibitory CPG circuit. (a)-(c) show the variations of the amplitude, the period and the phase difference with respect to A, τ and α.

Unfortunately, α has a coupling impact on both the the amplitude and the period. For the impact on the amplitude, the growth of α results in slight increase of A, except for the growth of α from 0 to 0.2. For the impact on the period, the growth of α increases the period almost linearly.

Since A has independent impact on the amplitude, it is possible to apply additional gain on A to eliminate the amplitude coupling influence when altering α. Figure 3.12(a) shows the variation of the amplitude with respect to A and α. The resultant amplitude $f_a(A, \alpha)$ is not a plane but a curved surface. In order to decouple α and its influence on amplitude, it is necessory to map $f_a(A, \alpha)$ to a plane that is parallel to the α axis. At the same time, in order to keep the one-to-one mapping between A and the amplitude, the mapping plane is designed as:

$$f'_a(A, \alpha) = A \tag{3.27}$$

Additional gain for A is therefore defined as:

$$gain(f_a, \alpha) = \frac{A}{f_a(A, \alpha)} \tag{3.28}$$

Figure 3.12(b) illustrates the gain of A with respect to A and α. The gain has values around 1 except the area where α is smaller than 0.2. Once α is determined, to get a desired amplitude A_0, A is designed as:

$$A = A_0 \cdot gain(A_0, \alpha) \tag{3.29}$$

In this way, altering α will not affect the amplitude any more. The only disadvantage is that when α is smaller than 0.2, the available range for desired amplitude is shrinked due to the gain and the fixed range of A.

Similarily, to eliminate the period coupling influence for α, additional gain on τ is applied. As shown in Figure 3.13(a), the resultant period $f_p(\tau, \alpha)$ with respect to τ and α is a surface with a small curvature. To remove the coupling influence but remain the relationship between τ and the period, according to equation 3.26, the mapping plane and the gain of τ are designed as:

$$f'_p(\tau, \alpha) = 670 \cdot \tau \tag{3.30}$$

$$gain(f_p, \alpha) = \frac{670 \cdot \tau}{f_p(\tau, \alpha)} \tag{3.31}$$

The gain of τ is tested in the range of (0.7, 1.3), as shown in Figure 3.13(b). α is the dominant factor that affects the gain. The smaller value α has, the bigger gain it would get. Likewise, to get a desired period T_0, τ is designed as:

$$\tau = \frac{T_0}{670} \cdot gain(T_0, \alpha) \tag{3.32}$$

Thus, the change of α has no impact on the period at all. According to the mapping method, no matter how α changes, the desired amplitude and period can be maintained by equations 3.29 and 3.32.

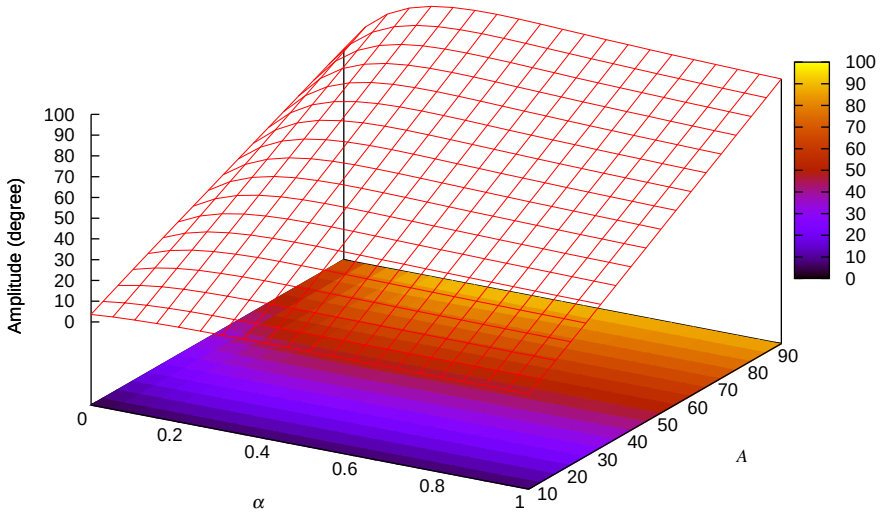
Figure 3.15(c) shows an example of phase difference modulation. A, τ, α and β have values of 40, 0.4, 0.32 and 0, respectively, before the modulation. α is then set to 0.06 at the 4000th time step. The gains for A and τ are also modified at that moment. The phase difference begins to shift after tuning of α. After several periods of oscillation, a new phase lock appears.

Changing β

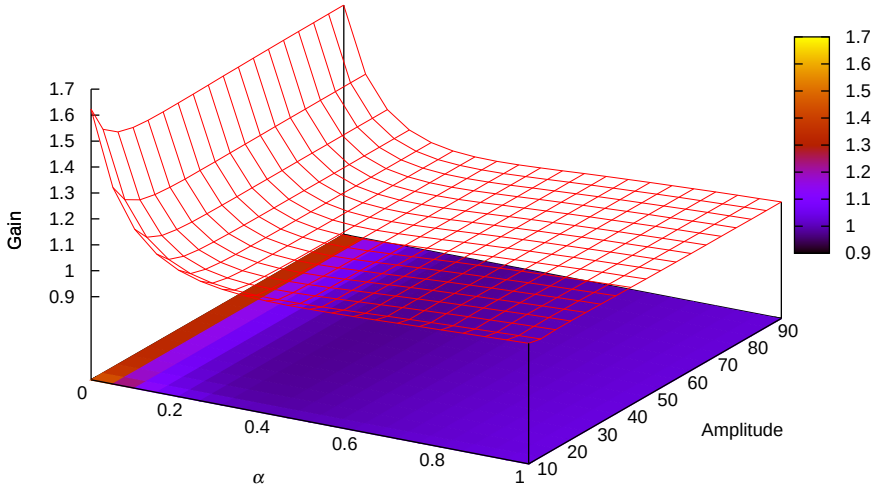
β is the control parameter used for offset modulation. The offset of oscillation is governed by means of interneurons MNs. By assigning opposite values of β to the two MNs in one oscillator, an oscillatory offset can be produced. As described in equation 3.21, the amount of offset is determined by the last term βA. Although the amount of offset is associated with A, the percentage of offset is more concerned. Here the relative offset is defined as a ratio between measured offset and amplitude:

$$relative\_offset = \frac{offset}{amplitude} \tag{3.33}$$

Figure 3.14 illustrates the relationship between the relative offset, A and β, where $\tau = 0.4$ and $\alpha = 0.7$. The resultant surface is a plane that is parallel to the A axis. The relative offset increases with the growth of β. This means A has nothing to do with offset modulation, while β determines the percentage of offset. In addition, through numerical simulation, it has been found that changing β has no impact on the amplitude and the period.

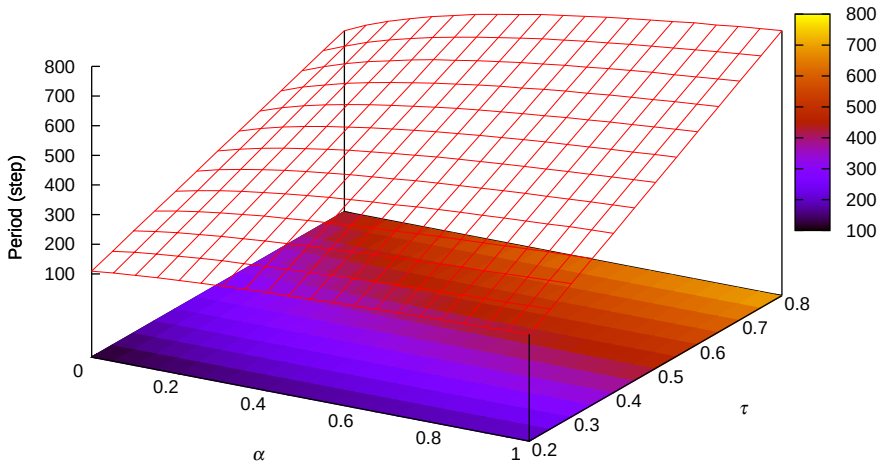


(a)

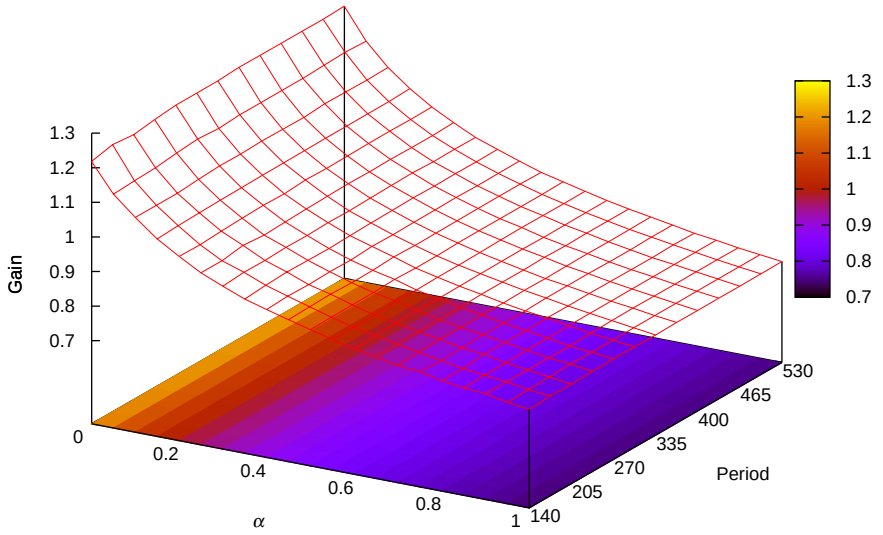


(b)

Figure 3.12: Decouple the influence of amplitude with α. (a) The variation of amplitude with respect to A and α. (b) The gain of A with respect to the desired amplitude and α. The desired ampitude will be maintained if the gain is applied on A.



(a)



(b)

Figure 3.13: Decouple the influence of period with α. (a) The variation of period with respect to A and α. (b) The gain of τ with respect to the desired period and α. The desired period will be maintained if the gain is applied on τ.

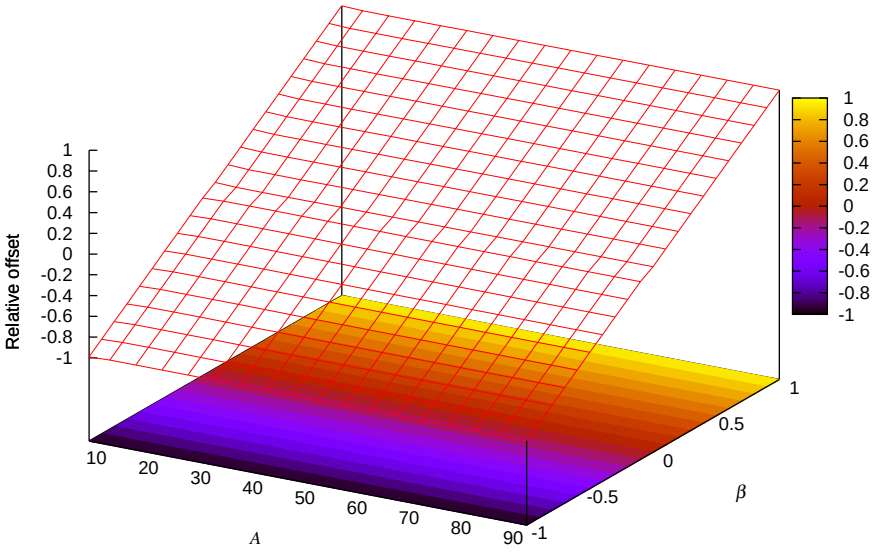


Figure 3.14: The variation of relative offset with respect to A and τ. Altering A will not affet the relative offset. The relative offset is only affected by β in a linear fashion.

To avoid the output exceeding the available range, β is constrained to:

$$|\beta| \leq \frac{90 - A}{A} \tag{3.34}$$

Considering the amplitude in most real applications is usually small (e.g. $A \leq 45$), β is set in the range of [-1, 1] for the rest of this thesis.

In Figure 3.15(d), an example of offset modulation is illustrated. At first A, τ, α and β are set to 30, 0.6, 0.7 and 0, respectively. Offset modulation occurs at the 4000th time step by setting $\beta = \pm 1$ to each side of the MN respectively. Subsequently, the oscillation recovers by setting β back to zero at the 6000th time step. Note that during the offset modulation, the amplitude, the period and the phase difference are not affected at all.

Summary

In general, the chained inhibitory CPG circuit contains explicit and uncoupled control parameters for online modulation. Parameters A, τ, α and β are separately responsible for modulating the amplitude, the period, the phase difference and the offset of oscillation. For each control parameter, the available range of each control parameter and the relationship between its corresponding characteristic in oscillation have been thoroughly investigated. Table 3.4 summarizes the 4 control parameters, as well as their available ranges.

Table 3.4: Control parameters in the chained inhibitory CPG circuit

Symbols	Description	Values
A	Amplitude parameter	(0,90]
τ	Period parameter	[0.2,0.8]
α	Phase difference parameter	[0,1]
β	Offset parameter	[-1,1]

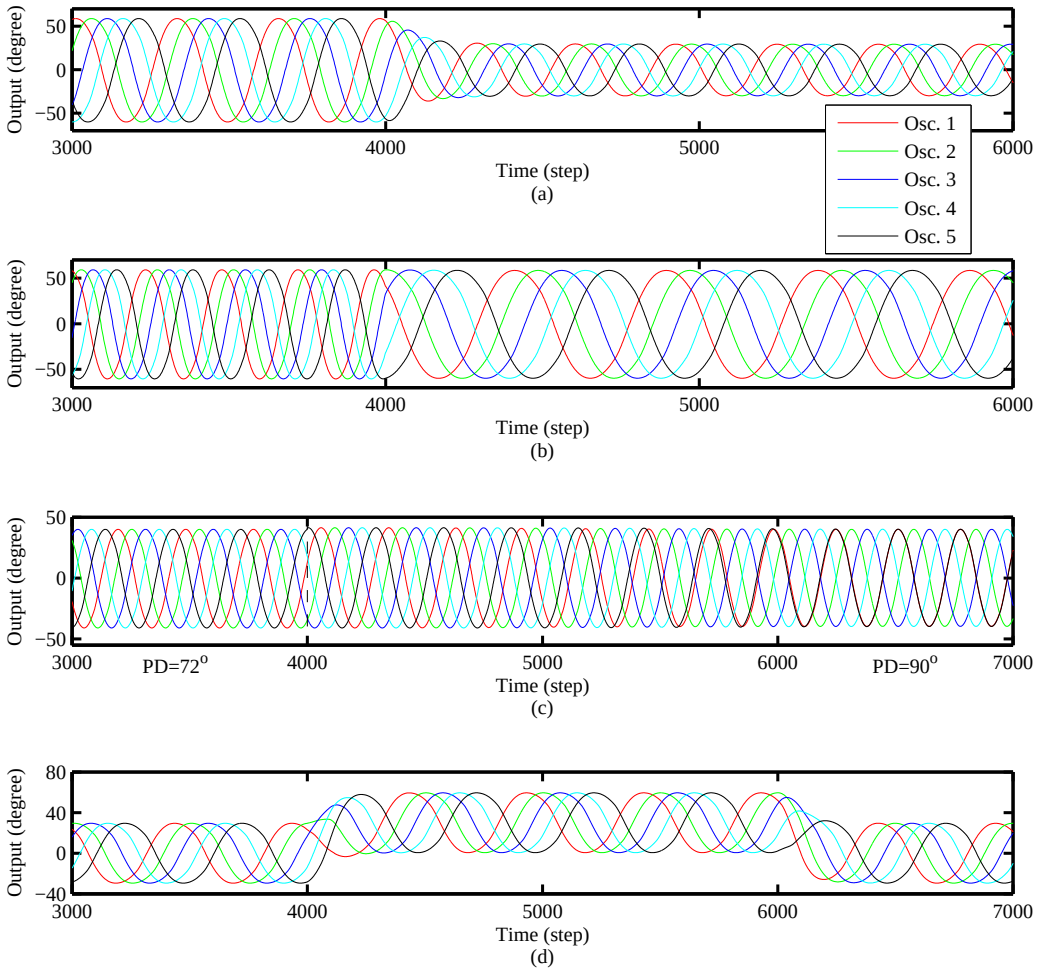


Figure 3.15: Modulation examples for chained inhibitory CPG circuit. (a) Amplitude modulation: alter the parameter A from 60 to 30. (b) Period modulation: double the parameter τ at the 4000th time step. (c) Phase difference modulation: alter the parameter α to shift the phase difference from 72° to 90°. (d) Offset modulation: set the parameter $\beta = \pm 1$ to both the MNs at the 4000th time step, and set β to zero at the 6000th time step to recover.

3.5 Cyclic inhibitory CPG circuit

A mutually inhibitory synapse is another way of connection between oscillators. Instead of producing phase difference between oscillators, the mutually inhibitory synapse forces connected oscillators to generate unified behaviors. Figure 3.16 shows how two identical oscillators are mutually connected by inhibitory synapses. For each of the two oscillators, it projects inhibitory synapses from a pair of EINs interneurons in itself to a pair of ipsilateral LINs interneurons in the other oscillator. The 4 inhibitory synapses comprise the mutually inhibitory connetion. The simplified form of connection is also illustrated in this figure.

In this section, a cyclic inhibitory CPG circuit is designed based on the mutually inhibitory connection. As illustrated in Figure 3.17, inhibitory synapses are emitted to the neighboring oscillators on the ipsilateral side bilaterally. In addition, the first and the last oscillators are also mutually connected by inhibitory synapses, which forms a loop of inhibitory connected oscillators. To facilitate analysis of the CPG circuit, it is assumed that the synaptic weights ω_c among oscillators have a unified value. By adjusting ω_c, the cyclic connected inhibitory oscillator loop exhibits two interesting oscillatory phenomena: synchronization and maintenance.

3.5.1 Synchronization activity

When small inhibition, e.g., $-0.2 \leq \omega_c < 0$, is applied among oscillators in the cyclic inhibitory CPG circuit, the synchronisation phenomena occurs. All the oscillators will not only produce the same amplitude and period, but also keep the oscillation in phase. In this thesis ω_c is set to -0.1 for the generation of synchronization activity.

The modulation of synchronization activity is studied. As discussed before, there are 3 tunable parameters A, τ and β in oscillators that control the amplitude, the period and the offset of oscillation, respectively. In the modulation, they are supposed to have the same available ranges as the corresponding control parameters listed in Table 3.4. To analyze how they affect the synchronization activity, the following investigates each parameter and tries to find out its relationship between the circuit's output.

Changing A

To evaluate the relationship between A and the related characteristics of the synchronized oscillation, τ and β are fixed with values of 0.3 and 0, respectively. Figure 3.18(a) illustrates the variation of the amplitude, the period and the offset with respect to A. The result shows that altering A has no impact on the period and the offset, but results in a linear change of the amplitude. The relationship between the amplitude and A can be described as:

$$Amplitude = 0.5 \cdot A \tag{3.35}$$

Through numerical simulation, it is observed that the available range of A shrinks. The synchronization activity disappears once A exceeds a value of 70.

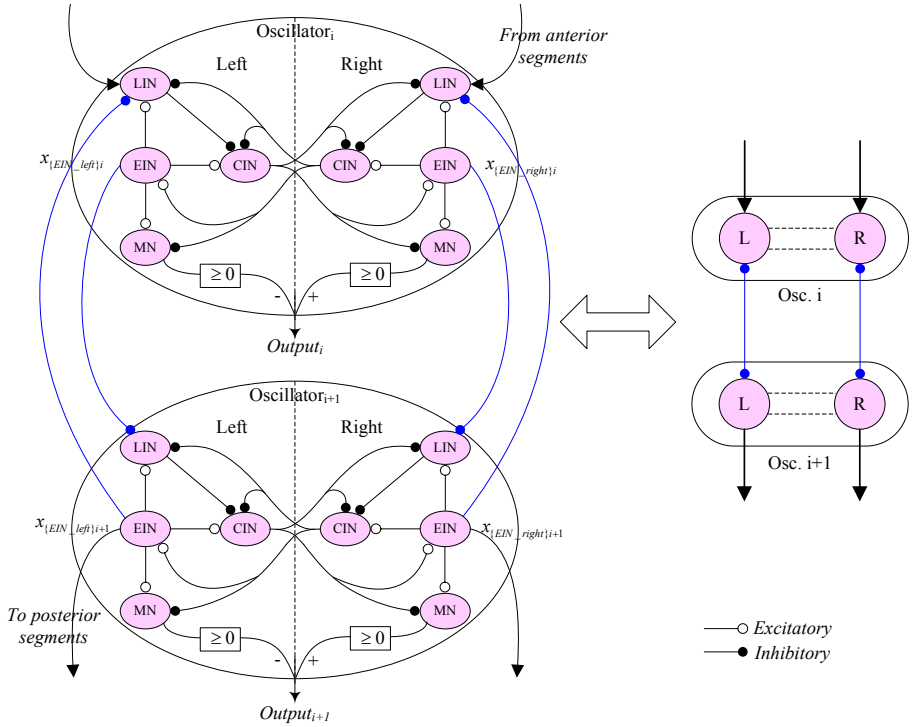


Figure 3.16: The mutually inhibitory connection between two oscillators and its simplified form. The two oscillators both emit a pair of inhibitory synapses to each other, from EINs in itself to the ipsilateral LINs in the other oscillator. For concise reason, the mutually inhibitory synapses are further simplified as a line with solid circle on each end.

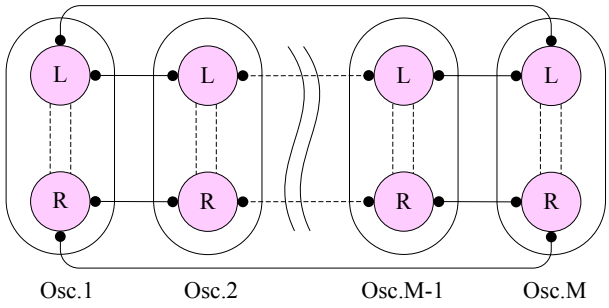


Figure 3.17: The cyclic inhibitory circuit. The ipsilateral side of oscillators forms an inhibitory loop. Altering the synaptic weight among the oscillators will cause synchronization or maintenance activity.

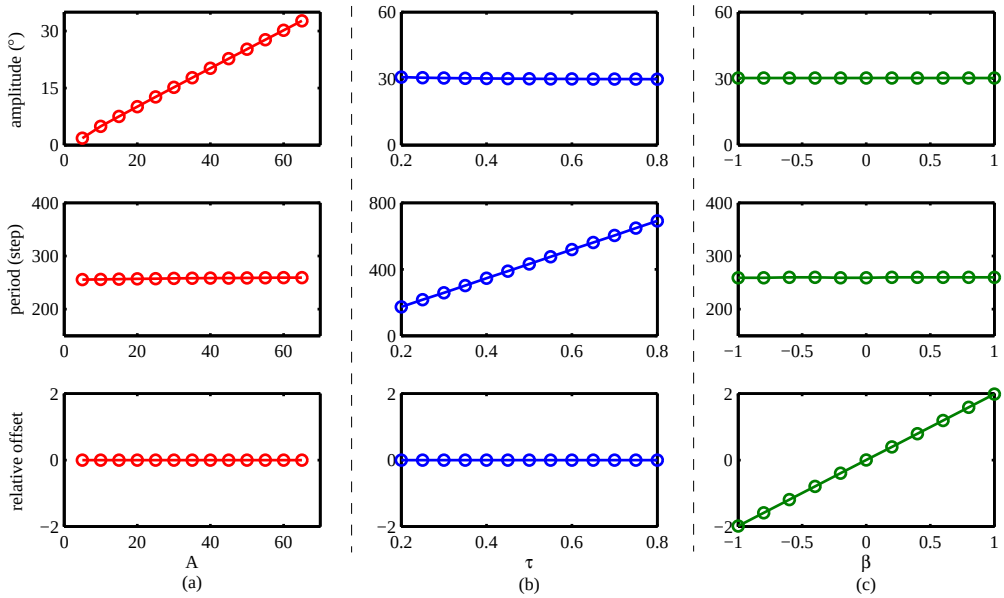


Figure 3.18: Relationship between tuning parameters and characteristics of synchronized oscillation in the cyclic inhibitory CPG circuit. (a)-(c) show the variations of the amplitude, the period and the offset with respect to A, τ and β.

Therefore, to perform a reliable amplitude modulation for the synchronized oscillation, A is constrained in the range of (0, 70).

An example of amplitude modulation is illustrated in Figure 3.19(a). A, τ and β are separately set to 60, 0.3 and 0 at the beginning. Within one period all oscillators are synchronized. By decreasing A by half at the 1500th time step, the amplitude modulation starts. A rapid amplitude change from 30 to 15 degrees appears thereafter without influencing the period and the offset of oscillation.

Changing τ

Likewise, A and β are fixed so as to examine how τ influences the circuit's output. Figure 3.18(b) shows the amplitude, the period and the offset of the synchronized oscillation vary with respect to τ, where $A = 60$ and $\beta = 0$. It has been found that the amplitude and the offset are not affected during the change of τ. Instead, τ is only involved in modifying the period. The result reveals that the period is governed by τ in a approximate linear fashion, as described bellow:

$$period \approx 860 \cdot \tau \tag{3.36}$$

Figure 3.19(b) is an example that explains the period modulation under the synchronized oscillation. Before the modulation, A, τ and β have fixed values of 40, 0.3 and 0, respectively. The period modulation occurs at the 1500th time step. By doubling the value of τ, the period scales twice as wide as the original one. The amplitude and the offset are found to be not affected during the modulation.

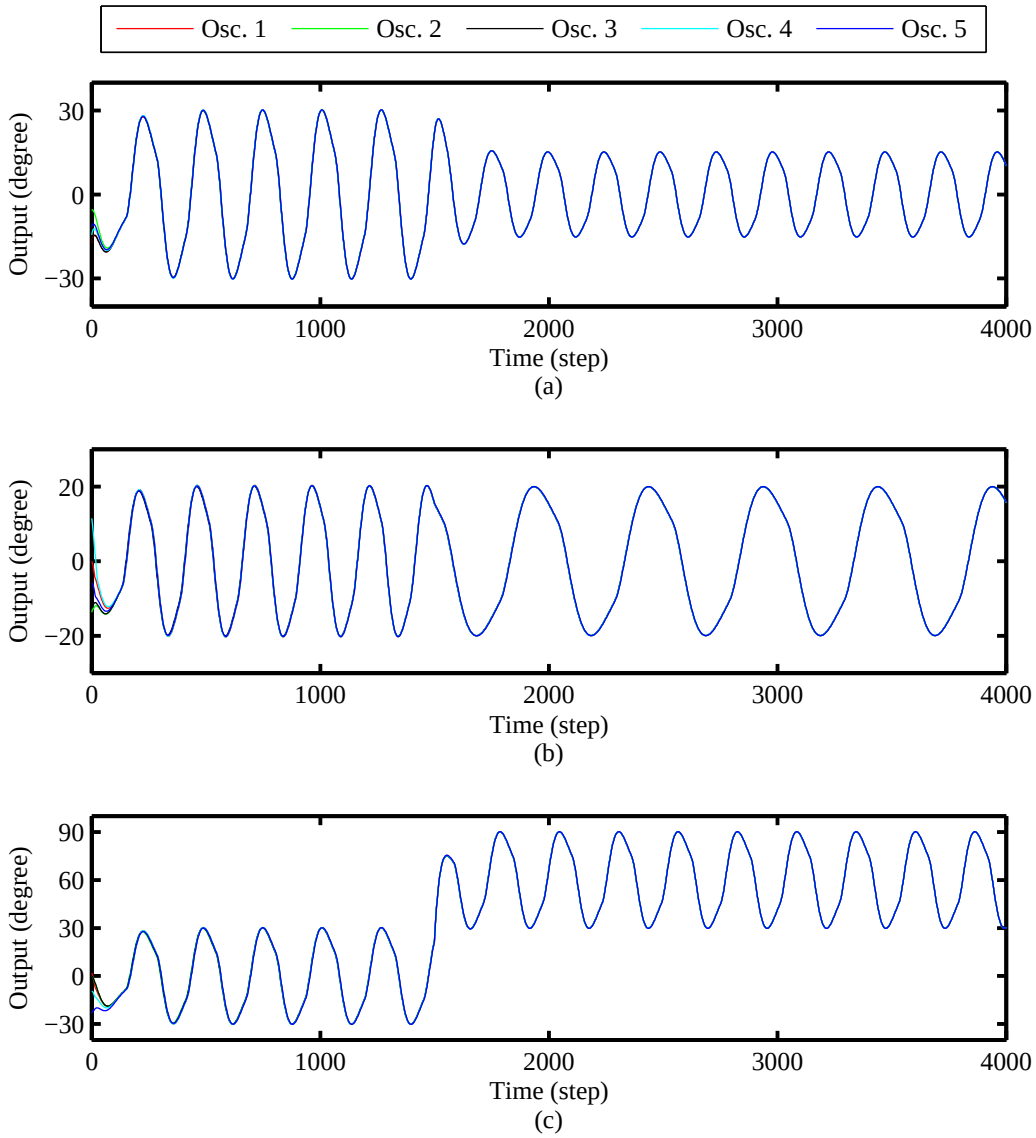


Figure 3.19: Modulation examples for synchronization activity in cyclic inhibitory CPG circuit. (a) Amplitude modulation: A is modified from 60 to 30 at the 1500th time step. (b) Period modulation: τ is doubled from 0.3 to 0.6 at the 4000th time step. (c) Offset modulation: $\beta = \pm 1$ is assigned to both the MNs at the 1500th time step.

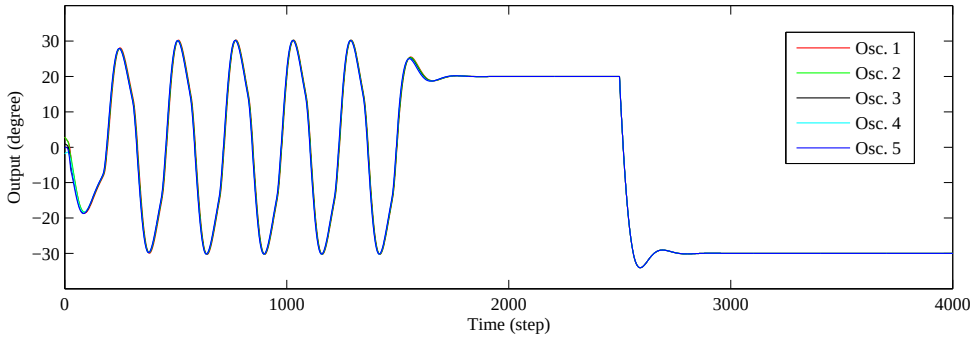


Figure 3.20: Modulation examples for maintenance activity in cyclic inhibitory CPG circuit. The maintenance activity starts at the 1500th time step by modifying ω_c from -0.1 to -1 and A from 60 to 20. Modulation with negative output occurs at the 2500th time step, with $A = 30$ and value exchange between CINs in oscillators.

Changing β

For the sychronized oscillation, opposite values of β attached on the two MNs in each oscillator is still valid for offset modulation. How β affects the characteristics of the synchronized oscillation is tested with $A = 60$ and $\tau = 0.3$. From Figure 3.18(c), It is observed that altering β has no effect on the amplitude and the period of oscillation. β only determines the degree of offset oscillation. The relationship between β and the relative offset is linear proportion, which can be described as follows:

$$relative\_offset = 2 \cdot \beta \tag{3.37}$$

In Figure 3.19(c), an offset modulation under sychronized oscillation is performed, with fix values $A = 60$ and $\tau = 0.3$. By altering β from 0 to ± 1 at the 1500th time step, the offset modulation occurs that it makes an offset of the oscillation upward by 60 degrees. It is noted that the modulation is finished within one period and no amplitude and period are affected.

3.5.2 Maintenance activity

Maintenance is another phenomena that appears only when strong inhibition, e.g. $\omega_c < -0.2$, is applied among oscillators in the cyclic inhibitory CPG circuit. In the maintenance activity, the output of all oscillators are fixed and can be held at the same value over time, taking the place of the synchronized oscillation. This thesis uses $\omega_c = -1$ as the default value for generation of the maintenance activity.

The modulation is also feasible for the maintenance activity. It has been found that the output of all oscillators are equal to the value of A. By adjusting A, the maintained value can be modified in a linear manner. Since A is always positive in the available range, it is impossible to generate negative output for the maintenance activity. Nevertheless, this problem can be resolved by exchanging the values of the two interneurons CINs in each oscillator. This would cause a series of interaction

between interneurons and finally reverse the output of the maintenance activity.

Figure 3.20 explains the modulation of the maintenance activity. At the beginning, synchronized oscillation is performed by setting $\omega_c = 0.1$, with other parameters $A = 60$, $\tau = 0.3$, $\beta = 0$. The maintenance activity and its modulation start at the 1500th time step. Assigning $\omega_c = -1$ and $A = 20$ result in the generation of the maintenance activity with a fixed output value of 20. At the 2500th time step, another modulation occurs. By setting A to 30, together with exchanging values between CINs in oscillators, the circuit in maintenance activity successfully produces a negative output with a value of -30.

3.6 Summary

This chapter introduces the design of a lamprey spinal generator. We investigate some famous CPG models and analyze their advantages and disadvantages. Based on our design goals, we present a connectionist CPG model derived from the lamprey's spinal cord. The proposed CPG model uses a set of excitatory and inhibitory interneurons for oscillation generation. It has four interesting features. First, it has rich dynamics of oscillatory activities that is benefit to the design of limbless gaits. Second, it provides simple but uncoupling parameters for output modulation. Third, it is easy to integrate sensory reflex mechanisms. And fourth, it allows sensory feedback integration so that adaptive limbless locomotion can be realized. This chapter only emphasizes the first two properties.

In the CPG model, the oscillator is designed to have a symmetric connection with four types of interneurons. For the oscillator, we use sigmoid function to describe the variation of the EINs type of interneurons, while use the leaky integrator to simulate the rest types of interneurons. By the interaction of these interneurons, the oscillator model forms the self-sustaining mechanism.

Baesd on the oscillator model, we design a chained inhibitory CPG circuit and explore its control parameters in detail. This circuit is able to generate travelling waves between oscillators. Through numerical simulations, the CPG circuit is verified for providing explicit parameters for output signal modulation, including the modulation of amplitude, period, offset and phase difference. To extend the activity of oscillation, we also design a cyclic inhibitory CPG circuit. Two type of activities, i.e. synchronisation and maintenance activities, are found and investigated in this circuit, including their control parameters and the corresponding characteristics.

Chapter 4

Design of Limbless Locomotion

Contents

4.1 Introduction

So far the proposed CPG model and its control parameters have been thoroughly studied in the last chapter. From this chapter we begin to apply the CPG model to limbless robots and study how to use it to implement 3D limbless locomotion patterns.

In the field of robotics, in order to reduce the consumption of real robots during experiments, simulation is a preferred way used to test and verify proposed ideas. In this thesis, open dynamics engine (ODE) is utilized as the simulation platform (Smith, 2008). ODE is an open source, high performance library that is good at simulating rigid body dynamics. It has advanced joint types and an integrated collision detection mechanism. Therefore it is usually used for simulating robotic locomotion in virtual reality environments.

Here a pitch-yaw connected modular robot is constructed in the ODE environment, as seen in Figure 4.1. For the simulated robot, pitch modules in yellow and yaw modules in red are alternately connected. Each module is designed to have two parts. Each part is represented as a simple rigid box. A joint is located between the two parts of a module, so that the module can rotate along the horizontal or vertical axis within a range of $\pm 90°$. The related parameters of the module are listed in Table 4.1. In addition, in order to simulate the real-world environment, gravity and friction are also taken into account. An acceleration of gravity with a value of -9.81 m/s^2 and a friction coefficient of 0.4 are used in this simulation.

In the following sections, four types of limbless gaits, namely, sidewinding, rolling, turning and flapping are studied, not only for the construction of the CPG network but also for the analysis of the locomotion speed with respect to the tuneable parameters of the CPG network. Furthermore, an on-site experiment is also carried out to verify the effectiveness of these CPG-based limbless gaits.

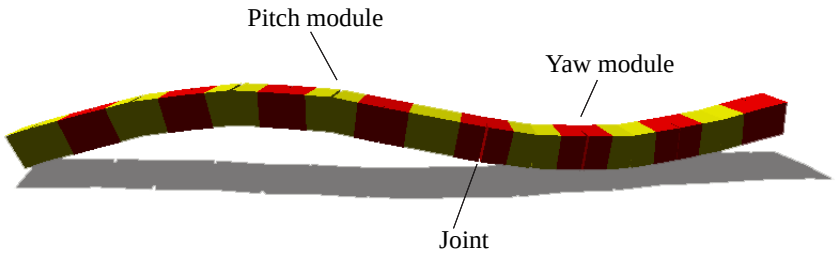


Figure 4.1: The simulated limbless robot.

Table 4.1: Specification of simulated robot module

Component	Parameter	Value
Module	Length (mm)	72
	Width (mm)	52
	Height (mm)	52
	Weight (g)	150
Joint	Rotate speed (rad/s)	≤ 1.45
	Torque $(N \cdot m)$	≤ 0.314

4.2 3D gait implementation

In this section, we utilise the chained and cyclic inhibitory CPG circuits discribed in the last chapter to implement four types of 3D limbless locomotion patterns, namely, side winding, rolling, turning and flapping.

4.2.1 Sidewinding gait

Sidewinding is a sideway type of locomotion that is used by snakes when moving on the sandy surfaces of the desert. Sidewinding is derived from lateral undulation, but differs in the pattern of bending. First, the head is lifted off the ground and laterally set down again a short distance away. Then the body follows the path of the head. During the following phase, the head begins a new round of lateral movement while the rear part of the body completes the old track. The movement has only two points in contact with the ground during the movement, which prevents the snake from overheating due to excessive contact with the desert sand.

4.2.1.1 Gait design

The modules of the constructed modular robot are categorized into two groups: one group with all the pitch modules and the other group with all yaw modules.

Considering that both the chained and cyclic inhibitory CPG circuits have rich dynamic activities, it is possible to utilize one of the CPG circuits to control one of the grouped modules. Once the two CPG circuits that apply on the two grouped modules are properly connected, a limbless locomotion pattern is generated.

For the design of the sidewinding gait, two chained inhibitory CPG circuits are utilized for the pitch and yaw grouped modules, as shown in Figure 4.2. The purpose of employing chained inhibitory CPG circuits is to generate travelling body waves along the modular robot. For the pitch group, an additional command oscillator is used for generating phase differences between the oscillators. For the yaw group, there is no additional command oscillator. Instead, an inhibitory synapse emitted from one oscillator in the pitch group is projected to the first oscillator in the yaw group.

The inhibitory synapse between the two groups has two functions. First, it makes the oscillators in the yaw group have the same phase difference as the one in pitch group, which therefore enables the pitch and yaw grouped modules to propagate body waves in the same direction. Second, since it is emitted from one oscillator in the pitch group, the two groups of oscillators maintain a phase difference. This makes all the modules not in contact with the ground at the same time, and results in the body shape looking like a flattened coils of a helix.

The phase difference for the two groups of oscillators is defined as the phase difference between the head oscillators in pitch and yaw groups, respectively. Since the inhibitory synapse between the two groups is projected to the first oscillator in the yaw group, therefore the phase difference for the two groups is related to the position where the inhibitory synapse emits. In this thesis, we suppose that the phase difference for the two groups is close to 90°. Thus, the emitted position of the inhibitory synapse *pos* can be calculated as:

$$pos = \lfloor \frac{90}{pd} \rfloor \tag{4.1}$$

where *pd* represents the phase difference among the oscillators in the pitch group.

Based on the sidewinding circuit, two more conditions need to be met to generate a feasible sidewinding gait. First, all the oscillators, no matter in the pitch group or in the yaw group, should have the same amplitude and period. Second, the inhibitory synapse between the two groups should have the same synaptic weight as the one in the chained inhibitory CPG circuit. If any one of the conditions is not met, the resultant movement is unstable and chaotic.

A simulated example is carried out with control parameters $A = 25$, $\tau = 0.3$, $\alpha = 1$ and $\beta = 0$ in the chained inhibitory CPG circuit. A phase difference of 45° among oscillators is obtained according to Figure 3.10. In addition, the emitted position of the inhibitory synapse between the two groups is determined according to equation 4.1. It is the second oscillator in the pitch group that emits the inhibitory synapse to the first oscillator in the yaw group.

Figure 4.3 shows the detailed process of the sidewinding gait, where the robot is shifting its body laterally. During the movement, each module along the body of

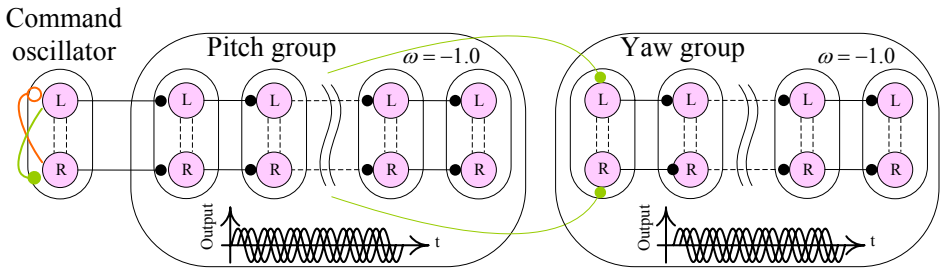


Figure 4.2: Sidewinding circuit. Two chained inhibitory CPG circuits with a synaptic weight of -1.0 among them form a sidewinding circuit.

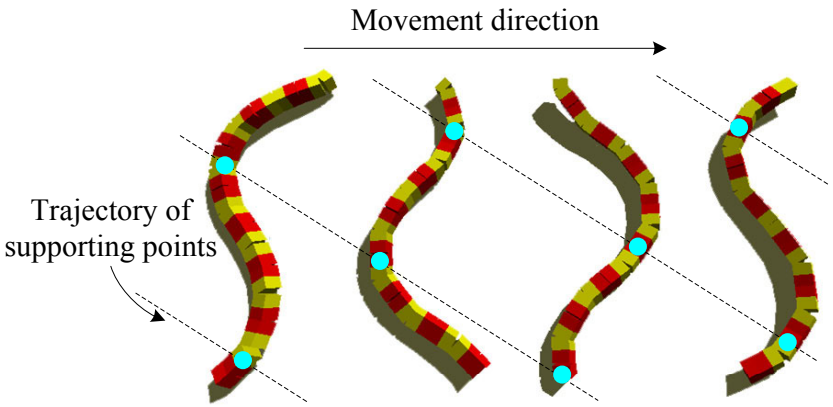


Figure 4.3: The simulation of the sidewinding gait.

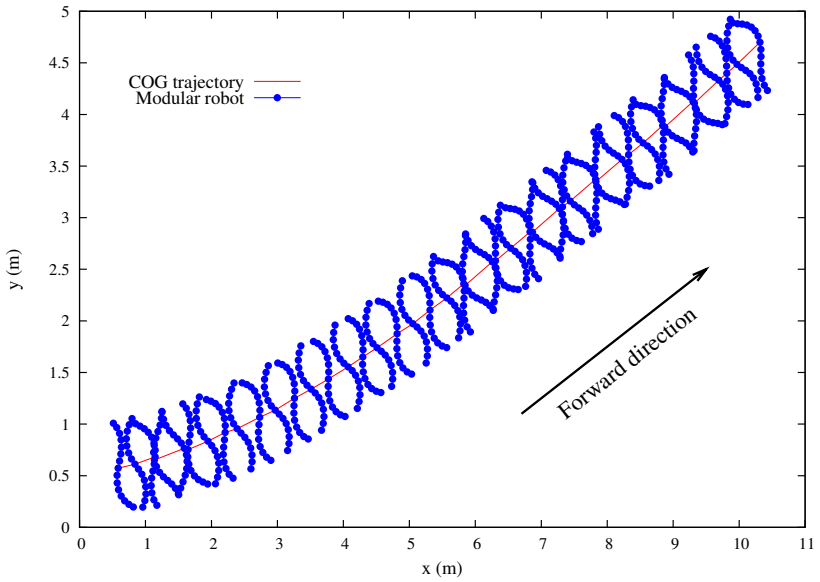


Figure 4.4: The trajectory of the sidewinding gait.

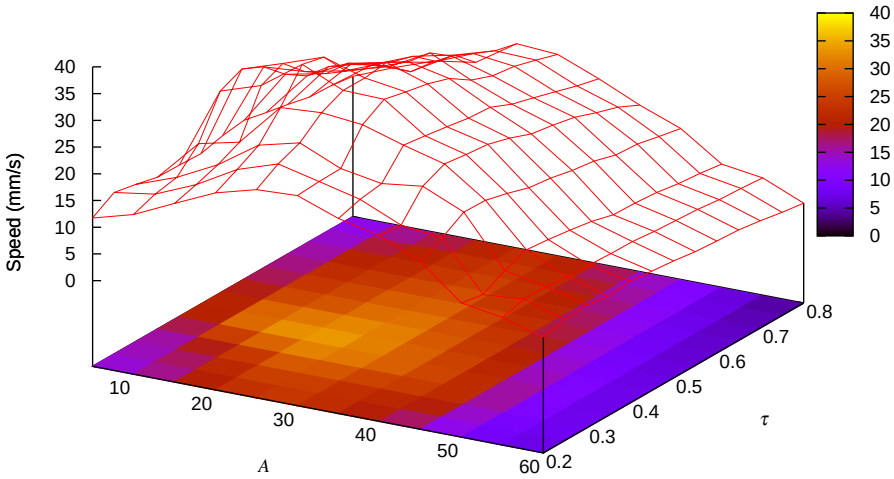


Figure 4.5: Sidewinding speed variation with respect to A and τ, where $\alpha = 0.8$.

the robot is sequentially placed in static friction with the substrate. Note that only two supporting points remain in contact with the ground. Because of the static contact and lifting of the body, the robot travels roughly diagonally relative to the tracks it forms on the ground.

Figure 4.4 shows the trajectory of this movement. There is slippage happening when the robot starts from a standstill, resulting in a slight change of the forward direction at the beginning. In spite of this, after several periods of sidewinding movement, the gait becomes stable and the forward direction is no longer changed.

From the example, it shows that the sidewinding circuit can generate the sidewinding gait that is similar to the locomotion pattern of snakes in nature.

4.2.1.2 Speed analysis

As discussed in Section 3.4, A, τ, α and β separately control the amplitude, period, phase difference and offset of oscillation in the chained inhibitory CPG circuit. In the speed analysis, A is constrained in the range of (0, 60] due to security concerns of real applications. Small value of α has more intensive sampling points owing to its asymmetric affect on phase difference (see Figure 3.10). In addition, since a non-zero value of β leads to the distortion of the sidewinding movement, β is fixed to 0 and is not involved in the sidewinding gait generation. Below we investigate how the lateral motion speed of sidewinding varies with the parameters A, τ and α, respectively.

In Figure 4.5, the speed of sidewinding with respect to A and τ is examined. The speed increases when A increases from 0 to 30, but decreases when A continues to grow. τ has no significant influence on the speed. Although tuning of τ can change the rotational frequency of individual modules (see details in 3.11(b)), a high speed of sidewinding appears only when τ is around 0.4.

Figure 4.6 illustrates how the speed varies with parameters τ and α. Likewise, τ

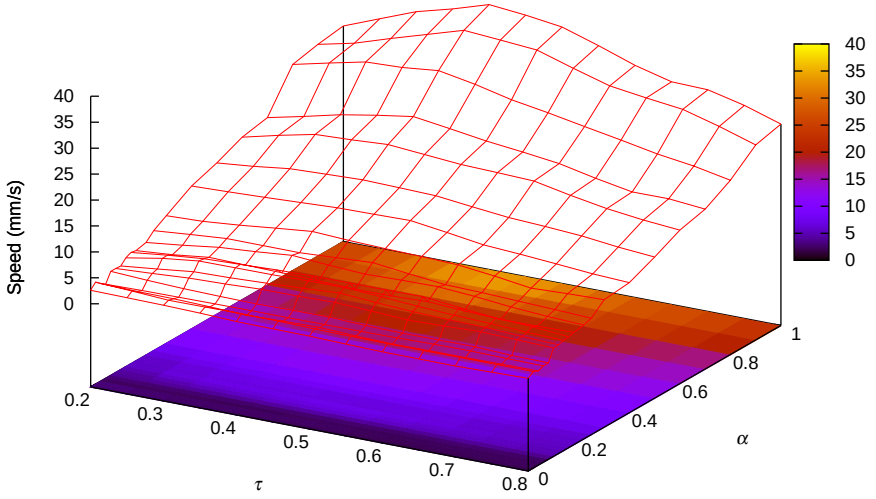


Figure 4.6: Sidewinding speed variation with respect to τ and α, where $A = 20$.

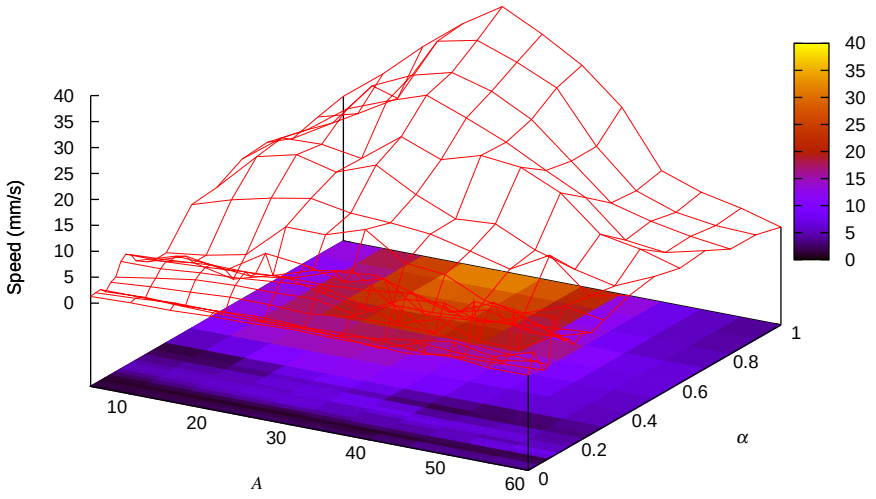


Figure 4.7: Sidewinding speed variation with respect to A and α, where $\tau = 0.4$.

does not affect the speed too much. This is because effective sidewinding is not only determined by the rotating speed of individual modules, but depends even more on the joint coordination of the pitch and yaw grouped modules. α determines the phase difference between modules, thus also affect the lateral speed of sidewinding. From this figure, it reveals that the speed increases with the growth of α.

From Figure 4.7, it is shown that the lateral motion speed decreases when A increases or α decreases. A high speed of sidewinding appears only when A is in the range of [15, 35] while α is in the range of [0.6, 1]. The reasons are twofold. First, when sidewinding, the curvature of the robot body shape grows proportionally with respect to the growth of A, which gradually decreases the distance of the two supporting points on the ground and makes the gait unstable. Even though the step length per period increases as A increases, the unstable motion that leads the center of gravity (COG) backward counteracts the forward displacement. Second, as discussed in Chapter 3, the phase difference among the modules can be increased by decreasing the parameter α (see Figure 3.11(c) in detail). While the curvature of the robot body shape decreases proportionally with respect to the growth of the phase difference among the modules, which in turn reduces the lateral motion speed. That is why decreasing α results in the decrease of the lateral motion speed.

4.2.2 Rolling gait

Rolling is a special behavior that can be seen when animals are preying, fighting or escaping. For example, a crocodile starts rolling under the water after it catches its meal. The rolling action allows the crocodile to disorient and drown the prey at its own pace.

Inspired from nature, rolling can also be applied on limbless robots. A limbless robot curving slightly into a 'C' shape and rotating along its body axis can produce lateral momentum, resulting in the lateral rolling of the robot. Rolling is a quick and reliable way to move the limbless robot sideways. It is useful in complex environments, especially when traversing on uneven terrains. Since rolling friction is much less than sliding friction, rolling is considered as an energy efficient gait.

4.2.2.1 Gait design

A rolling circuit is constructed with two cyclic inhibitory CPG circuits, as shown in Figure 4.8. For each group, small inhibition with a value of -0.1 is utilized among oscillators in the cyclic inhibitory CPG circuit, which is on purpose to generate synchronized oscillation on modules. Due to the two groups of synchronized modules, two circular body waves would appear, one in the horizontal plane and the other in the vertical plane. In order to generate a stable rolling gait, one prerequisite is to ensure all oscillators in the two cyclic inhibitory CPG circuits to have the same amplitude and period. If the requirement is met, the resulting body shape of rolling gait would be the superposition of two identical circular body waves.

Besides the requirement on the amplitude and period, a stable rolling gait also

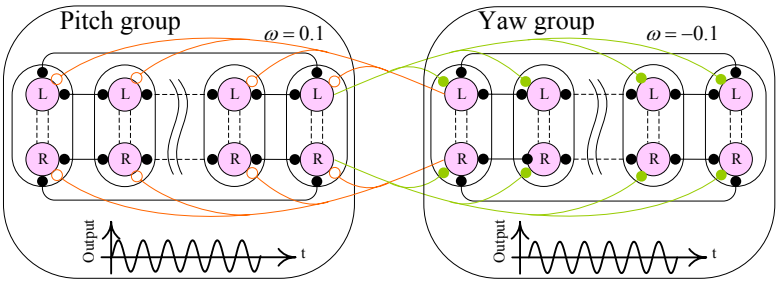


Figure 4.8: Rolling circuit. Two cyclic inhibitory CPG circuits with two groups of additional exitatory and inhibitory synapses form a rolling circuit.

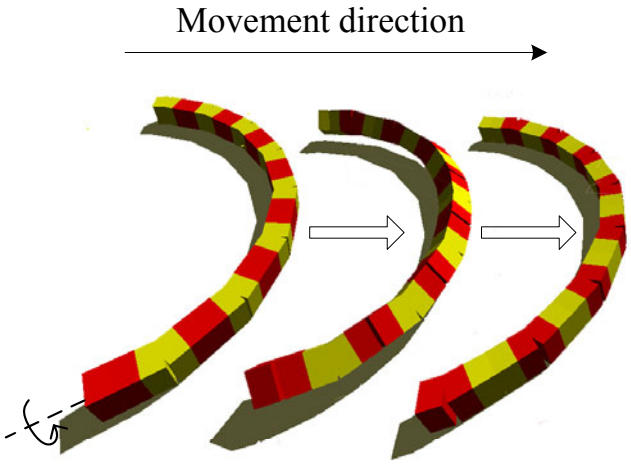


Figure 4.9: The simulation of the rolling gait.

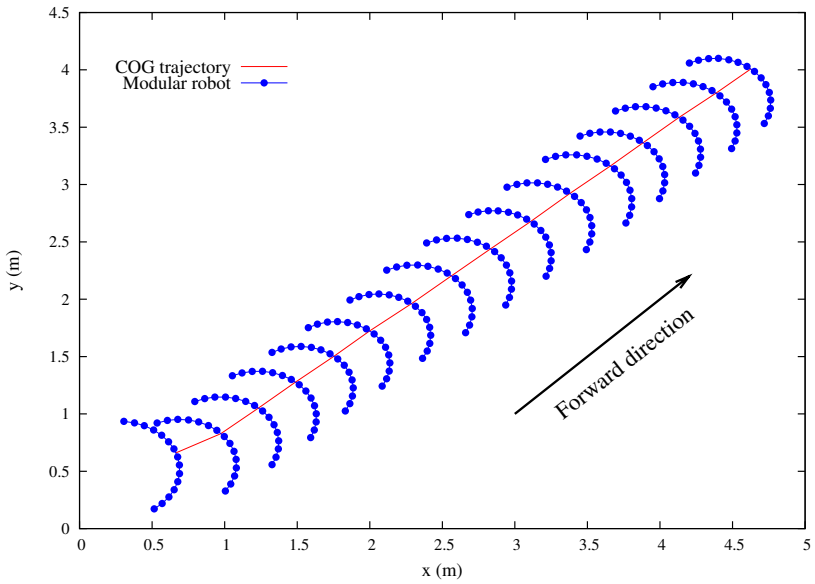


Figure 4.10: The trajectory of the rolling gait.

requires a phase difference between the pitch and yaw grouped modules. If the bending angles between the two grouped modules are out of phase 90°, the center of gravity could be maintained approximately in the same height. That is because when some angles in one group reach their maximum values, the angles in the other group will be zero and vice versa. To meet this condition, additional synapses have been added between the two cyclic inhibitory CPG circuits. The border oscillator in each of the cyclic inhibitory CPG circuit separately emits excitatory and inhibitory synapses to each oscillator in the other group, so as to produce a fixed phase difference between the two oscillator groups. Among these additional synapses, synaptic weights of 0.1 for exitatory synapses and synaptic weights of -0.1 for inhibitory synapses can generate a phase difference of 90°. In addition, changing the sign of the two groups of additional synapses would generate an opposite phase difference of $-90°$, which finally leads to the change of rolling direction.

Figure 4.9 shows a simulated example with control parameters $A = 20$, $\tau = 0.3$ and $\beta = 0$ in the cyclic inhibitory CPG circuit. When the modules in the pitch group bend to maximum values, there is no bending in the yaw group. The simulated limbless robot therefore has only one circular body wave along the pitch group. With the decrease of angles in the pitch group and the increase of angles in the yaw group over time, two circular body waves appear. Due to the torque among modules, the robot gradully lifts up both of its ends and bends in a leaned posture. This state continues until the modules in the yaw group bend to maximum values while the modules in the pitch group bent back to zero values. The robot then again has only one circular body wave which bends along the yaw group. As a result of the alternate appearance of the three states, the robot succeeds to roll sideways.

Figure 4.10 illustrates the trajectory of the rolling gait. Similar to the case of sidewinding, the robot slips from a standstill due to the sliding friction, which causes the change of forward direction at the beginning. Nevertheless, once the robot starts to roll, its forward direction is not changed anymore.

From the example, it shows that the rolling gait can be realized with the help of the rolling circuit.

4.2.2.2 Speed analysis

As discussed in Section 3.5, the cyclic inhibitory CPG circuit has three tunable parameters A, τ and β that control the amplitude, the period and the offset of the synchronized oscillation. Since offset modulation is not used in the synchronized oscillation when generating rolling gait, only A and τ have impact on the forward speed of rolling movement.

Figure 4.11 illustrates how the forward speed varies with respect to A and τ. For parameter τ, it hardly affects the rolling speed. Because decreasing τ can correspondingly decrease the period of the synchronized oscillation (see details in 3.18(b)), it finally causes the increase of rotating speed of individual modules. Therefore, the rolling speed increases with the decrease of τ. However, this variance is not obvious when A has large values. A high speed of rolling can be achieved if

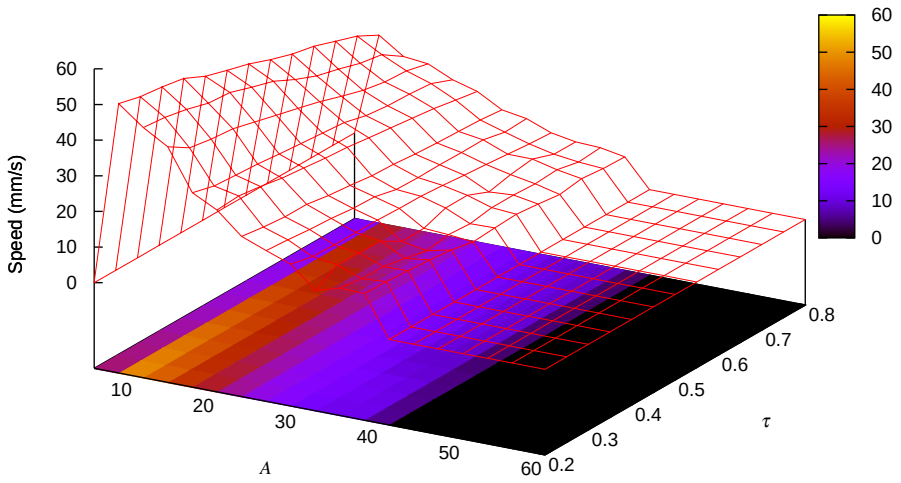


Figure 4.11: Rolling speed variation with respect to A and τ.

τ is in the range of [0.2, 0.4].

For parameter A, it dominates the speed change of forward movement. When A is small, i.e. less than 8, the simulated limbless robot performs a circular body shape with a small curvature. The curvature is so small that the variation of the body shape can not provide enough torque for rolling. Thus, the robot can only lift both ends up and down and move them back and forth. As a result, the robot slides forward and backward with a small speed. When A becomes larger, the rolling movement appears. Because increasing A will correspondingly increase the bending curvature of the body shape, the internal force among modules is also increased, resulting in a more and more slow and laborious rolling behavior. A high speed of rolling appears when A is in range of [10, 20]. When A continues to increase, i.e, exceeding the value of 40, collision may occur between each end of the robot. Sicne the interaction at both ends makes the robot get stuck, the forward speed decrease to zero, as shown in the black region of Figure 4.11.

4.2.3 Turning gait

Caterpillars achieve linear motion by squeezing muscles in sequence, forming a series of undulating body waves from tail to head. A step cycle begins by lifting up the terminal prolegs forward and anchoring one step ahead. Once these prolegs are set, the body wave propagates forward and muscles from each body segment contract serially.

Turning is a variant of linear movement. Caterpillars' body movement occurs not only in a vertical plane, but also in a horizontal plane. Caterpillars are able to bend body segments horizontally during the linear movement. Such turning behavior is usually seen when they are searching for food. By using the turning gait, caterpillars can easily change forward direction, thus facilitating them to discover and approach food.

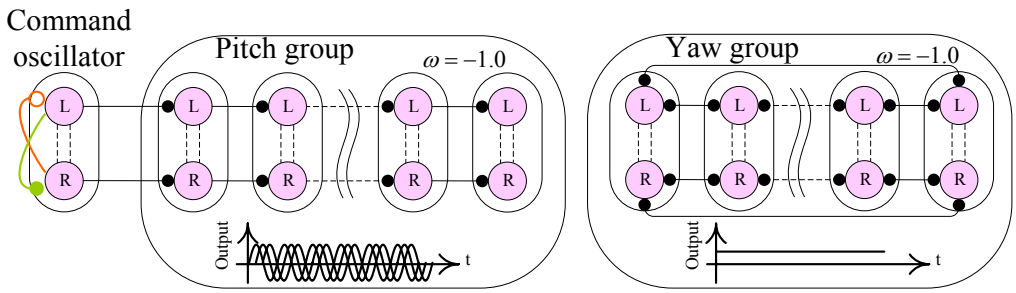


Figure 4.12: Turning circuit. A chained and a cyclic inhibitory CPG circuits with no synapses among them form a turning circuit.

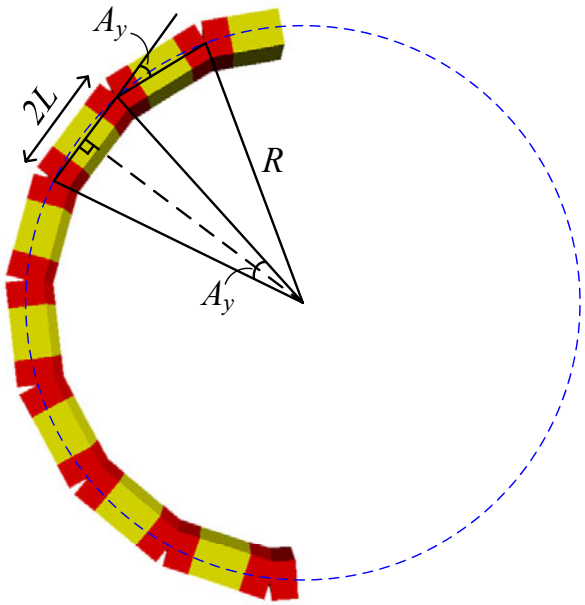


Figure 4.13: The geometry of the turning radius.

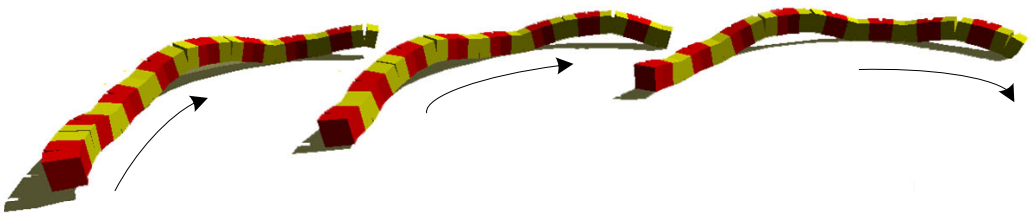


Figure 4.14: The simulation of the turning gait.

4.2.3.1 Gait design

Since a turning gait can be decomposed into two independent movements: a linear movement and a bending behavior, a turning circuit is designed to have two independent CPG circuits, as shown in Figure 4.12. For the pitch group, a chained inhibitory CPG circuit is applied. The purpose is to produce linear movement in the vertical plane. For the yaw group, it is equipped with a cyclic inhibitory CPG circuit. To achieve the bending behavior, strong inhibition is applied among oscillators in the cyclic inhibitory CPG circuit. Once the maintenance activity occurs, a fixed circular body wave along the yaw modules appears.

The output of oscillators in the cyclic inhibitory CPG circuit determines the degree of bending behavior. As discussed in Section 3.5.2, their outputs are equal to parameter A (To distinguish parameter A between the pitch and yaw groups, here A_p are used for the pitch group while A_y used for the yaw group). Considering the extreme situation when the robot is bending to a circle, collision may occur at both ends. To avoid this situation, A_y should satisfy:

$$A_y < \frac{360}{\frac{M}{2}} = \frac{720}{M} \tag{4.2}$$

where M is the total number of modules the robot has.

In the turning gait, the robot follows a circular path with a radius of R, as shown in Figure 4.13. Assume there is no vertical displacement in the pitch grouped modules. From the top view of the robot, it is found that the radius R is related to the parameter A_y and the module length L. According to geometrical relationship, R can be approximated by :

$$R \approx \frac{L}{sin(\frac{A_y}{2})} \tag{4.3}$$

A simulated example is shown in Figure 4.14, with control parameters $A_p = 20$, $\tau = 0.4$ and $\alpha = 0.18$ in the pitch group, and control parameters $A_y = 10$, $\tau = 0.4$ and $\beta = 0$ in the yaw group. According to Figure 3.10, a phase difference of 90° is produced among oscillators in the pitch group, which results in a travelling wave in the vertical plane. In addition, due to the maintenance activity in the yaw group, each yaw module bents 10° right, forming a fix circular body wave. As a result, the simulated limbless robot moves along a circular arc in a clockwise direction.

Figure 4.15 illustrates the trajectory of the turning gait. Strictly speaking, the robot is not fully forward along its body arc. On the one hand, the linear movement and the bending behavior force the robot to follow the circular path. On the other hand, because the propulsion at the rear is not in the same direction as the one at the head, the robot has a slight rotation around the center of mass during the movement. That is why the center of mass of the robot is on the circular path while the other parts of the robot are not.

Given $A_y = 10$ and $L = 0.072m$ (see details in Table 4.1), the radius of the circular path is obtained according to equation 4.3. In this situation the robot

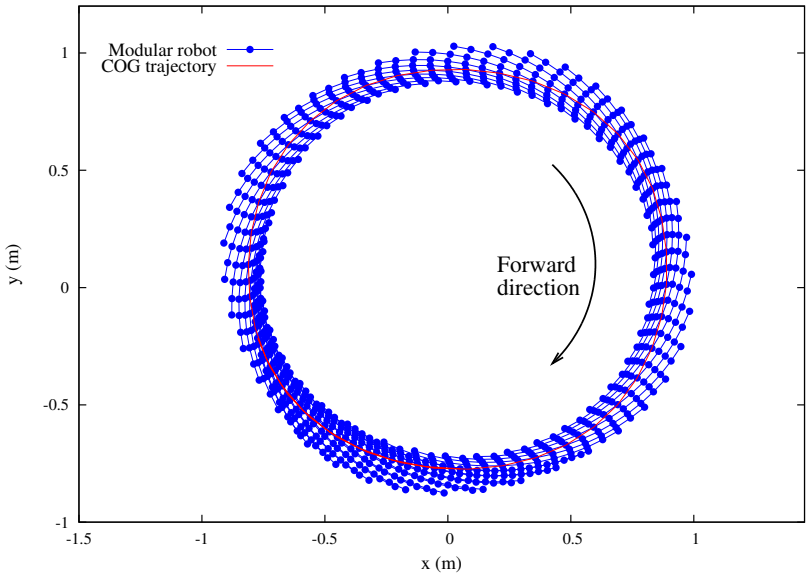


Figure 4.15: The trajectory of the turning gait.

should have a circular shape with a radius of $0.83m$. From Figure 4.15, it is found that the radius obtained through simulation is consistent with the theoretical value.

From the example, it shows that the turning circuit is effective in generating the turning gait.

4.2.3.2 Speed analysis

As mentioned in the previous section, the turning gait comprises a forward motion and a bending behavior. Since the bending behavior is only responsible for the change of forward direction, the forward motion dominates the forward speed of the turning gait. Therefore, of more concern for the speed analysis is how the control paramters A, τ, α and β in the chained inhibitory CPG circuit affect the speed of turning gait. Similar to the analysis of the sidewinding gait, we only examine A in range of (0, 60] and make intensive sampling when α is small. In addition, β is ignored since it does not participate in the forward motion generation.

Figure 4.16, 4.17, and 4.18 illustrate the variation of the forward motion speed with respect to A, τ and α. The simulated result shows that each of the parameters has a nearly linear relationship to the forward speed. The speed increases with the growth of A and α, while decreases with the growth of τ.

Parameters A, τ and α control the speed of the motion from three independent aspects. For parameter A, increasing A increases the height but decreases the length of a complete body wave. Since a step size per cycle is equal to the difference between the length of modules forming a complete body wave and the length of a complete body wave, increasing A finally increases the step size per cycle.

For parameter τ, unlike the sidewiding gait, in which parameter τ does not af-

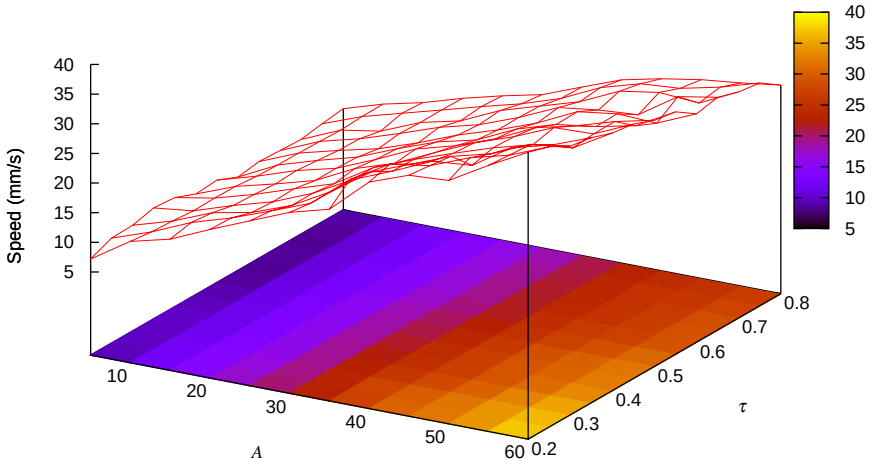


Figure 4.16: Turning speed variation with respect to A and τ, where $\alpha = 0.18$.

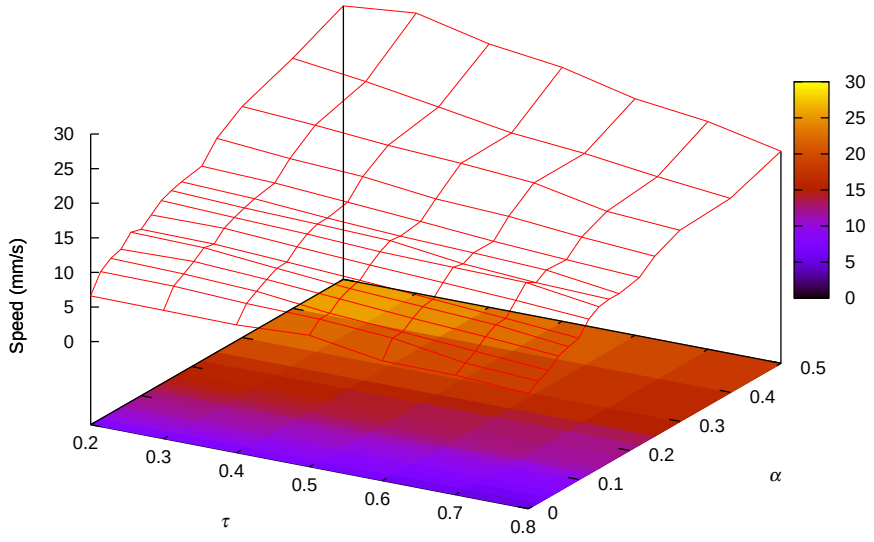


Figure 4.17: Turning speed variation with respect to τ and α, where $A = 20$.

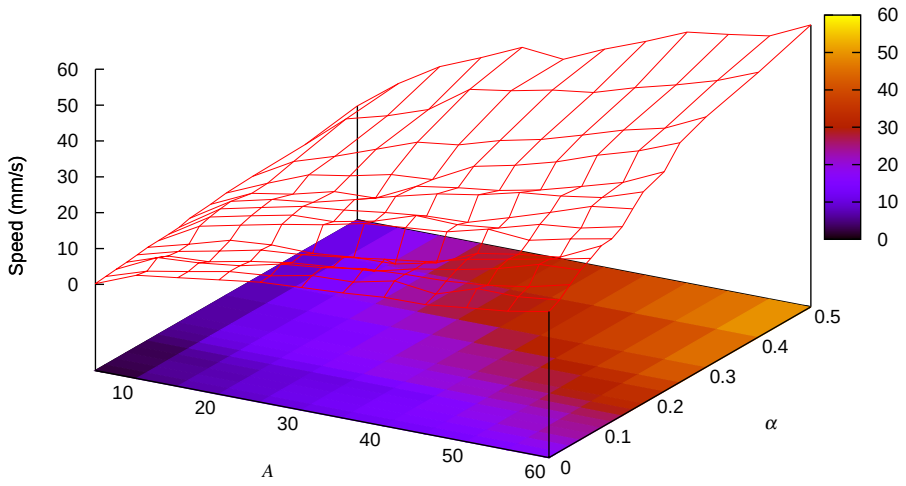


Figure 4.18: Turning speed variation with respect to A and α, where $\tau = 0.4$.

fect too much the speed due to the coordination between the pitch and yaw group, τ works on the modulation of the forward speed. τ determines pitch modules' rotational speed (see details in 3.11(b)). Increasing τ increases the period of oscillation, slowing down the rotational speed of individual pitch modules. Correspondingly, the forward speed generated with pitch modules decreases.

For parameter α, it determines the number of modules in a complete body waves. Increasing α decreases the phase difference among the modules in the pitch group (see details in 3.11(c)). Correspondingly, the number of modlues that form a complete body wave increases. Therefore, as the difference between the length of modules forming a complete body wave and the length of a complete body wave, the step size per cycle increases. However, α is not acceptable in the whole range of [0, 1]. A high value of α will reduce the phase difference as well as the number of the body waves, which makes it easy for the robot to get stuck due to the lack of torque to maintain the body shape. Therefore, α is limited in the range of [0, 0.5] in this simulation.

4.2.4 Flapping gait

Flapping is another sideways type of locomotion. A limbless robot moves laterally like the way a rower in a boat, who swings oars at both sides at the same time so as to propel the boat forward. As a type of in-phase motion, flapping does not exist in nature but is useful in limbless robot motion. It could help limbless robots to deal with complicated terrains, such as avoiding obstacles and climbing over slopes.

4.2.4.1 Gait design

Flapping circuit is a special cyclic inhibitory CPG circuit. As shown in Figure 4.19, oscillators in the pitch and yaw groups are mutually inhibited by their neighboring oscillators. Besides, oscillators at both ends in one group are also mutually inhib-

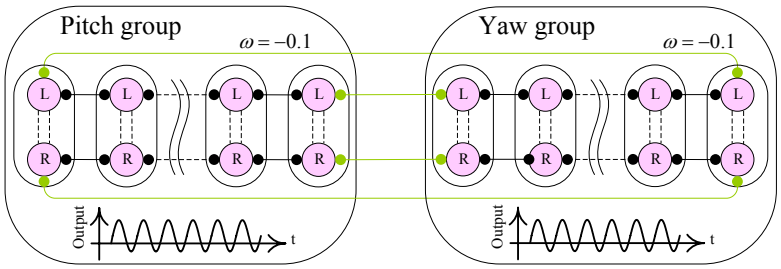


Figure 4.19: Flapping circuit. The pitch and yaw groups together comprise a cyclic inhibitory CPG circuit, forming a flapping circuit.

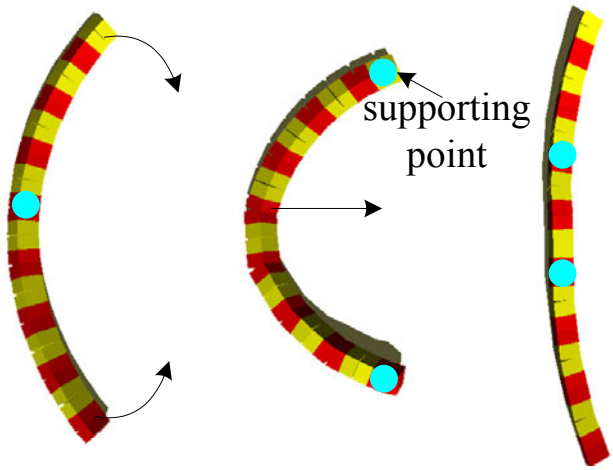


Figure 4.20: The simulation of the flapping gait.

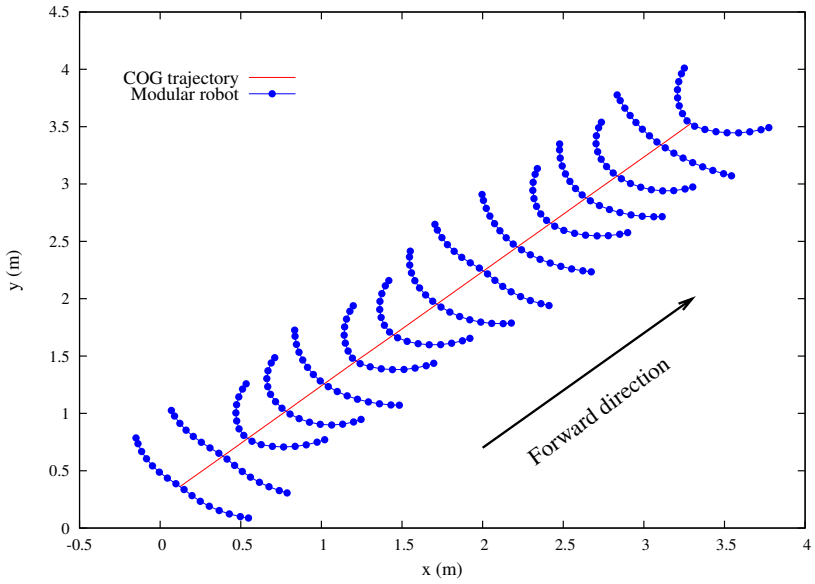


Figure 4.21: The trajectory of the flapping gait.

ited by the oscillators at both ends in the other group, forming a complete cyclic inhibitory CPG circuit. The purpose of the flapping circuit is to generate synchronized oscillation, so as to achieve in-phase motion on the robot. Therefore, the two groups are required not only to have the same amplitude and frequency, but also to maitain the inhibition with a uniform synaptic weight of -0.1.

Furthermore, in order to generate lateral movement, an additional offset needs to be applied on all the oscillators in the flapping circuit. As mentioned in Section 3.5, an oscillatory offset occurs when assigning opposite values of β to the two motoneurons (MNs) in each oscillator. Thus, all the modules in the pitch and yaw groups would produce asymmetric rotation, resulting in the robot always bending at one direction. Note that if the sign of β on the two MNs in each oscillator exchanges, an opposite oscillatory offset would be generated and finally reverses the flapping direction. As a result of in-phase oscillation and offset modulation, the robot could generate lateral movement towards the bending direction.

Figure 4.20 shows a simulated example with control parameters $A = 13$, $\tau = 0.4$ and $\beta = 0.7$ in the cyclic inhibitory CPG circuit. First, because of in-phase oscillation, the simulated limbless robot swings its ends forward with the growth of module angles. The center of the body is anchored during this movement. As far as the module angles reach maximum value, both ends come down to get into contact with the ground. After that, with the decrease of module angles, the robot begins to drag the center of the body forward with both ends anchoring on the ground. Due to offset modulation, the robot cannot make reverse bending. Therefore, when module angles reach the minimum value, the robot nearly resumes to a straight line. By using these steps successively, the robot succeeds to flap forward. Figure 4.21 is the trajectory of the flapping gait. It shows that flapping gait is also somewhat similar to the motion of a butterfly stroke.

From the example, it shows that the flapping circuit is feasible to generate flapping movement.

4.2.4.2 Speed analysis

As discussed in Section 3.5, there are three tunable parameters A, τ and β in the cyclic inhibitory CPG circuit that control the amplitude, the period and the offset of the synchronized oscillation. Although τ can affect the speed of individual modules, the forward speed is more dependent on the coordination between the pitch and yaw grouped modules. Therefore the analysis of τ is omitted here.

In the flapping gait, parameters A and β determine the body shape of the robot. Due to the in-phase oscillation and offset modulation, the robot would reach maximun curvature when all the modules achieve maximum angles. Combining equations 3.33, 3.35 and 3.37, the maximum angle a module can reach is:

$$\begin{aligned} max\_angle &= amplitude + |offset| \\ &= A \cdot (0.5 + |\beta|) < 90 \end{aligned} \tag{4.4}$$

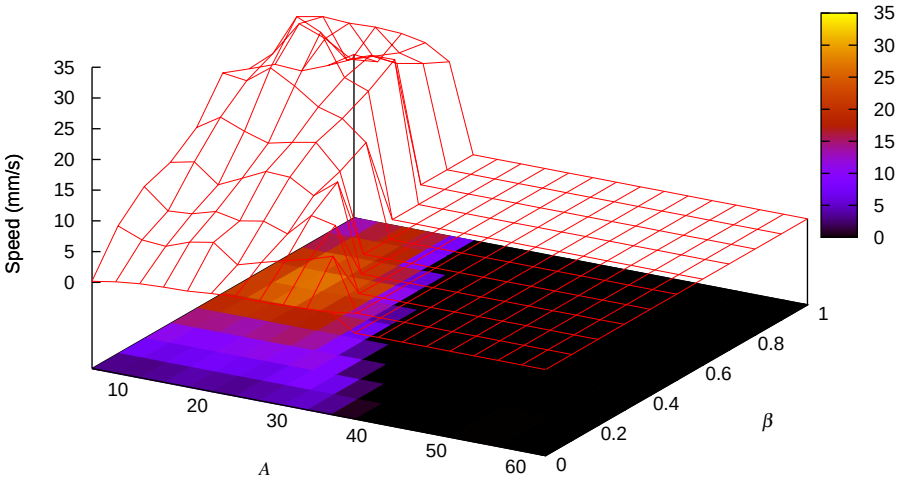


Figure 4.22: Flapping speed variation with respect to A and β.

To avoid collision occuring at both ends of the robot, the robot is prevented from forming a ring. Considering the robot is not in a flat plane when it forms a ring, a constraint can only be approximated to:

$$max\_angle \cdot M < 360 \tag{4.5}$$

where M is the total number of modules in the robot. According to equations 4.4 and 4.5, the constraint of A is obtained:

$$\begin{cases} A < \frac{90}{0.5+|\beta|} & if\ M \leq 4 \\ A < \frac{360}{M \cdot (0.5+|\beta|)} & otherwise \end{cases} \tag{4.6}$$

Figure 4.22 shows the speed variation of the flapping gait with respect to parameters A and β. For parameter A, since it only determines the degree of bending during the movement, it has slight influence on the speed of flapping. The simulation result verifies that a high value of A can cause both ends of the robot to collide with each other, which stops the robot from flapping, as shown in the black region of the figure. For parameter β, the measured range shrinks by half, i.e. from 0 to 1. This is because opposite value of β only changes the flapping direction, but does not affect the forward speed. The result shows that a low value of β fails to generate efficient lateral movement and makes the robot flap in situ. With the growth of β, the forward speed increases. However, as β closes to the maximum value, excessive offset aggravates the internal forces among modules, which finally leads to a more and more toilsome flapping movement. From the figure, an effective flapping gait occurs when A is in range of [10, 20] while β in range of [0.5, 0.8].

4.2.5 Summary

Through simulations, the speed variation with respect to parameters A, τ, α and β for these four limbless locomotion patterns is investigated. It is noted that if

Table 4.2: Parameters for fast limbless locomotion

Gait	Pitch and yaw groups			
	τ	A	α	$\lvert\beta\rvert$
Side winding	[0.4, 0.6]	[15, 35]	[0.6, 1]	0
Rolling	[0.2, 0.4]	[5, 25]	–	0
Turning	[0.2, 0.5]	[30, 60]	[0.2, 0.5]	0
Flapping	[0.2, 0.8]	[10, 20]	–	[0.5, 0.8]

these control parameters is set in a proper range, the generated limbless gait would have a relatively higher speed. Table 4.2 summarizes the acceptable range of these parameters for achieving such fast limbless locomotion.

4.3 On-site experiment

In order to verify the effectiveness of the CPG model, four types of limbless gaits, namely sidewinding, rolling, turning and flapping are realized. A pitch-yaw connected modular robot with five GZ-I modules is constructed as the test bed of this experiment. The GZ-I module is one of our mechanical prototypes that has the small dimension and the flexible connecting capability (Zhang et al., 2008). It weighs around 0.15 kg and has a length×width×height of $72mm \times 56mm \times 56mm$. The GZ-I module is equipped with a RC servo (Futaba s3003) as the actuator of the module, which provides a maximum speed of 1.45 rad/s and a maximum torque of 0.314 Nm. By actuating the servo, the GZ-I module can rotate in one degree of freedom within $\pm 90°$. In addition, a MSCC20(B) servo controller is employed to control all the servo motors on the robot.

The modular robot is controlled by a PC via a cable. The control is an open loop. It works as follows: First, the desired angle of each module is continuously calculated online by the PC. Then, all the desired angles are encoded as a command and sent to the servo controller with an empirical sampling time period of 80 ms. Finally, the servo controller decodes and executes the command, driving the robot to achive the desired posture. By using such real-time CPG control, the four limbless locomotion patterns are realized. The control parameters for these locomotion patterns are given in Table 4.3.

Figure 4.23 shows a series of pictures taken from a video recorded during the locomotion experiment. Figure 4.23(a) shows a side winding locomotion, where the robot moves laterally and drives the robot body one step forward per cycle. The rolling gait, as shown in Figure 4.23(b), is implemented by the robot rotating along its body axis. It should be noted that the control parameters are tuned to avoid any collision between robot joints. Figure 4.23(c) shows the robot implementing a turning gait, with the pitch modules generating the forward motion and the yaw

Table 4.3: Parameters for on-site gait generation

Gait	Pitch group				Yaw group			
	τ	A	α	$\lvert\beta\rvert$	τ	A	α	$\lvert\beta\rvert$
Side winding	0.4	20	0.8	0	0.4	20	0.8	0
Rolling	0.6	30	–	0	0.6	30	–	0
Turning	0.4	25	0.05	0	0.4	20	–	0
Flapping	0.6	15	–	0.6	0.6	15	–	0.6

modules generating bending behavior. In Figure 4.23(d), the robot swings both ends forward first and then lifts and moves the body center one step forward to accomplish a flapping gait.

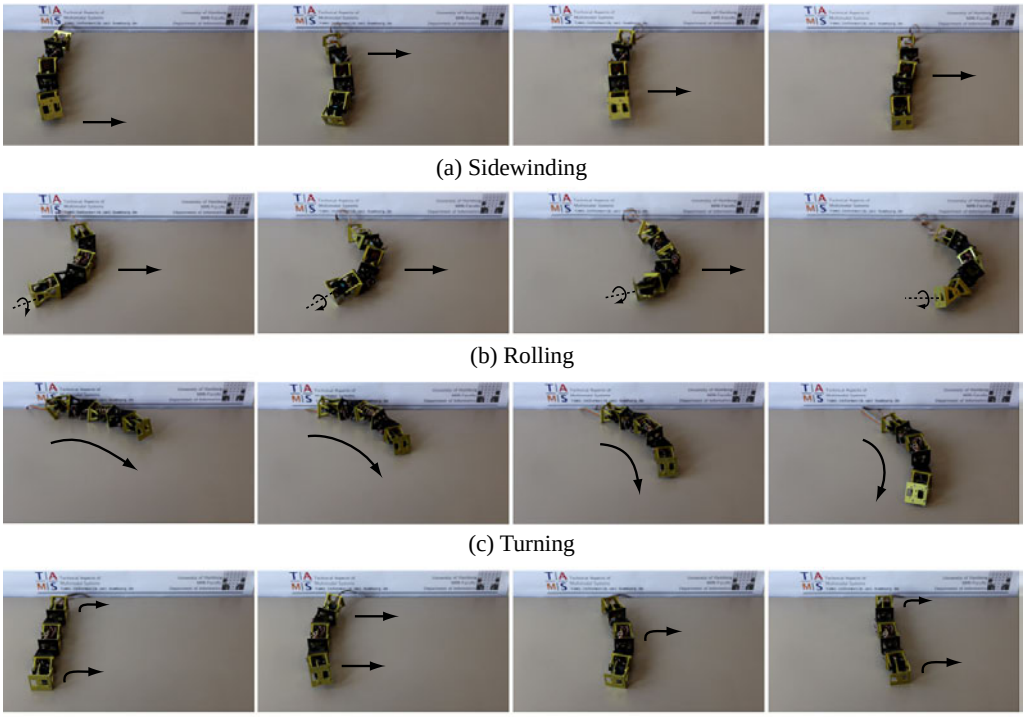
Based on the experimental results, we conclude that the proposed CPG model is effective in generating limbless locomotion patterns.

4.4 Summary

This chapter has applied the CPG model presented in Chapter 3 to limbless robots. From the configuration point of view, a limbless robot has one pitch grouped modules and one yaw grouped modules, which allows us to easily and systematically combine the CPG circuits with each of the groups. For the gait design, two CPG circuits that control the two groups of modules are well connected, so that a CPG circuit not only plays a role in generating rhythmic movement on one grouped modules, but also takes effect on coordinating the motion between the other grouped modules. Once the control parameters in the two CPG circuits are determined, a meaningful limbless locomotion pattern would be generated.

Based on the dynamic properties of the CPG model, four types of limbless gaits have been realized. For the sidewinding gait, taking advantages of two travelling waves along the two grouped modules and a phase difference between them, the sidewiding circuit enables the limbless robot to move sideways. For the rolling gait, the rolling circuit produces synchronization activity among the two grouped modules and a phase difference between them, which manages to make the robot roll along its body axis. For the turning gait, a travelling wave along the pitch group and a maintenance activity in the yaw group generated by the turning circuit drive the robot to turn around during forward motion. For the flapping gait, the flapping circuit produces unified synchronization activity and offset modulation among the two grouped modules, which causes the robot to move laterally like a butterfly stroke.

The four types of limbless gaits have been simulated in the ODE environment. Each gait is illustrated by an example as well as its trajectory. In addition, how the locomotion speed of each gait varies with respect to the tunable parameters in



(a) Sidewinding

(b) Rolling

(c) Turning

(d) Flapping

Figure 4.23: On-site gait experiment. Four limbless gaits are implemented using 5 pitch-yaw connected modules.

the CPG circuits has been investigated. The acceptable range of these parameters for realizing fast limbless gaits have also been summarized. In the end, on-site experiment shows the effectiveness of the proposed method in generating limbless locomotion patterns.

Chapter 5

Design of Sensory Reflex Mechanism

Contents

5.1 Introduction

Biological investigations show that an animal's movement consists of inherent locomotion patterns and reflexive actions. The inherent locomotion patterns generated by the CPGs enable the animal to move around, while the reflexive actions help them to simple and quick respond to external triggers. The reflex reactions are essential for survival in dangerous situation. They are an involuntary instinct that react when the body of the animal needs to regulate itself.

In the field of robotics, as the lowest level of sensor based response, the concept of reflex is widely used and applied for the control of robots. Espenschied et al. (1996) incorporated biological principles into biomechanics, and implemented the reflex control on a hexapod robot. They realized three types of postural reflexes. The first one is a stepping reflex. It is developed to re-establish a stable posture after the pertubations of a leg of the robot, which helpes the robot's leg to lift and move back to the center of its range of movement. The second is an elevator reflex. It deals with the case of obstacles. If the robot's leg encounters an obstacle, it reverses the motion and lifts higher before swinging forward again. Searching reflex is another reflex mechanism that is used if the robot's leg loses contact with the substrate. In this case, it searches a foothold by moving the end of the leg in circles with an increasing radius. As a result of the three reflexes, the robot could effectively deal with irregular terrain with no prior knowledge.

Huang and Nakamura (2005) proposed a control system with sensory reflex control for humanoid walking. They used zero moment point (ZMP) based control method for biped locomotion generation and combined the sensory reflex in case

of unexpected sudden events. The sensory reflex is defined as the additional joint angle appended to the joint angle specified by the dynamic pattern. If there are unexpected factors, the value of the sensory reflex becomes non-zero, and modifies the pattern of biped locomotion. After the unexpected event, the value of the sensory reflex gradually goes back to zero, eliminating the influence on the biped locomotion. Three independent sensory reflexes, namely the ZMP reflex, the landing-phase reflex and the body-posture reflex are developed to cope with different unexpected situations. With the help of dynamic pattern and these sensory reflexes, the robot could walk in complex environments.

Spenneberg and Kirchner (2007) proposed a hybrid bio-inspired approach that combines posture control with CPG-based control and reflex control for controlling locomotion in multipods. They used Bezier curves to describe a trajectory in the joint angle space. A simple reflex model is defined as an input function, an activation function and a response function. To control the posture of a joint, they integrated the ability to apply an offset to the rhythmic patterns. Thus reflexes take influence on the offset via their response function. In addition, a tumbling correction reflex is implemented. When the current of the thoracic joint exceeds a threshold, the reflex gets triggered and overwrites the signals from the CPG. Correspondingly, the reflex function moves the leg of the robot backward and up, and then moves forward again. By using this hybrid approach the robot is tested to be able to move through various terrains, such as rock fields, asphalt, sand, gravel and grass.

Kimura et al. implemented adaptive dynamic walking of a quadruped robot on natural ground based on biological concepts (Fukuoka and Kimura, 2003; Kimura et al., 2007). They defined a reflex as joint angle generation based on sensory information and a response as CPG phase modulation. In order to realize a self-contained quadruped walk of the robot on natural terrains, several new types of reflexes, including a flexor reflex, a sideways stepping reflex, a corrective stepping reflex/response and a crossed flexor reflex, are developed. They are all satisfying the necessary conditions of stable walking and would become active if their conditions were satisfied. As a result, the effectiveness of these reflexes is validated by indoor and outdoor experiments.

Besides legged robots, there are many other types of robots referred to the reflex control, such as manipulators (Kamata et al., 2007; Sakurai et al., 2010) and mobile robots (Chatterjee and Matsuno, 2001; Vaidyanathan et al., 2012). Nevertheless, the study of reflex control on limbless robots is seldom seen. Furthermore, the aforementioned methods for integrating sensory reflex mechanisms are somewhat stiff and unnatural. To develop reflex mechanisms on limbless robots in a natural manner, this chapter explores designing the fundamental reflex mechanisms combining the proposed CPG model for quick responses to unexpected sudden events.

5.2 Sensory reflex mechanism

As mentioned in Chapter 3, since the proposed CPG model is inspired by lampreys and developed at the neuronal level, it is possible to further integrate sensory reflex

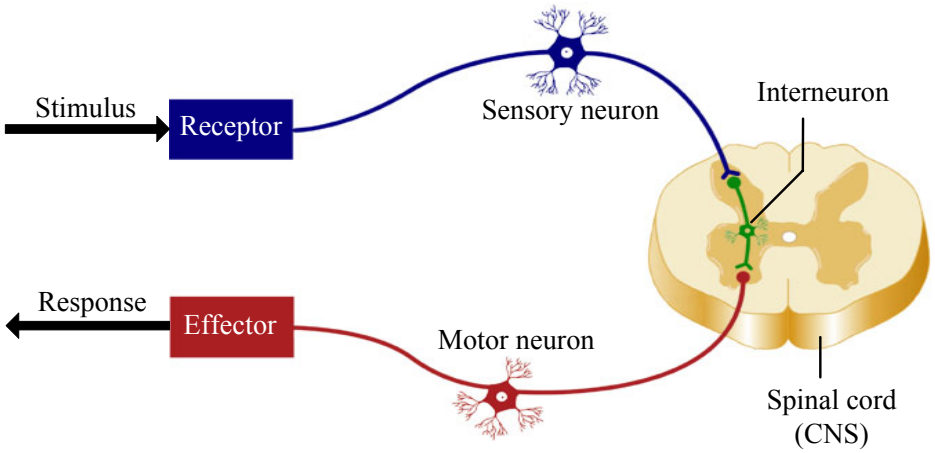


Figure 5.1: Reflex arc. It is a particular pathway followed by nerve impulses to produce a reflex action. It consists of a receptor, a sensory neuron, a reflex center (spinal cord), a motor neuron and an effector.

mechanisms into the CPG model, enabling the CPG model to quick respond to external stimuli. Here, we investigate the sensory reflex mechanisms originating from biological studies and propose a similar method for the integration of sensory reflex mechanisms into the CPG model.

5.2.1 Reflex arc

In neuroanatomy, it has been found that the response of an animal to external stimuli is coordinated by its central nervous system (CNS) (Brodal, 1998). The CNS contains the majority of the nervous system and consists of the brain and the spinal cord. It works with the help of nerves or neurons, which conduct the signals or impulses in the nervous system.

The number of neurons involved in a task partially determines the speed of information processing. For example, when an animal tries to identify and reach for a fresh leaf, it has to make perception and cognition before the final decision. This is a conscious stimulus-response procedure, which involves multiple brain areas and many neuron connections. Therefore, the elapsed time from perception to the response behavior is relatively long. But when the safety of an animal demands a very quick response, such as rapid withdrawal from dangerous stimuli, the signals may directly generate instant and unconcious action. This is a reflex action. Reflex reactions in animals are usually controlled by the reflex arc, which involves only a few neurons.

As shown in Figure 5.1, a reflex arc is a neural pathway that makes such a fast, automatic response possible (Creed, 1972; Ganong, 2009). In the reflex arc, the entire signaling sequence, from sensing, processing to response, is handled by the spinal cord without the need to wait for instructions from the higher brain regions. First, a receptor detects a painful stimulus and relay this information along an

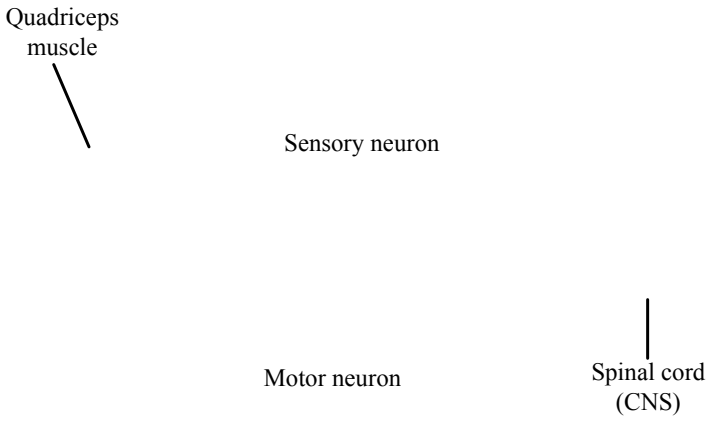


Figure 5.2: Knee jerk reflex. It is a reflex contraction of the quadriceps muscle resulting from striking below the knee, which gives rise to a sudden extension of the leg. There is no interneuron in the spinal cord involved in the pathway.

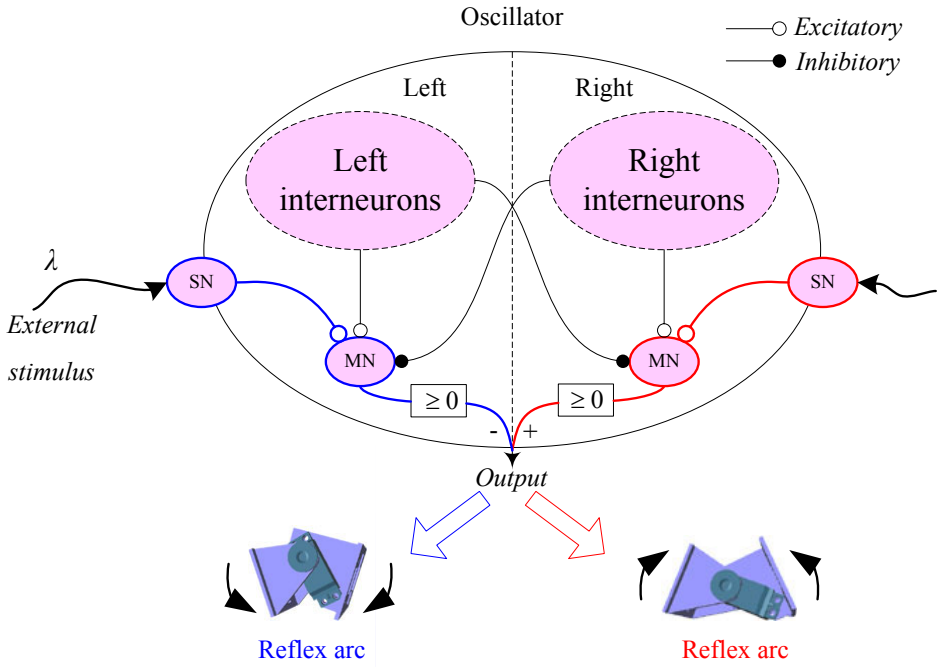


Figure 5.3: Border reflex integration. The "interneurons" of the oscillator indicates the CIN, LIN and EIN for simplicity. Two additional SNs are employed on each side of the oscillator. A SN together with an ipsilateral MN forms a monosynaptic reflex arc. Border oscillators in the CPG circuit are equipped with this type of reflex.

afferent fiber. The information is carried all the way from the receptor to the interneurons within the spinal cord by a single sensory neuron without any other intervening synaptic connections. These interneurons, in turn, pass the information along an efferent fiber to a motor neuron. As the efferent fiber ends in an effector, the effector is stimulated by the activated motor neuron and its reaction is the response.

Nevertheless, CNS involvement of the interneurons in the spinal cord along the pathway is not always necessary. A reflex arc may only consist of a sensory neuron and a motor neuron, namely, the monosynaptic reflex, which allows reflex actions to occur relatively quickly by activating the two neurons without the delay of routing signals through the CNS. The knee jerk reflex is the most common and classic example of the monosynaptic reflex arc (Weiner and Shin, 2010). As shown in Figure 5.2, it is tested by striking just below the knee with a tendon hammer. From there, a stretch receptor conducts an efferent impulse to the sensory neuron. There are no CNS interneurons in the pathway taking part in the reflex arc. Instead the sensory neuron directly sends the nerve impulse to a motor neuron. The result of this motor neuron activity is contraction of the quadriceps muscle, causing the lower leg to suddenly jerk forward.

5.2.2 Sensory neuron integration

In the field of biology, lampreys are found to have sensory neurons (SNs) that are located within the spinal cord and activated by external stimuli (Grillner et al., 1991; Matthews, 2001). These SNs are found on each side of the spinal cord and are considered as part of the reflex circuit.

To generate this kind of sensory reflex mechanisms, the SNs and the concept of the reflex arc are both added into the CPG model. Two types of sensory reflex, called border reflex and body reflex respectively, are designed based on the monosynaptic reflex arc.

Figure 5.3 illustrates the border reflex with two monosynaptic reflex arcs integrated. To make the diagram clearer, the oscillator shown in detail in Figure 3.5 is lumped together on each side, with a substitution of “interneurons”. Each of the monosynaptic reflex arcs is made up of a SN and a motoneuron (MN) on the ipsilateral side. Because of the signal filtering on the MN, the excitatory synapse from the SN to the MN would exacerbate the external stimulus and thus help the CPG model to generate quick response. The border reflex is only applied to these oscillators on the border of the CPG circuit.

The body reflex has also two integrated monosynaptic reflex arcs, as shown in Figure 5.4. In contrast to the monosynaptic reflex arc in the border reflex, the monosynaptic reflex arc in the body reflex consists of a SN and a contralateral MN. This means the body reflex could generate opposite reflex behaviors compared to the border reflex. The body reflex is utilized by the non-boundary oscillators in the CPG circuit.

A simple reflex model is designed in Figure 5.5. A SN is directly correlated with

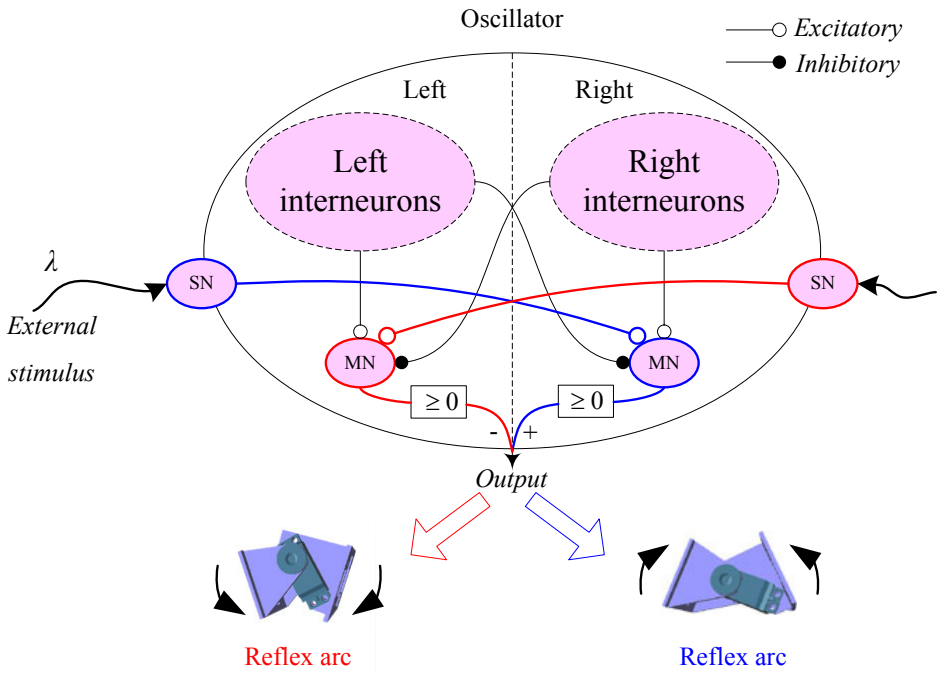


Figure 5.4: Body reflex integration. The monosynaptic reflex arc contains a SN and a MN on the contralateral side. The body reflex is applied to the internal oscillators in the CPG circuit.

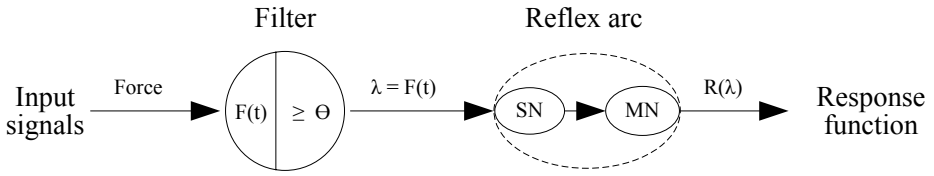


Figure 5.5: Reflex model. The reflex model deals with raw sensor data and generates a response function via the reflex arc.

sensory devices such as a force sensor. The processing flow of the reflex model is as follows: First, input signals that measured from the force sensor are handled by a filter. To avoid disturbance caused by sensor noise, the filter activates the reflex only if the measured force exceeds a pre-defined threshold. The reflex model takes the measured force as the afferent stimulus and sends it to the reflex arc. Then, the SN propagates the external stimulus along the reflex arc and serves to reinforce the MN. The MN further affects the output of the oscillator and finally generates the response behavior. In contrast to the canonical assumption that sensory reflexes are fixed reactions, the reflex model associates the response function with the value of the measured force, and thus it is able to generate non-fixed response behaviors.

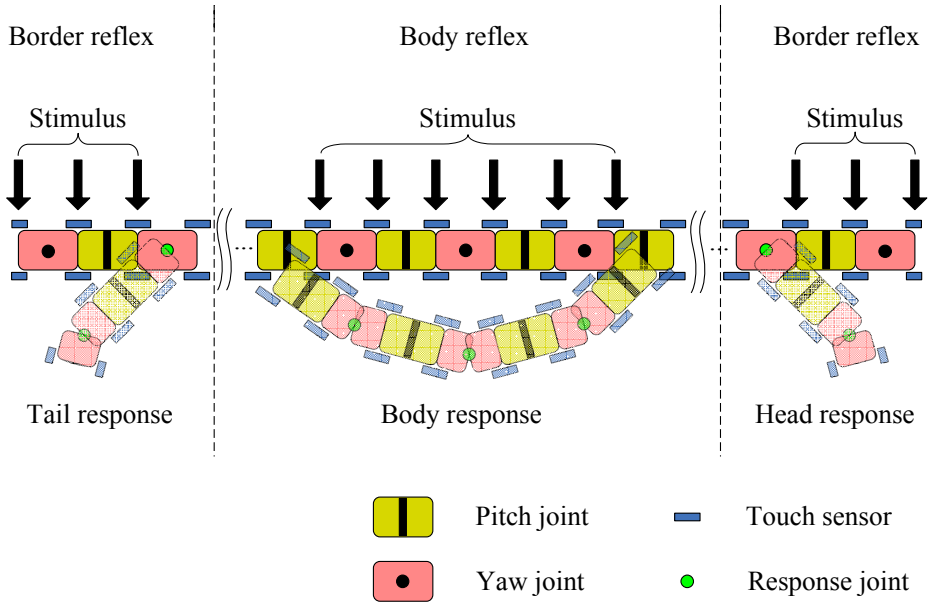


Figure 5.6: Reflex behavior. The two types of reflexes could generate two opposite reflex behaviors after external stimuli are afferent to the robot.

The dynamic of the SN is given by the following piecewise function:

$$\begin{cases} \tau_{response}\dot{x}_{\{SN\}i} = -x_{\{SN\}i} + p\lambda & if \quad \lambda > \Theta \\ \tau_{recover}\dot{x}_{\{SN\}i} = -x_{\{SN\}i} & otherwise \end{cases} \tag{5.1}$$

where $\tau_{respone}$ and $\tau_{recover}$ are time constants that control the response and recover speed of the state of the SN, respectively; p is a proportional constant and Θ is a threshold for activation; λ indicates the external stimulus, which equals to the absolute value of the force received by the touch sensor on the robot.

When the force λ on the touch sensor exceeds the pre-defined threshold Θ, the external stimulus $p\lambda$ is afferent and transmitted to one of the SNs. Then, the SN affects the output by means of the reflex arc, resulting in a quick reaction at the corresponding joint. If there is no stimuli afferent into the reflex model after the reaction, according to the second equation in 5.1, the SN will gradually eliminate the influence on the modified joint and finally recover it to the normal state. That completes the reflex response.

5.2.3 Response behaviors

As described above, the border reflex and the body reflex have opposite response behaviors due to the different composition of the monosynaptic reflex arcs. The two types of reflexes are both applied to the CPG circuit. It is supposed to apply the border reflex on the border oscillators of the CPG circuit, while to apply the body reflex on the non-boundary oscillators of the CPG circuit. Correspondingly, the

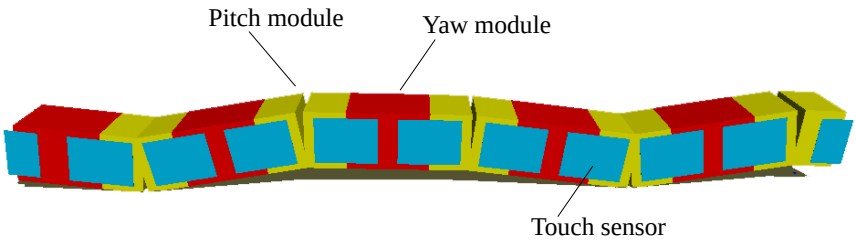


Figure 5.7: The simulated limbless robot with touch sensors on both sides.

response behaviors of a limbless robot on the border modules should be opposite to the response behaviors on the other modules of the robot. This would help the robot to generate resonable reflex responses.

Taking a pitch-yaw connected modular robot with touch sensors on both sides as an example, the top view of the robot is shown in Figure 5.6. Suppose the two types of reflexes are only applied to the yaw grouped modules. When external stimuli act on the border modules of the robot and activate their border reflexes, these border modules are designed to bend to the opposite direction compared to the direction of the afferent stimuli. In contrast, if external stimuli happen to the internal modules of the robot and activate their body reflexes, these modules are designed to respond in an opposite way, namely, to bend toward the direction of the afferent stimuli.

As a result of the two types of reflexes, the resulting response behaviors always try to make the robot bend correctly and get rid of external stimuli, so as to avoid potential damages to the robot.

5.3 Ball hitting experiment

This section examines the sensory reflex mechanisms developed in the previous section. In order to validate the feasibility of the reflex model, a ball hitting experiment is designed. Both simulation and on-site experiment are performed for the ball hitting experiment. On the one hand, in the ODE environment, a pitch-yaw connected modular robot mounted with force sensors on both sides of its body (in blue colour) is configured, as shown in Figure 5.7. On the other hand, a real pitch-yaw connected modular robot with five aluminium modules is constructed for on-site experiments. Each end of the aluminium module is surrounded by six touch switches. The specifications for the module are shown in Table 5.1.

We employ the turning gait (see details in Section 4.2.3) in this experiment, where the pitch joints generate the forward movement with gait parameters A=22, τ=0.4 and α=0.06, respectively, and the yaw joints with two types of reflexes integrated produce reflex turning behaviors under external stimuli. The speed of pitch modules is controllerd by a PD motor controller. It computes the speed as a function of the difference of the desired angle and the real angle plus the derivative

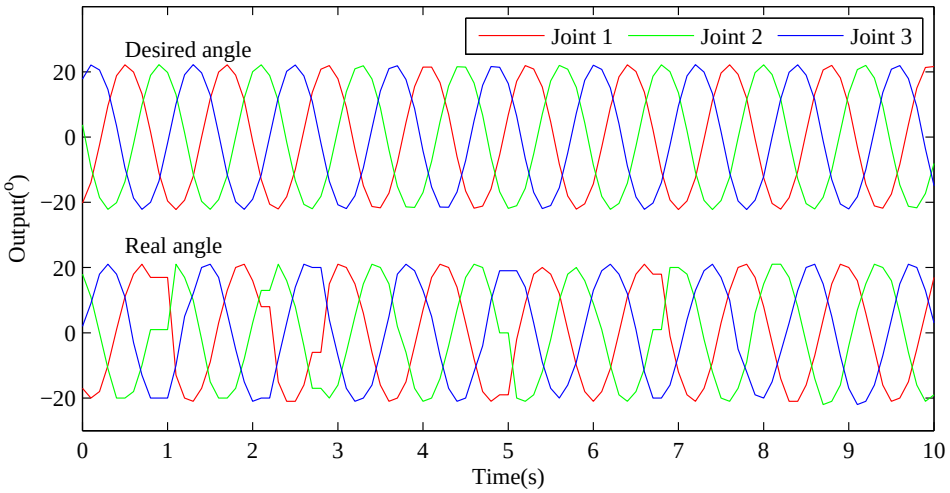


Figure 5.8: Variation of pitch joints. By using an empirical timer of 100 ms, the real angles are found to be able to follow the desired angles.

of them, which enables the real angle to follow the desired angle without losing accuracy, as shown in Figure 5.8.

The control of the robot is divided into two channels. Within an interval of 100 ms, the desired angles and the speed of each pitch module are calculated by a PC and sent to the robot via a bluetooth connection. Asynchronously, sensor information is gathered and sent back to the PC in every 50 ms. By separating the control channel and the sensing chanel, the robot succeeds to not only generate smooth movement, but also collect sensory information.

However, more interest lies in the response of the yaw joints rather than the pitch joints in the reflex experiment. Therefore we only focus on the yaw joints and their touch sensors in the following simulation and on-site experiment.

Table 5.1: Specification of the aluminum module

Specification	Value
Size (mm)	$90 \times 52 \times 52$
Weight (kg)	0.12
Actuator	RC servo motor (Futaba s3003)
Touch sensor	On-off touch switch (Omron D2MQ)
Position sensor	Built-in sensor of Futaba s3003
Comunication	Bluetooth

5.3.1 Simulation

As mentioned above, the ball hitting experiment is carried out to check whether the robot is able to respond correctly to external stimuli. A small ball, similar to the hammer in the knee jerk experiment, is utilised here as the source of external stimuli. A complete reflex process is shown in Figure 5.9(a)-(c). The procedure is as follows: The robot was suddenly hit on the head by the ball during forward motion, and it made a quick response by using the first yaw joint to turn to avoid the ball. When the ball no longer hit the robot, the turning behavior gradually disappeared and the robot recovered from the external perturbation and resumed its forward motion.

In the experiment, the activation threshold Θ is chosen low with a value of 1 N. The purpose is to make the reflex mechanisms sensitive to external stimuli. At the same time, in order to make a quick response and a slow recover behavior, the response speed $\tau_{response}$ is chosen low with a value of 0.4, but the recover speed $\tau_{recover}$ is chosen relative high with a value of 6. Furthermore, the proportional constant p is set to 2. This means the response angle has a theoritical maximum value that is twice as much as the value of measured force. However, due to a very short period time of collision in the experiment, the theoritical maximum value can not be reached. As a matter of fact, the actual maximum response angle is related to the amount of the afferent stimilus λ, the response speed $\tau_{response}$ and the period time of collision Δt, and can be approximately described as:

$$A_{max} \approx \frac{p\lambda}{\tau_{response}} \Delta t \tag{5.2}$$

Figure 5.9(d) illustrates the afferent stimulus as well as the response angle of the corresponding joint. Taking the second complete response as an example, it receives a stimulus with a value of 6.5 N, and lasts the bending response caused by collision about 320 time steps (correspond to 0.32 s in the real world). According to equation 5.2, the maximum response angle is about 10.4 degrees, which is consistent with the maximum value as shown in the figure. In addition, the recover period is about 14 times as long as the response period, which is close to the ratio between $\tau_{recover}$ and $\tau_{response}$.

5.3.2 On-site experiment

The on-site experiment also successfully exhibits the response and recovery behaviors, as shown in Figure 5.10(a)-(d). The only difference lies in that the real robot uses touch switches as touch sensors instead of force sensors. Thus, the dynamic of SN in equation 5.1 should be rewritten as:

$$\begin{cases} \tau_{response}\dot{x}_{\{SN\}i} = -x_{\{SN\}i} + \Delta t \cdot p\lambda_0 & \textit{if } \text{switch is pressed} \\ \tau_{recover}\dot{x}_{\{SN\}i} = -x_{\{SN\}i} & \textit{otherwise} \end{cases} \tag{5.3}$$

where λ_0 is a constant that indicates the afferent speed of the external stimulus (with a value of 15 N/s in this experiment) and Δt is a period of time which depends

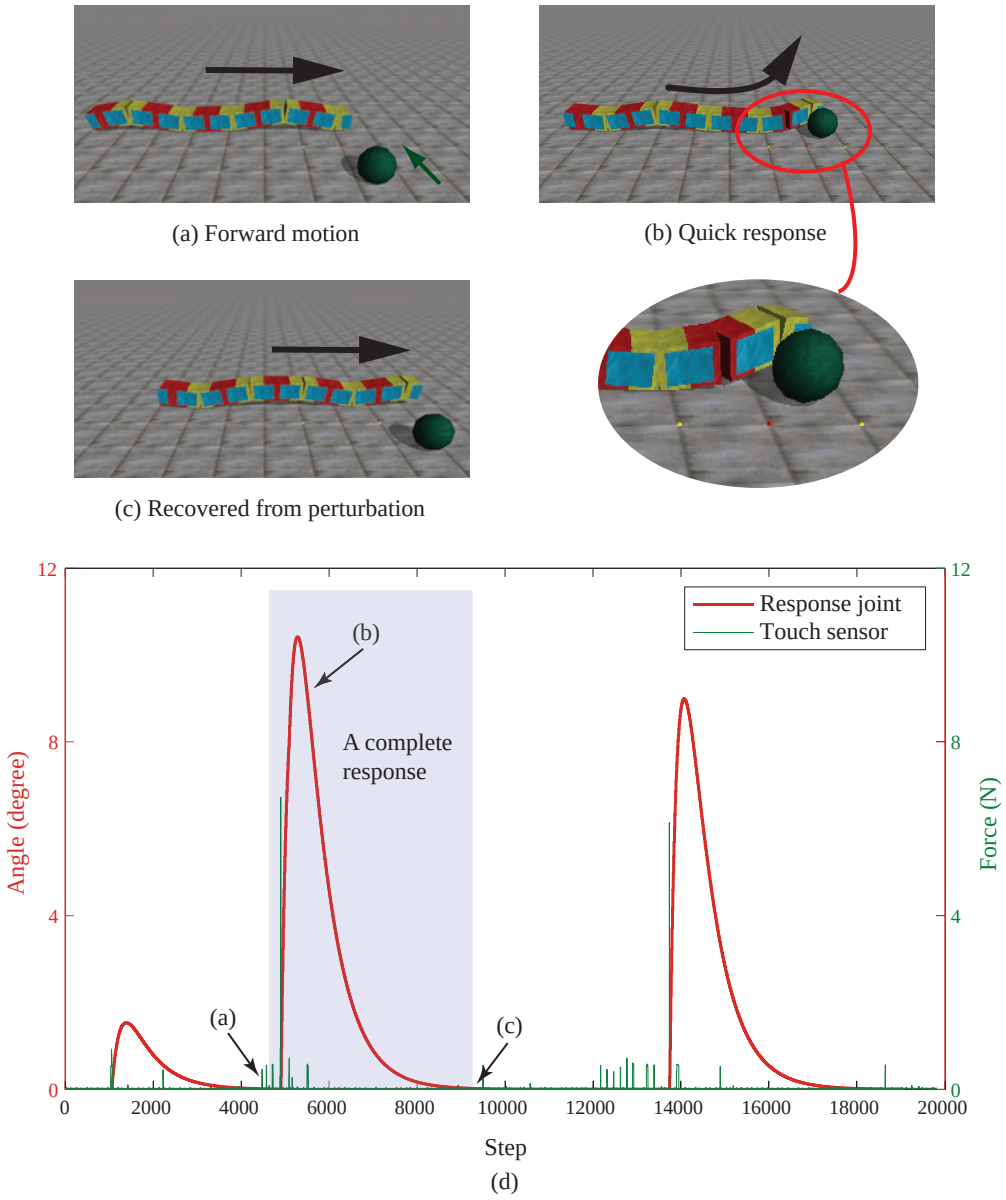


Figure 5.9: Ball hitting simulation. (a) A ball is hitting the robot in forward motion. (b) The force on the touch sensor over a threshold activates the afferent stimulus and makes the robot respond quickly. Note that the force is proportional to the maximum bending angle according to the equation 5.2. (c) The robot recovers since there is no further afferent stimulus. (d) The force on the touch sensor and the response angle of the corresponding joint.

on the state of the touch switch. Δt can be defined as a function of the difference between the two time points t and t_0:

$$\Delta t = t - t_0 \tag{5.4}$$

where t_0 represents the moment when the touch switch is pressed, while t represents the current time if the touch switch is still pressed. Note that the longer the duration of time that the touch switch is pressed, the larger the amount of external stimuli $\Delta t \cdot p\lambda_0$ that will be generated at the corresponding joint.

Accordingly, the actual maximum response angle can be rewritten as:

$$A_{max} \approx \frac{p\lambda_0}{\tau_{response}}(\Delta t)^2 \tag{5.5}$$

Figure 5.10(e) illustrates the relation between the state of the touch switch on the right side of the head and its corresponding response on the first yaw joint. At time 6.8 s, the touch switch on the right side of the head was hit by the ball. At the same time, an external stimulus, generated by equations 5.3 and 5.4, was afferent to the corresponding sensory neuron in the oscillator at the first yaw joint. Along the reflex arc, the output of the oscillator was forced to modulate to respond the external stimulus. The respone time lasted 0.5 s. According to the equation 5.5, the response angle can be achieved to a maximum value of 19 degrees. As long as there is no longer any perturbation, the oscillator resumes its maintenance activities. That is why from time 7.3 s to 15 s, the angle of the first yaw joint gradually got back to normal state. Compared to Figure 5.9(d), the result shows that the touch switch plays the same role as the force sensor in the reflex of external stimuli. Through the ball-hitting experiment, the feasibility of the integrated sensory reflex mechanisms is validated.

5.4 Corridor passing experiment

A more complicated scenario is an application of the reflex mechanisms, which requires the robot with a forward motion to pass through a narrow and winding corridor. Two response strategies are involved in this experiment. On the one hand, when a collision on the head or tail is detected, the border reflex is activated. External stimuli will be transmitted to its corresponding yaw joint and initiate an opposite turning behavior. On the other hand, if body collision occurs, the body reflex is enabled. External stimuli will cause the corresponding yaw joint to bend away from the collision point. Since too many collisions occur during the robot is crossing the corridor, only the touch sensors on the head and on the middle, as well as the response of the first yaw joint and the middle yaw joint, are considered.

5.4.1 Simulation

In the simulation, the corridor is designed to have a length of 2500 mm and a width of 130 mm, which is about 3.5 times as long and 2.5 times as wide as the robot

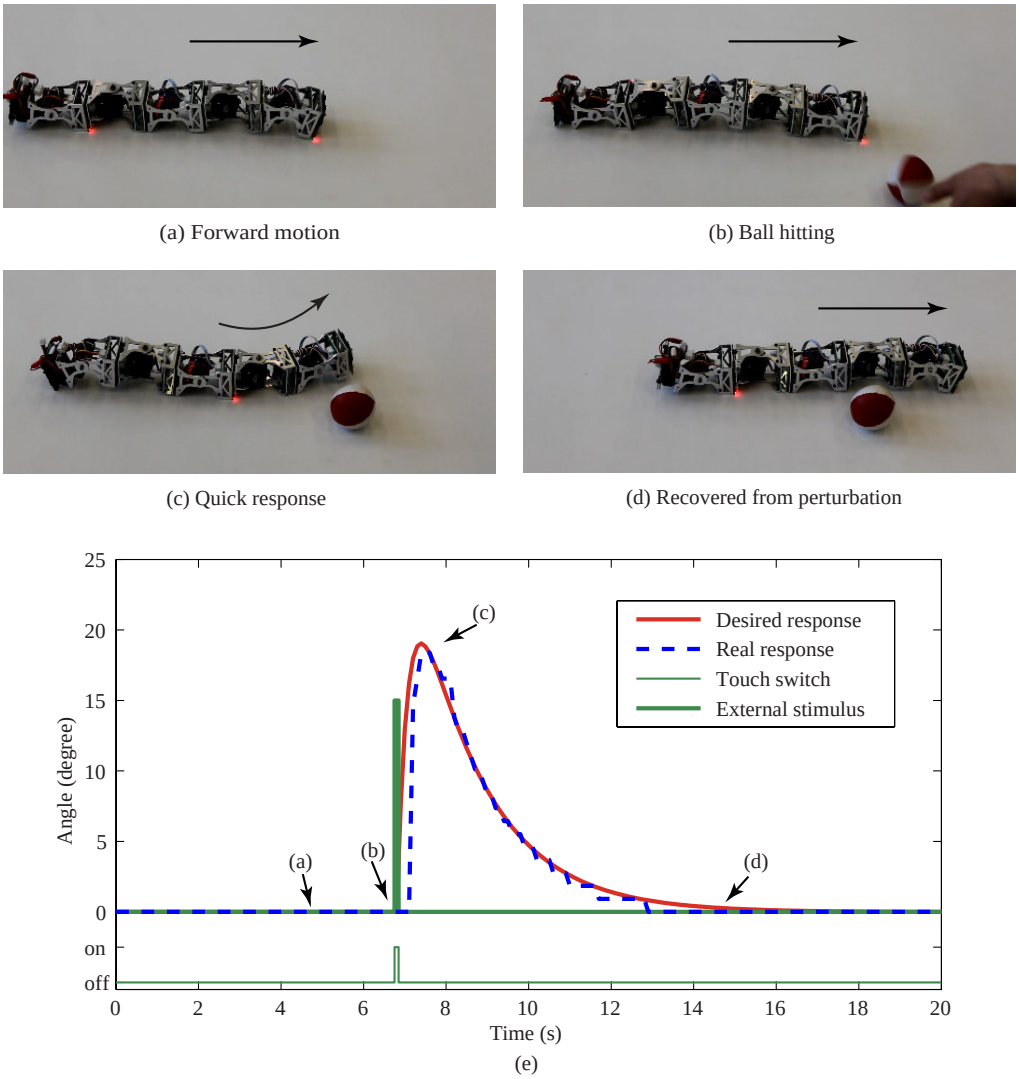


Figure 5.10: On-site ball-hitting experiment. (a)-(d) Scenes of the experiment. (e) The state of touch switch on the right side of the head and the response of the first yaw joint.

used in this experiment. Figure 5.11 shows the reflex response on the head of the robot. In the figure, the red points indicate the sampled head collision positions corresponding to the points in Figure 5.11 (b)-(e). Since the corridor is curved, inevitable collisions always occur on the head of the robot, especially when the robot moves at each corner in the corridor. To guide the robot to pass the corridor, the border reflex is applied on the head of the robot, which helps the robot to turn the head in the opposite direction to avoid further collisions.

Figure 5.11(f) illustrates the reaction of the head reflex, namely the variation of the first yaw joint over all the time steps. For the forces measured from touch sensors, a negative value indicates the force measured by the touch sensor on the

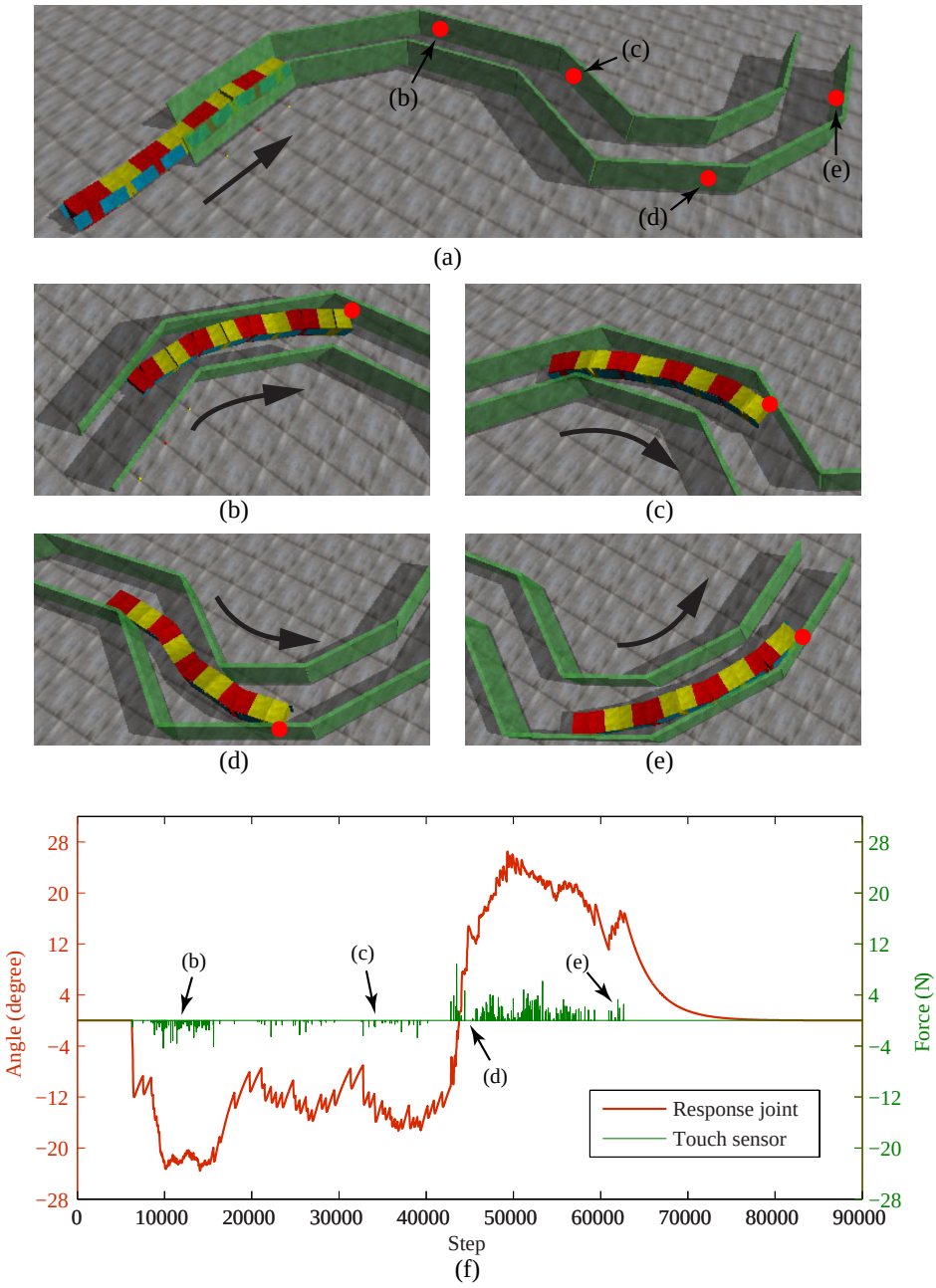


Figure 5.11: Border reflexes in the corridor passing simulation. (a) Overview of the corridor. (b)-(e) Different head turning reactions based on the border reflex. (f) The angle variation of the first yaw joint and the force detected by its corresponding touch sensors on the head.

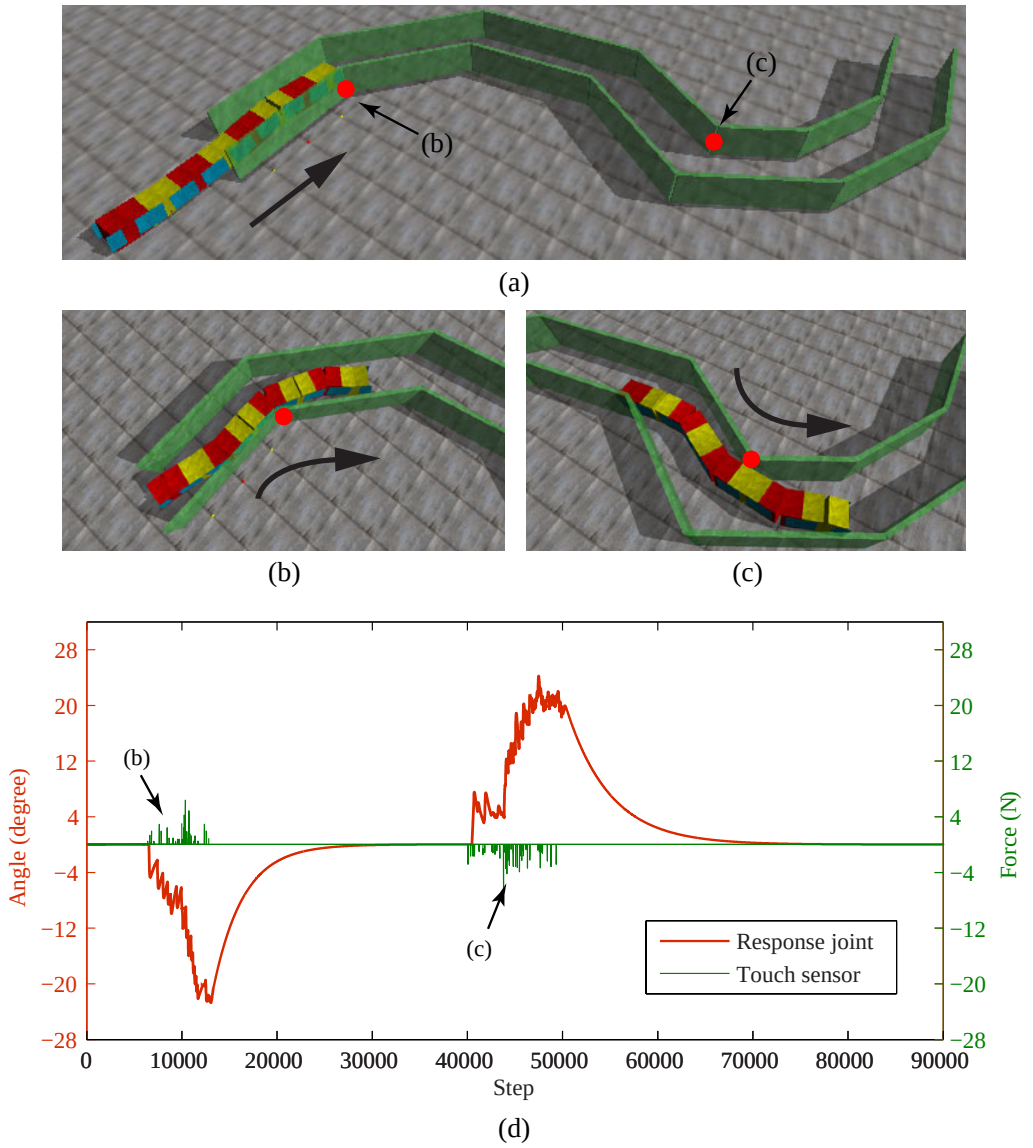


Figure 5.12: Body reflexes in the corridor passing simulation. (a) Overview of the corridor. (b)-(c) Different body turning reactions based on the body reflex. (d) The angle variation of the second yaw joint and the force detected by its corresponding touch sensors.

left side, while a positive value represents the force caused by the touch sensor on the right side. For the angle of the response yaw joint, a negative value implies a right turning of the module, while a positive value means a left turning of the module. The force detected by touch sensors on both sides not only affects the values of the bending angle, but also determines the bending direction. From this figure, it can be seen that the detected forces always result in response angles with the same sign. This means the response direction is opposite to external stimuli, which is consistent with the function of the border reflex.

Figure 5.12 shows the reaction on the body of the robot. Likewise, the red points shown in Figure 5.12(a) are the positions where the body collision occurs, which corresponds to the positions in Figure 5.12(b) and (c). The body reflexes appear much less than the border reflexes. This is because most of the collisions happen to the head of the robot. While body collisions happen only when the robot moves to a sharp corner and passes half of its body. On such a situation, collisions usually first occur on the head of the robot, resulting in border reflexes. Then the border reflexes change the turning of the head. Due to the momentum, the consequence of the head response also shifts the body of the robot, which finally leads to the body collisions.

Figure 5.12(d) illustrates the reaction of the body reflex, including the variation of the second yaw joint and its corresponding sensor information. During the body reflex, it can be found that the sign of the detected forces are opposite to the sign of the response angles. Compared to Figure 5.11(f), the response behavior of the body reflex is totally opposite to the response behavior of border reflex. That exactly complies to the expectation of the two types of reflexes.

Through the simulation, it verifies that the robot is able to pass through the corridor with the help of the two types of reflexes.

5.4.2 On-site experiment

A similar on-site experiment is carried out using the same response strategies, as shown in Figure 5.13(a)-(d). During the experiment, the robot responds quickly at the yaw joints where collisions are detected by their touch switches, allowing the robot to behave adaptively to the curving of the corridor. Furthermore, in a similar manner to the ball-hitting experiment, the yaw joints that had responded to external stimuli can swing back to their initial state gradually, which helps the robot to recover normal linear motion. By alternating between the response and recover states during the forward motion, the robot successfully passes through the corridor.

Due to the number of yaw modules on the real robot (one for the head reflex and the other for the tail reflex), the body reflex does not appear. Hence we only analyze the border reflex here. Figure 5.13(e) illustrates the state of touch switches on the head, as well as the reaction of the first yaw joint. Compared to Figure 5.11(f), the trend of their responses on the first yaw joint is found almost the same. During the first half of the experiment, both angles of the first yaw joint were negative, while

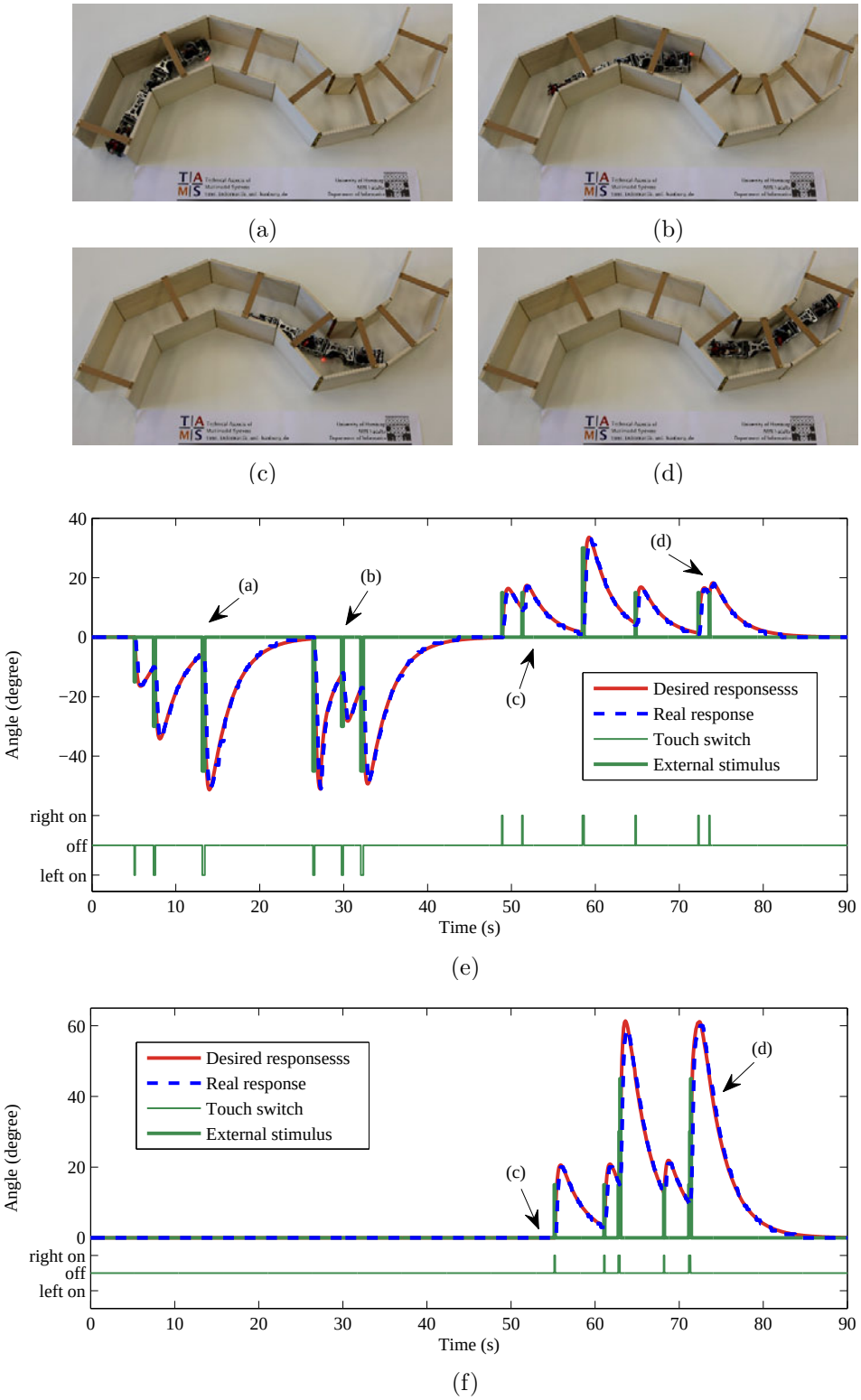


Figure 5.13: Border reflexes in on-site corridor passing experiment. (a)-(d) Scenes of the experiment. (e) Head reflex. (f) Tail reflex.

in the second half, they became positive. However, the number of reflexes occurring in Figure 5.11(f) is found much larger than that of reflexes in Figure 5.13(e). The reason is that the touch switch in the real world is less sensitive than the force sensor in simulation. Although the number of reflex decreases in the on-site experiment, the result shows that it does not inffluence the robot to cross through the corridor. Figure 5.13(f) shows the angle variation of the yaw joint at the tail side, together with the touch states detected by its corresponding touch switches. The tail reflex only occurs at the latter half of the experiment. It regulates the behavior of the yaw module at the tail side so that the robot manages to cross through the last corner of the corridor.

As a result of the on-site experiment, we validate the effectiveness of the reflex mechanisms in responding to external stimuli.

5.5 Summary

This chapter realizes the integration of sensory reflex mechanisms into the proposed CPG model. Through an investigation of reflex control in robotics, we find that although the concept of reflex has been widely applied to different types of robots, the application of reflex control on limbless robots is rarely seen. Also the ways for integrating reflex mechanisms are not so natural.

To develop natural reflex mechanisms, we realize them at the neuronal level that matches the level of the proposed CPG model. We study the pathway of reflex arc in the nervous system, and investigate the function of sensory neurons in lampreys. Because our CPG model is based on the lamprey's neural circuit, the sensory neurons are easily designed and integrated into the CPG model at the neuronal level. The reflex arcs that involve the sensory neurons are therefore further established, forming a simple reflex model.

A border reflex and a body reflex are developed and applied to the limbless robot at the border part and the body part separately. Due to the inverted structure of reflex arcs in the two types of reflexes, they can generate opposite response behaviors. This design helps the robot to respond reasonably to external stimuli. The feasibility is confirmed by simulation and on-site testing. In the ball hitting and corridor passing experiments, we test these mechanisms with a limbless robot under a turning gait. The robot embedded with the sensory reflex mechanisms on the yaw modules is able to respond actively and correctly to external stimuli, which further endows the robot with the ability to behave adaptively to different environmental conditions. Hence, we conclude that the integrated sensory reflex mechanisms works well with our CPG model to respond to external environment stimuli.

Chapter 6

Development of Adaptive Locomotion Based on a Feedback Coupled CPG Model

Contents

6.1 Introduction

So far, we have introduced the fine-scale level of the hierarchical control architecture (see details in Figure 1.2), including the CPG model, the limbless gaits and the reflex mechanism. Now we turn our focus on the large-scale level and develop adaptive locomotion for limbless robots.

6.1.1 Roles of sensory feedback

Although in Chapter 3, it has been mentioned that CPGs can endogenously produce coordinated patterns of rhythmic activity without any sensory feedback, CPGs can also respond to sensory feedback to alter the generated pattern (Hooper, 2000). However, how the sensory input is routed to CPGs is not yet fully understood. Only two possibilities exist. One is that sensory input is transmitted to different neurons in the CPG circuit, while the other is that sensory input is fixed to some neurons in the CPG circuit. As a consequence of sensory input, no matter which one is true, sensory feedback varies appropriately the coordinated pattern. Indeed, sensory feedback plays three roles in altering CPGs and facilitate adaptation to the environment (MacKay-Lyons, 2002). First, it can reinforce CPG activities.

For instance, if a foot in walking is in the stance phase, sensory feedback would help to descend to enhance stability. Second, sensory feedback contributes to gait transition. For example, a cat on a treadmill could change the gait from walking to trotting if the speed of the treadmill increases. Third, sensory feedback has a timing function on CPGs that ensures the modified outputs to adapt to the environment. In order to maintain certain coordination relationships, sensory feedback received from external environment may not only change the local part of the pattern but also further affect the other part of the pattern. This also implies that even though sensory input occurs within a short time, a series of patterned outputs may change.

6.1.2 Adaptive locomotion using sensory feedback

In robotics, the third role of sensory feedback is the most attractive to engineers. Since sensory feedback is helpful to form a control loop feedback mechanism, it is of great benefit to achieving adaptive locomotion. When the environmental conditions change, the closed-loop robot is aware of the environmental change by means of onboard sensors. According to available sensory information, the controller modifies the generated pattern and correspondingly the robot adapts to the environmental change.

Regarding the development of adaptive locomotion using the CPG-based controller, several issues need to be considered. On the one hand, although sensory information is important that contains the environmental change, it is impossible to be directly used by CPGs due to the possibility of disrupting the output pattern. On the other hand, CPG models usually contain tunable parameters that control relevant characteristics of oscillation. However, it is difficult to appropriately adjust the control parameters to adapt to the environment without any sensory feedback. Considering the lack of association between the sensory information and the CPG model, it is necessary to design a reasonable pathway to integrate sensory information into CPGs in a systematic manner. In the literature, many approaches have been proposed to establish the connection. The following introduces the ways of integration with some famous CPG models.

The pathway of sensory feedback can be successfully coupled with the Matsuoka's model (see Section 3.1.2) and the Ekeberg's model (see Section 3.1.3). Ryu et al. (2010) proposed a frequency-adaptive oscillator that can learn and adapt to changes in the frequency of sensory feedback signals. They applied an accelerometer on the head of a snake-like robot and used the measured speed and a Hopf oscillator to generate periodic input signals, as described below:

$$\begin{cases} \dot{\omega} = k_1 e + k_2 \dot{e} \\ \dot{x_1} = (\mu - {x_1}^2 - {x_2}^2)x_1 - \omega x_2 \\ \dot{x_2} = (\mu - {x_1}^2 - {x_2}^2)x_2 - \omega x_1 \end{cases} \tag{6.1}$$

where ω is the frequency of the Hopf oscillator; k_1 and k_2 are gains; e and $\dot{e}$ are the error and the derivative between the desired and real speed; μ is the damping

coefficient; x_1 and x_2 are the internal states of the oscillator. The periodic signal x_1 is designed to further integrate into the Matsuoka's CPG model (see introduction in 3.1.2, as a feedback term in equation 3.6). Thus, the detected forward speed together with the desired speed can alter the periodic input signal x_1 and finally affect the CPG output. The control strategy is verified by employing the snake-like robot moving with constant velocity over terrain with varying frictions.

Another successful example is from Inoue et al. (2007), who developed a simulated snake-like robot that is adaptive to changing ground friction. The robot is equipped with force sensors on the bottom. Two kinds of CPG models, namely, the Ekeberg's model and Matsuoka's model, are separately adopted as controllers of serpentine locomotion. For each of the controllers, they used sensory interneurons to bridge the sensory information and the controller. The dynamic of the sensory interneuron is described as follows:

$$\dot{p} = \frac{1}{\tau_D}(S_{t-\Delta t} - p) \tag{6.2}$$

$$u_L = \begin{cases} p & if \quad p > 0 \\ 0 & otherwise \end{cases} \tag{6.3}$$

$$u_R = \begin{cases} 0 & if \quad p > 0 \\ -p & otherwise \end{cases} \tag{6.4}$$

where p is an internal state; τ_D is a time constant; S is the measured value from the force sensor; Δt is the purposive phase shift to sensory information; and u is the output of the sensory interneuron. Thus the sensory information can be transmitted into the CPG model via the sensory interneurons and affect the CPG output. In order to achieve adaptive serpentine locomotion on the flat terrain with different frictions, a genetic algorithm is used to obtain CPG parameters. The performance of the adaptive serpentine locomotion is mainly evaluated by the averaged speed and consumed powers. After the parameter optimization, both of the two CPG models realize adaptation to changing friction condition.

Sensory feedback can also be integrated into Hopf oscillators. Righetti and Ijspeert (2008) proposed a way of designing CPGs with coupled Hopf oscillators. They added sensory feedback from touch sensors on the extremities of the quadruped robot into the network of oscillators, so as to strongly couple the CPG-based controller with the mechanical system. In order to modulate the period of swing and stance phases in quadruped locomotion, they rewrote the Hopf equations with the feedback u added on the y variable:

$$\dot{x_i} = \alpha(\mu - {r_i}^2)x_i - \omega_i y_i \tag{6.5}$$

$$\dot{y_i} = \beta(\mu - {r_i}^2)y_i + \omega_i x_i + \sum k_{ij} y_j + u_i \tag{6.6}$$

The feedback term takes effect in two cases. For the first case, when the phase transition is approaching, the feedback accelerates the oscillation for fast transition. For the other case, when the phase transition is happening, it stops the oscillation

for stable supporting. According to the current state and the sensor values, the feedback is designed as:

$$u_i = \begin{cases} -sign(y_i)F & \text{fast transition} \\ -\omega_i x_i - \sum k_{ij} y_j & \text{stop transition} \\ 0 & \text{otherwise} \end{cases} \tag{6.7}$$

where F is a constant that controls the delay before the transition actually occurs. In this way, the explicit change of the phase of the oscillator would be helpful to generate faster and more stable quadruped locomotion. Through simulations of slope climbing, the result reveals that the feedback is able to stabilize the robot and help it to adapt to the terrain.

Even in the phase oscillator, sensory feedback structure can be also appropriately developed. Sato et al. (2011) designed a discrepancy function based on phase oscillators. They first extracted the discrepancy between the control and the mechanical systems and then fed it back into the oscillators in the control system. Thus the discrepancy function would modify the phase of the oscillator and finally reduce the discrepancy. The following equations describes how the phasic feedback is integrated:

$$\begin{cases} \dot{\theta}_i = \omega + \varepsilon sin(\theta_{i-1} - \theta_i - \Psi) + f_i \\ \overline{\phi}_i = \phi_0 \cdot sin(\theta_i) \\ f_i = -\sigma \frac{\partial I_i^m}{\partial \theta_i} \\ I_i = \left| \phi_i - \overline{\phi}_i \right| \end{cases} \tag{6.8}$$

where f is the phasic feedback; I is the discrepancy function which is defined as the difference between the desired angle $\overline{\phi}$ and the real angle ϕ. In addition, they developed the tonic feedback η to control the joint stiffness:

$$\begin{cases} \dot{\eta}_i = \alpha(\beta I_i^m - \eta_i) \\ \overline{\phi_i^{(u)}} = \overline{\phi}_i + \eta_i \\ \overline{\phi_i^{(l)}} = \overline{\phi}_i - \eta_i \end{cases} \tag{6.9}$$

where $\overline{\phi_i^{(u)}}$ and $\overline{\phi_i^{(l)}}$ represent the upper and lower motors in one joint. They applied the phasic and tonic control on a snake-like robot to move on the upslope. Since the joint of the robot is special designed with silicone rubber, it acted not only as an actuator but also as a passive rotating element. The results show that a well balanced coupling between the phasic and tonic feedback could lead to successfully negotiation with environmental changes.

Besides aforementioned methods, several other contributions have also proposed ways to integrate sensory feedback in CPGs, such as phase reset (Aoi and Tsuchiya, 2005, 2006), phase entrainment (Kimura et al., 2007; Seo et al., 2010), amplitude optimization (Wu and Ma, 2010) and others (Manoonpong et al., 2008; Liu et al., 2011). However, in most cases, these methods are specific to a particular robot, as

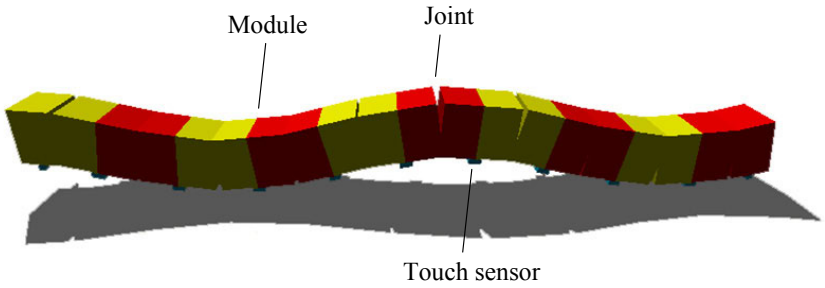


Figure 6.1: The simulated pitch-pitch conneted limbless robot.

well as a particular application. Furthermore, the development of adaptive locomotion for limbless robot on irregular terrains is rarely seen. In this chapter, we propose a genetic CPG-based control architecture and use the propose CPG model to realize adaptive locomotion of limbless robots under environmental changes.

6.2 Adaptive control system

Caterpillars are among the most successful climbers and can maneuver in complex three-dimensional environments (Mezoff et al., 2004). Therefore, it is worthwhile to use limbless robots to emulate their adaptive behaviors. Since we have studied the principles behind the locomotion of caterpillars (see details in Section 4.2.3), here we only focus on trying to imitate their adaptive and efficient locomotion behaviors.

6.2.1 Robot construction

For the simulation, a simplified modular caterpillar-like robotic configuration is designed for the locomotion research in ODE. As shown in Figure 6.1, each mechanical module of the robot is designed as a simple rigid box. A joint rotating along the horizontal axis, located between every two modules, allows adjacent modules to rotate in a vertical plane with a range of $\pm 90°$. The joint is designed to be driven by a servo motor, in which a maximum torque of 0.314 $N \cdot m$ and a maximum speed of 1.45 rad/s can be obtained. Moreover, touch sensors are installed on the abdominal parts of the robot, with a function similar to that of the prolegs of living caterpillars. As a result, the control system of the robot is able to collect terrain contact information regularly. The related parameters of the robot module are listed in Table 6.1.

To imitate the motion of real caterpillars in this experiment, we use a chained inhibitory CPG circuit to generate a linear gait and follow the investigation of the caterpillar-like locomotive parameters in Zhang et al. (2009). Thus, all oscillators in the chained inhibitory circuit are predefined, with an amplitude of 20° and a phase difference of 120°.

Table 6.1: Specification of simulated robot module

Component	Parameter	Value
Module	Length (mm)	72
	Width (mm)	52
	Height (mm)	52
	Weight (g)	150
Touch sensor	Length (mm)	10
	Width (mm)	52
	Height (mm)	10
	Weight (g)	10
Joint	Rotate speed (rad/s)	≤ 1.45
	Torque ($N \cdot m$)	≤ 0.314

6.2.2 Control architecture

Figure 6.2 illustrates the control architecture of the robot. It consists of three main components: The locomotion control part, the environment part, and the reaction control part.

The locomotion control component plays a role in gait generation. As introduced in Chapter 3, the proposed CPG model inspired by the lamprey's spinal circuit is as well-developed as the controller of the robot. On the one hand, the CPG model is able to generate traveling waves among oscillators. The output of the model is therefore designed as desired joint angles to each module of the robot. Due to the phase lag between the modules, the robot can perform peristaltic crawling behavior like real caterpillars. On the other hand, since the CPG model is developed at the neural level, the pathway of sensory feedback can be created via additional sensory neurons. Thus sensory input generated by "Parameter modulator" can be afferent to the CPG model and shape the output of the model.

In the environment component, touch sensors on the bottom of the robot are responsible for collecting the raw sensory data. In order to effectively observe and analyze the interaction between the robot and the environment, the sensory information is regularly sampled and stored as time series data. After interacting with the environment, the sensory information is transmitted to the reaction control component.

The reaction control component is an essential part of the control system and enables the robot to move adaptively when confronted with irregular terrain. It can be further divided into three sub-functional parts, namely a sensor processor, a reaction maker, and a parameter modulator:

- **The sensor processor** filters the raw sensory information and converts it

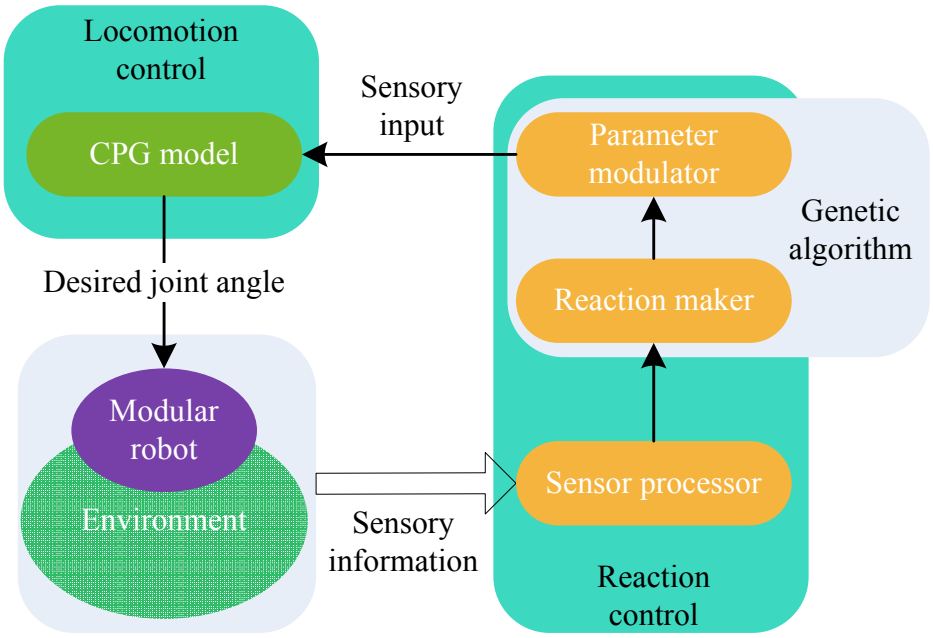


Figure 6.2: Adaptive control architecture.

to a sequence, or string, of binary bits. The binary string indicates the contact states of the robot. Each binary bit value implies accumulated contact information over a period of time instead of an instantaneous contact event at that particular touch sensor.

- **The reaction maker** determines module reflex behavior. For each module of the caterpillar-like robot, the reaction maker maps the contact states into a reaction rule that determines whether to generate a sensory input to the module in a given time step.
- **The parameter modulator** is responsible for generating sensory input for the CPG model according to the reaction rules, so as to adaptively shape the locomotion gait to be compliant with the environment.

Although the control architecture described above may seem sufficient for our closed-loop control system, there are still several key issues that must be solved. First, the mapping mechanism between the contact states and reaction rules need to be considered. Second, one must determine what amount of sensory input should be generated in such a manner as to successfully achieve adaptive movement. Together, these two aspects require that a large number of parameters be simultaneously and appropriately chosen. Since this task is not easily solved with analytical methods, a genetic algorithm (GA) is selected as the tool to find an optimal solution that yields satisfactory adaptive behavior. Once the set of related parameters has evolved, the reaction maker and the parameter modulator can operate properly. As a result, the robot is endowed with the capability of traversing over complex terrain.

6.3 Sensor processor

The sensor processor handles the raw touch sensor data and identifies the contact state for each module at regular intervals of time. It has two fundamental functions. The sensor processor is able to filter the raw sensory data into instantaneous contact states. Since the instantaneous contact states change very frequently, they are not suitable to be directly used for reaction. In contrast, an accumulated contact state that contains recently passed instantaneous contact states is much better. This is due to that an accumulated contact takes resent time series data into consideration and averages the frequent changes of instantaneous contact state. Therefore, the second function of the sensor processor is to convert instantaneous contact states into accumulated contact states.

Here, we use "module state" to indicate accumulated contact information rather than instantaneous contact state. Two types of module states, namely "periodic touch" and "hanging in the air", are designed, both of which imply an accumulated contact state over a period of time. If there is no contact detected from the touch sensor on the module in a fixed period of time, the module state is classified to "hanging in the air". Otherwise, it is considered to be in "periodic touch".

Although the module states are not as sensitive as instantaneous contact states, they can vary with the change of environmental conditions. Taking slope climbing for example, when a caterpillar-like robot moves from a flat terrain to a slope, two intermediate modules (module 3 and module 4) lose contact with the ground due to internal torques, as shown in Figure 6.3. In this case, the states of the two modules with loss of contact will change from "periodic touch" to "hanging in the air".

In order to identify the change of module states automatically, a filter is used to classify the raw data as binary values, representing either a touched or an untouched state, respectively. This filter can be described as follows:

$$t_{(i,n)} = G(R_{(i,n)}) \tag{6.10}$$

where t denotes the instantaneous touch state of the ith module at time step n, R is the raw data of the ith module's touch sensor, and G is a threshold function.

As mentioned above, because the instantaneous touch states change too frequently, using this information for CPG modulation may lead to unstable jerk behavior. Instead, accumulating the instantaneous values of the touch states over more than one period of locomotion yields an acceptable solution for module state identification:

$$s_{(i,n)} = \begin{cases} 1 & \text{if } \sum_{n'=n-kN}^{n} t_{(i,n')} > 0 \quad (k \geq 1) \\ 0 & \text{otherwise} \end{cases} \tag{6.11}$$

where $s_{(i,n)}$ is the state of the ith module at time step n; N denotes the number of time steps in one period of locomotion; and k is the number of historic touch states to consider. A module state value of 1 means that the module is touching the surface, whereas a 0 means that the module is in the air.

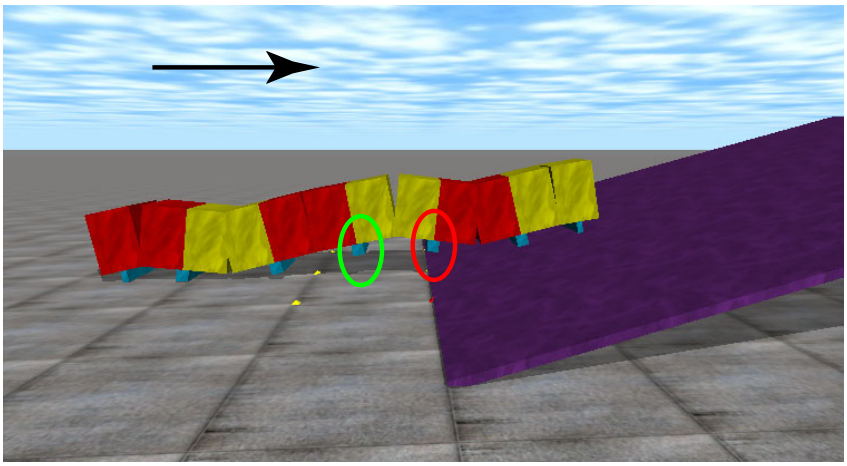


Figure 6.3: During the caterpillar-like robot moving towards the slope, internal torques make the third and fourth modules lose contact with the substrate. Correspondingly, their module states change from "periodic touch" to "hanging in the air".

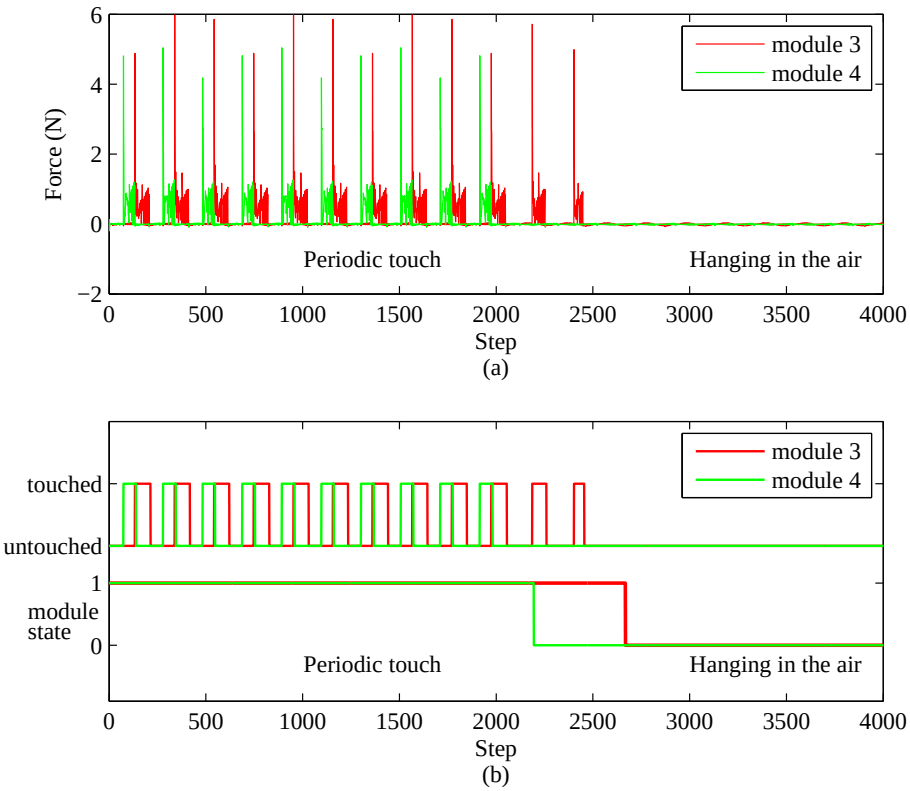


Figure 6.4: Sensor data processing. (a) Raw data of touch sensors are gathered during the robot is climbing a slope. (b) Module states are produced by filtering the raw data and accumulating the touch state over more than one period of time.

Figure 6.4 illustrates the corresponding data of the example. Figure 6.4(a) is the raw touch sensor data of the two consecutive modules 3 and 4 during the robot is climbing the slope, with spikes in the force measured by the touch sensors, indicating contact. From this example, it shows modules 3 and 4 have periodic touch for the first 2450 steps and 2000 steps, respectively, before they begin hanging in the air. The top of Figure 6.4(b) is the instantaneous module states that result from applying the threshold function to Figure 6.4(a), whereas the bottom of Figure 6.4(b) is the module states based on accumulated contact information. From the result, it shows that using the history of touch states to determine the module states causes a lag of approximately $(k-1)N$ samples between the raw touch sensor data and the processed binary module states. In the example in Figure 6.4, a history of $k = 2$ is used. Since there are about 200 steps between each detected touch, the delay in detecting a module state change reaches about 200 steps (see bottom of Figure 6.4(b)). Inspite of the time lag, it does not affect the sensor processor in identifying the change of module states.

6.4 Reaction maker

The reaction maker is used to analyze reasonable reactive behavior for each module of the robot. It contains two basic functions. The first role of the reaction maker is to help each module to evaluate its status. It takes neighboring module states into account. By means of weighted average, the reaction maker can generate neighboring state values on each side of the module. The other role of the reaction maker is to associate generated neighboring state values with reaction rules. Based on the reaction rule, the reaction maker can finally determine the generation of sensory input for each module.

In the reaction maker, l-nearest neighboring module states on both sides of the module are collected for the status evaluation. l is defined as an integer in range of $[1, m]$, where m is the total number of modules the robot has. We denote the status function as a set of module states:

$$S_l(k,n) = < s_{(k-l+1,n)}, s_{(k-l+2,n)}, ..., s_{(k,n)} > \quad (6.12)$$

$$S_r(k,n) = < s_{(k+1,n)}, s_{(k+2,n)}, ..., s_{(k+l,n)} > \quad (6.13)$$

where $S_l(k,n)$ and $S_r(k,n)$ represent the left and right side of status of the kth module at time step n, respectively.

Note that in 6.12 and 6.13, due to the introduction of l, the value of the first subscript k in $s_{(k,n)}$ may exceed the defined boundary, i.e. k may be smaller than 1 or larger than the module number m. Hence, the corresponding modules are not existing and their module states $s_{(k,n)}$ are meaningless. In such cases, we assume there are unlimited virtual modules connecting at each ends of the robot, and their module states are always "hanging in the air", namely with a value of 0.

Since the sets of module states S_l and S_r are binary strings, they are not intuitive enough before they have a numerical representation for status evaluation. One way

to convert the binary string into a numerical value is to add their binary bits together by means of weighted average. In addition, because the nearer module states have more effect on representing the status of the evaluated module, the weight is supposed to be reduced with the growth of the distance between the current detected module and the evaluated module. Thus, the neighboring state values are defined as:

$$V_l(k,n) = \frac{\sum_{i=0}^{l-1} \theta^i s_{(k-i,n)}}{l} \tag{6.14}$$

$$V_r(k,n) = \frac{\sum_{i=0}^{l-1} \theta^i s_{(k+1+i,n)}}{l} \tag{6.15}$$

where $V_l(k,n)$ and $V_r(k,n)$ are the neighboring state value on the left and right side of the kth module at time step n, respectively; θ^i $(0 < \theta < 1)$ is the weight that reflects the degree of influence of the current detected module state on generating the neighboring state value for the evaluated kth module. It is noted that with the growth of i, θ^i decreases. This means the farther the detected module is away from the evaluated module, the less importance it is in generating the neighboring state value.

As for the status evaluation for each module, a status threshold function is defined as:

$$\Theta = \frac{\sum_{i=0}^{l-1} \theta^i}{l} \cdot \rho \tag{6.16}$$

where Θ is the status threshold and ρ $(0 < \rho < 1)$ is a proportional constant that determines the threshold percentages of module states in "periodic touch". If the neighboring state value exceeds the status threshold, the status of the corresponding side of the evaluated module is considered to be in "periodic touch". Otherwise, it is treated as "hanging in the air".

By comparing the neighboring state values V_l and V_r with the status threshold Θ, the reaction maker identifies three types of conditions and develops three corresponding reaction rules:

- **Case 1:** $V_l < \Theta$ and $V_r \geq \Theta$; or $V_l \geq \Theta$ and $V_r < \Theta$
  The first case holds only if the neighboring status value on one side of the evaluated module exceeds the status threshold while the other side does not. This reveals the evaluated module has one side of status in "hanging in the air" and the other side in "periodic touch". In such a case, the reaction maker is designed to only react to the side which is "hanging in the air". A certain amount of stimulus (λ_1 if the head side is "hanging in the air", while λ_2 if the tail side is.) is transmitted to the corresponding oscillator of the evaluated module, so that the side that is "hanging in the air" is able to make contact on the substrate.

- **Case 2:** $V_l < \Theta$ and $V_r < \Theta$
  The second case holds when the neighboring status values on both sides of the evaluated module fail to reach the status threshold. In such a case, the evaluated module is considered in a "camelback" pose, with both of its ends "hanging in the air". To eliminate such situation, the reaction maker draws a certain amount of stimulus, represented as λ_3, into the corresponding oscillator of the evaluated module. The afferent stimuli would shape the output of the oscillator and finally help the evaluated module to adapt to the terrain.

- **Case 3:** $V_l \geq \Theta$ and $V_r \geq \Theta$
  The final case concerns the recovery of the evaluated module during locomotion. As the neighboring status values on both sides of the evaluated module exceed the status threshold, both sides of the evaluated module are in "periodic touch", namely coming into contact with the terrain. To prevent the evaluated module from excessively adapting to the terrain, any stimuli that have been afferent to the oscillator should be removed once the corresponding evaluated module has adapted to the terrain. From a robustness point of view, this would enable these evaluated modules to recover to normal state and finally resume the whole robot from the modified gait to the original one.

6.5 Parameter modulator

To deal with sensory information, sensory neurons (SNs) are integrated into the model, as shown in Figure 6.5. For simplicity, the ellipses labelled "interneurons" represent the CIN, LIN and EIN on each side of the oscillator. Unlike the sensory neurons used in Section 5.2.2, the SNs on each side of the oscillator set both excitatory and inhibitory synapses to motoneurons. Due to the mirrored functionality of these sensory neurons, the two SNs are considered as a single one. When an external stimulus λ is transmitted to a SN, it will in turn have an inhibitory effect on the motoneuron on the same side and an excitatory effect on the motoneuron on the opposite side. As a result, the external stimulus will add a contribution to the output of the oscillator. We can thus rewrite 3.21 as follows:

$$\tau \dot{x}_{\{MN\}i} = -x_{\{MN\}i} + \sum \omega_s s_{\{MN\}i} + \beta A + x_{\{SN\}i} - x_{\{\overline{SN}\}i} \tag{6.17}$$

where $x_{\{}SN\}$ represents the output of the SN and the last two terms with a positive and a negative sign separately indicate an excitatory and an inhibitory effect from the SNs to the motoneuron, respectively.

The dynamic change of the sensory neuron is a piecewise function. It depends on what case its corresponding module belongs to. For example, as described in the previous section, when a module satisfies case 1 or 2, a certain amount of stimulus λ_p is accumulated in its sensory neuron, as shown in 6.18.

$$\delta_m \dot{x}_{\{SN\}i} = -x_{\{SN\}i} + A \cdot \sum_{n_0}^{n} \lambda_p \quad p \in \{1, 2, 3\} \tag{6.18}$$

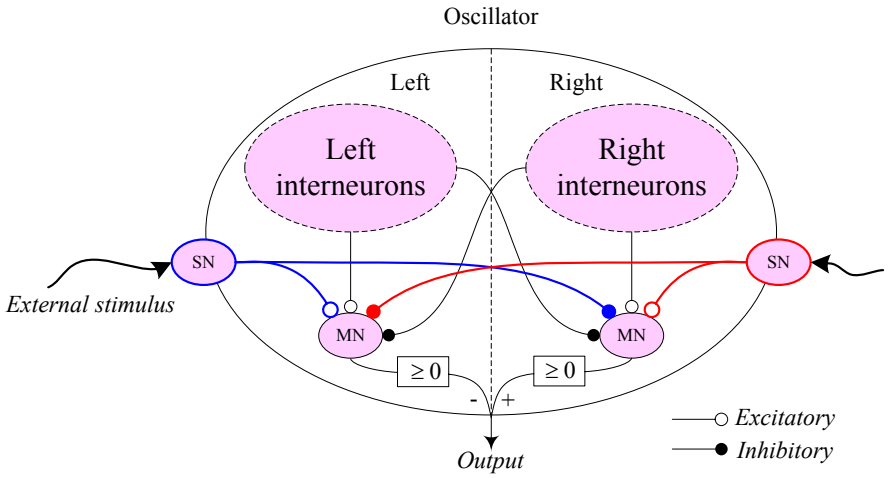


Figure 6.5: Sensory neuron integration. The oscillator with its interneurons simplified drawing, is the same to the one in Figure 3.5. Two additional sensory neurons with opposite effects on motoneurons are employed on each side of the oscillator for sensory feedback.

where δ_m is a time constant that controls how quickly the output signal x_{SN} will change; n is the current time step; n_0 is the time step when the state of the corresponding joint changes from case 3 to case 1 or 2; The sum of λ_p indicates an accumulation of stimuli from the time step n_0 to n; and A denotes the desired amplitude of the oscillator.

If a module belongs to case 3, its SN with a time constant δ_r will gradually remove the effect on motoneurons, as seen in 6.19:

$$\delta_r \dot{x}_{\{SN\}i} = -x_{\{SN\}i} \tag{6.19}$$

Figure 6.6 explains how the sensory input affects the CPG output via the SNs in the CPG model. Figure 6.6(a) shows normal CPG output generated using the chained inhibitory circuit, in which parameters are $\tau = 0.5$, $A = 20$ and $\alpha = 1$. In Figure 6.6(b), some amount of accumulated stimuli with parameters $\lambda = 1$ and $\delta_m = 0.5$ is transmitted into the SN in the third oscillator at the 2000th step, where its corresponding module satisfies case 1 or 2. After two seconds at the 2200th step, the corresponding module is found belonging to case 3 and thus all the afferent stimuli are removed from the oscillator. Correspondingly, the generated sensory input gradually recovers to a value of zero with parameter $\delta_r = 2$. Figure 6.6(c) shows the result of sensory integration for the third oscillator. Sensory integration is achieved by means of the superposition of the normal CPG output and the generated sensory input. From the figure, we see that the CPG output can recover to normal oscillation after the disappearance of afferent stimuli. Note that the goal of this example is not to emphasize the resultant coordination of all the oscillators after the integration of sensory neurons. Instead, it is only to emphasize the fact that

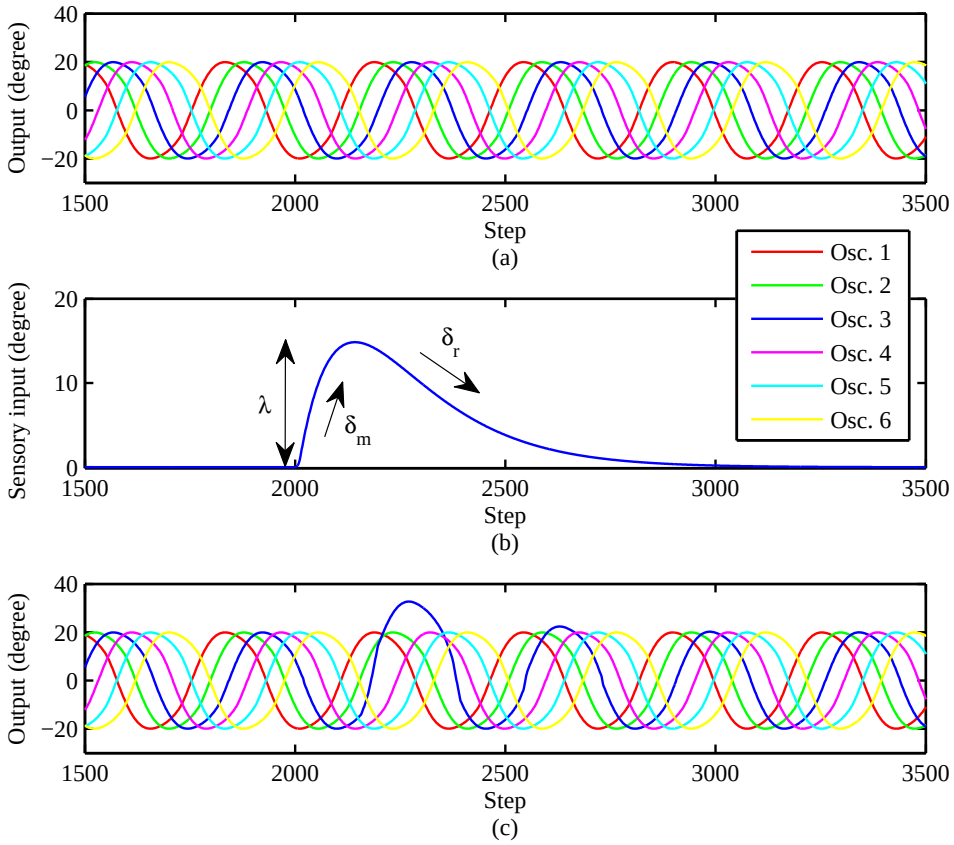


Figure 6.6: Effect of the sensory integration on the CPG output. (a) Normal CPG output. (b) The sensory input generated in oscillator 3 varies with the amount of afferent stimuli. (c) The variation of the CPG output after sensory integration over time.

the CPG model is able to respond to external stimuli with the help of integrated sensory neurons.

6.6 Motion optimization

As described above, the sensory neuron integrated in the CPG model plays an important role in adaptive locomotion. However, it is difficult to determine the amount of stimulus λ for each case, as well as the time constant (δ_m and δ_r) for the output of the sensory neuron.

In this work, a genetic algorithm (GA) is employed as a solution to evolve the related parameters λ, δ_m and δ_r in the CPG. A real number GA derived from the standard binary GA is used, with genes containing real values within a specified range instead of employing binary values. Candidate solutions are encoded as genomes of fixed-length genes. Each gene corresponds directly to one parameter. For our control model, there are 5 parameters that need to be evolved in each

Table 6.2: Optimized parameters of CPG model

Parameters	Case	Description	Range
λ_1	1	Stimulus for head side bending	$[-1, 1]$
λ_2	1	Stimulus for tail side bending	$[-1, 1]$
λ_3	2	Stimulus for "camelback" pose	$[-1, 1]$
δ_m	1 or 2	Time constant for modulation	$(0, 20]$
δ_r	3	Time constant for recovery	$(0, 20]$

oscillator, as shown in 6.2.

A fitness function is defined over the genetic representation and measures the quality of the represented solution, as described below:

$$fitness = \eta \cdot \frac{v}{v_0} + (1 - \eta) \cdot \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{s_{(i,j)}}{mn} \tag{6.20}$$

where η $(0 < \eta < 1)$ is a proportional variable. v is the average speed for the robot to climb over uneven terrain, whereas v_0 is the speed the robot climb on flat terrains. $s_{(i,j)}$ is the module state. m and n are module number and time step respectively.

The aim of the GA is to find the best genome that enables the fitness function to reach a maximum value. The fitness function in 6.20 is a weighted sum of two different components of locomotion. The first component rewards the velocity of the adaptive locomotion for the robot climbing over uneven terrains relative to the velocity over flat terrains. The second component rewards locomotive stability measured as the percentage of the average number of modules that are in contact with the terrain. The weighting factor η in 6.20 determines which component has a higher priority during the evolutionary process and ensures that the fitness function is constrained to the range (0, 1). In this study, η is set to 0.85, which means that the locomotive speed in uneven terrain is weighted more than locomotive stability.

6.7 Simulation and experiment

Simulation and on-site experiment are both carried out in this section to validate the effectiveness of the control system in realizing adaptive locomotion with sensory feedback.

6.7.1 Simulation

In the ODE simulation, we construct a pitch-pitch connected robot with 6 box modules (see details in Section 6.2.1) as the test bed of this experiment. A linear gait with an amplitude of 20° and a phase difference of 120° is applied on the

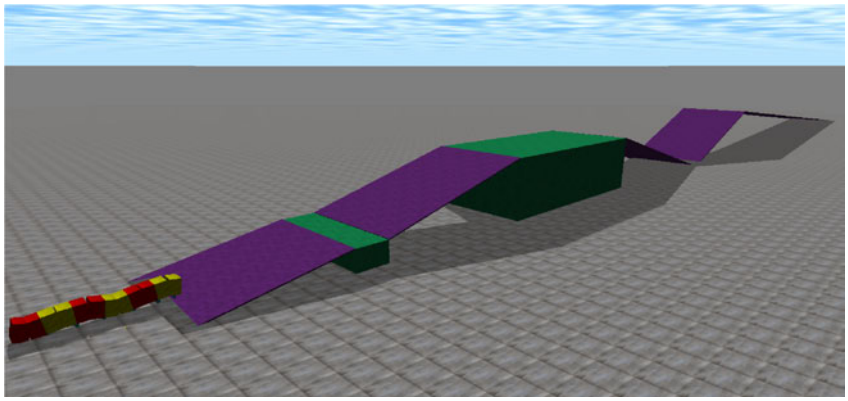


Figure 6.7: The simulated scenario.

Table 6.3: GA operations and parameters

Genome type	Real genome
Selection	Roulette selection and elite selection
Crossover	Uniform crossover
Mutation	Random number
Population size	100
Crossover rate	0.9
Mutation rate	0.01
Termination criterion	Converge to 0.99 after 50 generations

caterpillar-like robot as the basic locomotive pattern. A simulation scenario is also constructed in ODE, consisting of ramps with different inclinations and different angles between them, as shown in Figure 6.7.

The goal of this experiment is to make the caterpillar-like robot learn to climb over the complex terrain adaptively and autonomously. As described in the adaptive control system (see Figure 6.2), all the components in it are deterministic except for the "reaction maker" and the "parameter modulator", in both of which the amount and the speed of sensory input that is fed back to the CPG model are required to be optimized. Here, the GA implemented in GAlib library (Wall, 1996) is utilized to evolve related CPG parameters listed in Table 6.2. The optimization procedure works as follows: For each generation, the GA produces a population of genomes. For each genome, it is tested in the simulated scenario with a fixed number of time steps. An evaluated value according to the fitness function in 6.20 is returned back to the GA after each simulation. Once the test of the whole population finishes, the GA reproducts a new population of genomes based on these returned values and some GA operations. A new round of simulation then starts until the termination criterion is met. Details of the GA operations and parameters applied

Table 6.4: Parameters in the simulation

Parameters	Value	Description
θ	0.9	Weight of module state
ρ	0.6	Percentage of status threshold
v_0	$0.1(m/s)$	Speed on flat terrains
m	6	Number of modules
n	100000	Total time steps

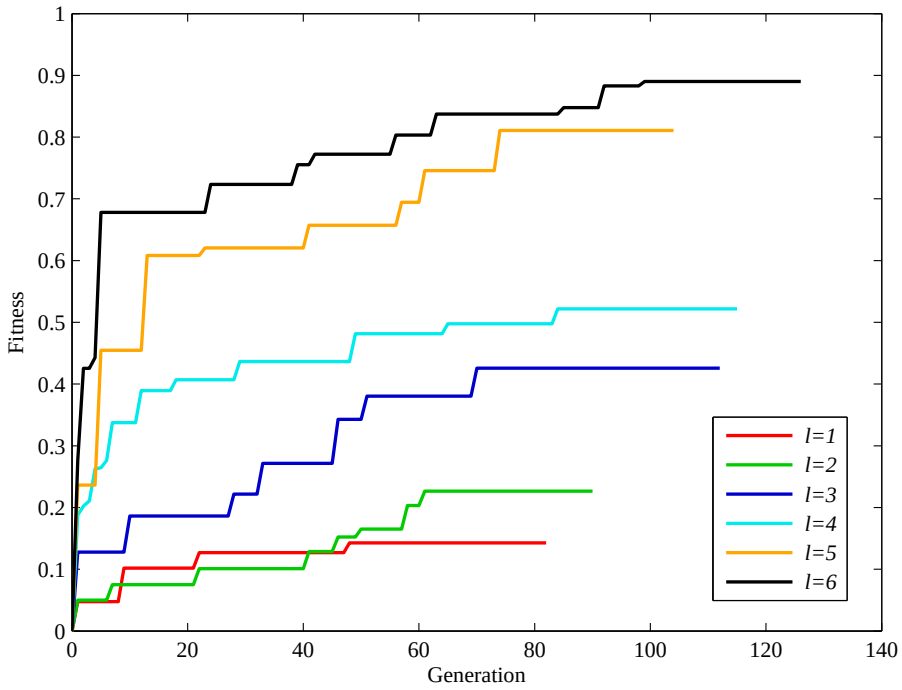


Figure 6.8: The fitness convergence over generations. The best fitness value increases with the growth of the number of neighboring module states.

Table 6.5: Performance of each GA procedure

l	Average speed(m/s)	Average touch(%)	Description
1	0.008	50.83	Get stuck at the fist slope
2	0.016	60.16	Get stuck at the second slope
3	0.037	71.34	Get stuck at the last but one slope
4	0.048	75.82	Get stuck at the last slope
5	0.081	84.99	Success to climb over slopes
6	0.089	91.01	Success to climb over slopes

in this experiment are listed in Table 6.3. As a result of the GA, the best genome is chosen as the evolved result of the CPG parameters. According to these evolved parameters, the robot can be controlled by the following routine at each time step n:

1. Get raw sensor data at time step n;
2. Process the data and update V_l and V_r;
3. For each module i, check which case it belongs to and determine reaction time constant δ_i as well as the amount of afferent stimulus λ_i;
4. Calculate the output of the sensory neuron in each oscillator;
5. Calculate the output of all the oscillators;
6. Apply the result to all the modules.

In order to promote the performance of adaptive locomotion, we investigate how it varies with the change of number of neighboring module states. As mentioned in Section 6.4, the reaction maker takes l-nearest neighboring module states into consideration and uses them to calculate the neighboring state values V_l and V_r. From equations 6.14 and 6.15, it is noted that changing the number of neighboring module states l will change the neighboring state values V_l and V_r, finally resulting in the change of selecting reaction rules. Hence, changing the number of neighboring module states l indeed has impact on the performance of adaptive locomotion.

Since the robot consists of six modules, the GA procedure is repeated 6 times with only the change of l ($l = 1, 2..., 6$). The other related parameters are fixed, as listed in Table 6.4. Figure 6.8 shows the fitness convergence over generations with different number of neighboring module states. Note that for each GA procedure, the number of generation is different. This is because we use a termination criterion that requires the similarity of two adjacent generations converging to 99 percent. The figure reveals that the more number of neighboring module states the control system uses, the higher performance of adaptive locomotion it gets. The result is reasonable because the more number of neighboring module states taking part in the calculation of the neighboring status can better objectively reflect the evaluated module's status, which thus is benefitial for further selecting the proper reaction rules. The performance of each GA procedure with different l is listed in Table 6.5.

Figure 6.9 illustrates the procedure of the adaptive locomotion with $l = 6$ for the robot climbing over the slopes. Note that the position of each sub-figure (b)-(g) corresponds to a position labeled in Figure 6.9(a).

Through the simulation, the feasibility of the control mechanism is validated.

6.7.2 On-site experiment

A real pitch-pitch connected modular robot with 6 aluminium modules (see more details in Table 5.1) is constructed for the on-site experiment. Although each end of

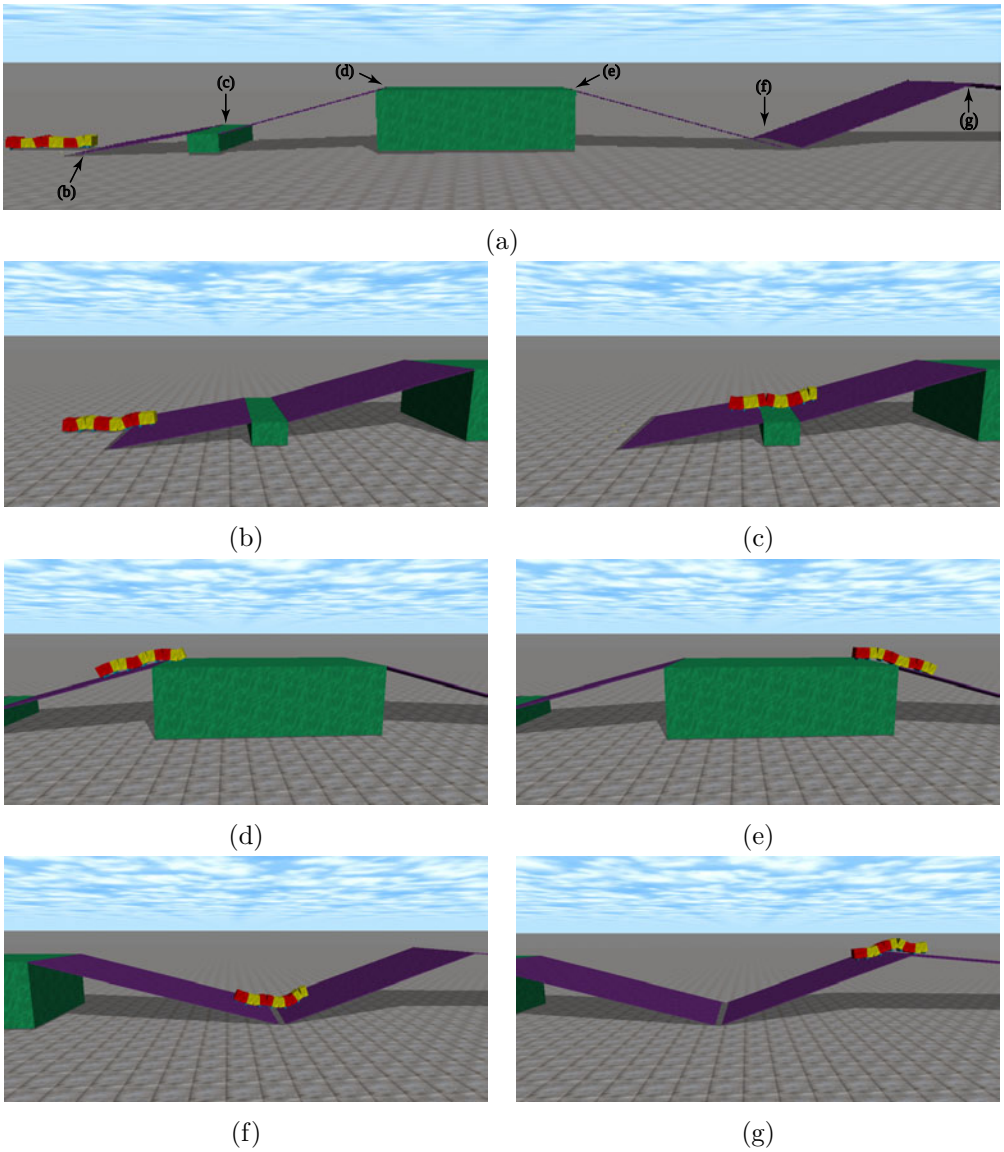
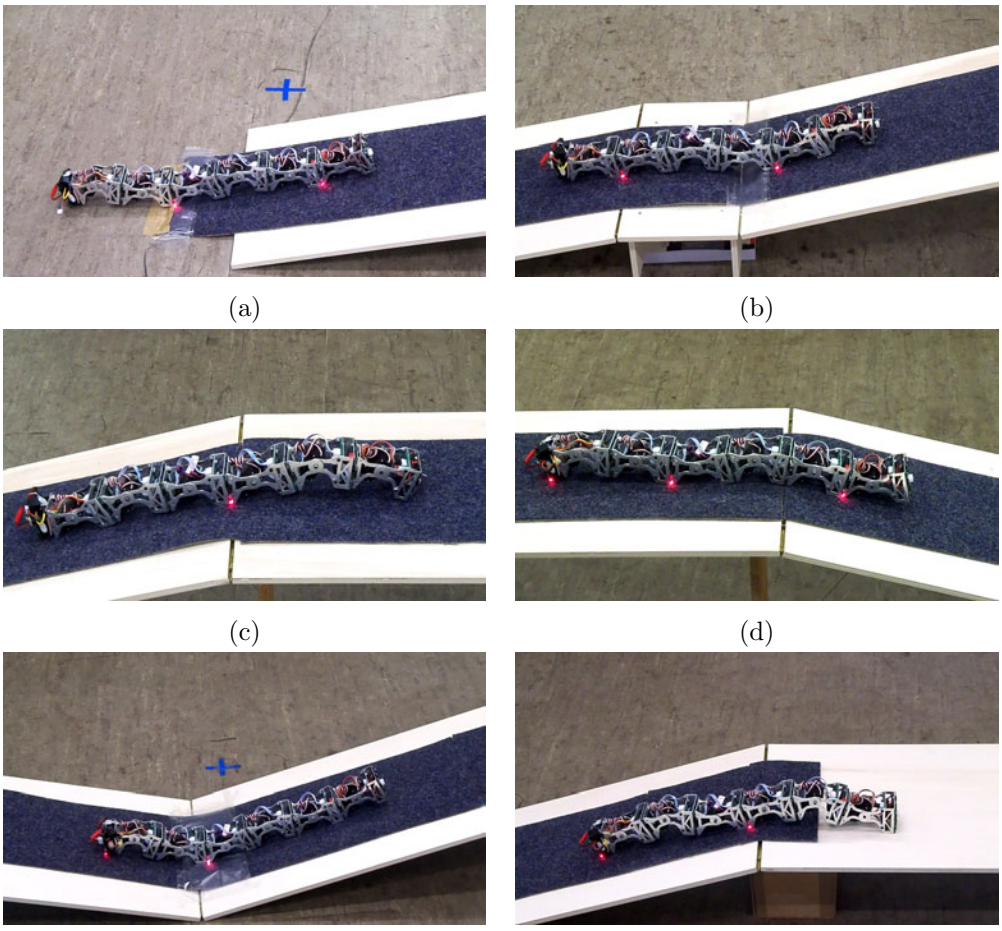


Figure 6.9: Simulation for adaptive locomotion. (a) The overall of the constructed environment. Slopes with different angles (from 5° to 20°) and boxes are spliced together. (b)–(g) Adaptation of caterpillar-like locomotion. The robot managed to climb over these slopes with the help of sensory feedback.

Table 6.6: Evolved parameters

Module	parameters				
	λ_1	λ_2	λ_3	δ_m	δ_r
No.1	0.40	-	-	6	1.5
No.2	0.55	0.10	-0.25	17.5	1.5
No.3	0.10	0.25	-0.30	11	3.5
No.4	0.25	0.45	-0.35	8	9
No.5	0.05	0.65	-0.15	6.5	17
No.6	-	0.40	-	17	3.5



(a) (b) (c) (d) (e) (f)

Figure 6.10: On-site experiment. (a)–(f) Scene of the robot climbing over slopes in the environment.

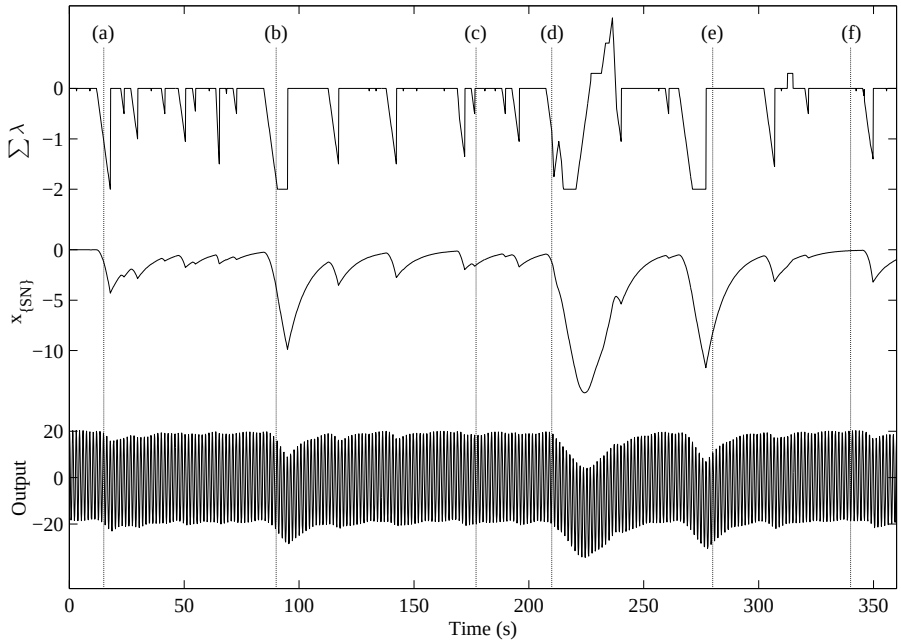


Figure 6.11: Tracked data of module 3 during the on-site experiment, including the afferent stimuli over time (top); the sensory input generated by the corresponding sensory neuron (middle); and output (bottom).

the module on the robot is surrounded by six touch switches, only the touch switches on the bottom are used to collect the terrain information in this experiment.

A testing scenario is also constructed. It is 6 meters long and 0.4 meters width, which is totally the same as the one constructed in the simulation. The on-site experiment is examined with the same set of evolved parameters when $l = 6$, as shown in Table 6.6.

Figure 6.10 shows a series of snapshots taken from a video recorded during the experiment. The robot successfully uses adaptive behaviors when climbing over the slopes. It is found that the body shape of the robot can be bent flexibly to adapt to the terrain with the help of integrated sensory information of the environment. In contrast, the caterpillar-like robot without sensory feedback gets stuck at the second slope (the position as in Figure 6.10(b)).

Figure 6.11 shows an example from the on-site experiment where a module in the center of the robot (module 3) is tracked and analyzed. The three curves in the figure from the top to the bottom represent the afferent stimuli, sensory input and joint output, respectively. The vertical lines indicate the moments when the robot is climbing over slopes shown correspondingly in Figure 6.10(a)-(f). Note that the maximum sum of afferent stimuli is limited to ± 2, for the sake of avoiding excessive output to the module. The output of the module is shifted when the robot is climbing over slopes. More specifically, the output of module 3 at the slopes of (b), (d) and (e) offsets more than the output at other slopes, due to more amount of stimuli being afferent to the corresponding sensory neuron. More data of the

on-site experiment can be found in Appendix A.

Based on the results, we conclude that the proposed control mechanism is effective in realizing adaptive locomotion pattern for caterpillar-like robot.

6.8 Summary

In this chapter, a framework of closed-loop control for adaptive locomotion of a caterpillar-like robot is presented. By investigating the roles of sensory feedback in biology, we find the sensory feedback is able to alter CPG output and facilitate adaptation to environment. We also investigate several methods that couple sensory feedback with CPG models for robotic applications.

Inspired by the literature, we develop an adaptive control system, in which the pathway of sensory feedback is successfully incorporated into our proposed CPG model. The control system applies the CPG model for gait generation and uses a reaction control component for adaptive behavious generation. The reaction control component is composed of a sensor processor, a reaction maker and a parameter modulator. This control scheme consists of the following stages: First, the sensor processor processes and transforms touch sensor information into module states. Then, the reaction maker calculates neighboring state values based on the module states and determines reactive strategies for each module, where the reactive strategies are further optimized by GA. Finally, according to the reactive strategies, the parameter modulator generates the sensory input and feeds it back to the CPG model. Thus the CPG output is modified and its corresponding gait is shaped to adapt to the environment. Incorporating the closed-loop controller in a caterpillar-like robot, both simulation and on-site experiment confirm that the robot flexibly adapts to, and manages to crawl across complex terrain.

Chapter 7

Development of Adaptive Locomotion using Policy Gradient Reinforcement Learning

Contents

7.1 Introduction

Although a closed-loop control system for adaptive limbless locomotion has been successfully developed in the previous chapter, the control system itself is not perfect. This is because it contains human intervention, such as the design of the reaction maker, which restricts the searching of optimal reactive strategies in the whole action space. An alternative approach is to use reinforcement learning to generate reactive strategies without any human intervention. In this chapter, we focus on removing the human intervention and improving the control system with a policy gradient reinforcement learning method.

7.1.1 Policy gradient reinforcement learning

Reinforcement learning (RL) is a machine learning paradigm that is well suited for use in robotics. In RL, an agent learns behavior through trial-and-error interactions with the environment. On each step of interaction, the agent perceives the environment by measuring states and then chooses an action according to its policy. The action changes the state of the environment and a scalar reward as a result of the state transition is obtained. The value of the reward reflects how good the selected action is in this round of interaction. The agent repeats the step of interaction over

time through systematic trial and error, receiving a numerical reward after every action choosing. The rewards are used to teach the agent to choose actions that tend to increase the long-term sum of rewards. In the end, the agent can learn an optimal policy that maps states to actions, which in turn maximize the accumulated future rewards (Sutton and Barto, 1998).

There are two main approaches for solving reinforcement learning problems: the value function approach and the direct policy search algorithm(Meuleau et al., 2001). The value function approach first searches for the optimal value function (Q-function, V-function), and then use the optimal value function to deduce the optimal policy. The most famous algorithms belonging to this category are Q-learning and SARSA(λ). Although the value function approach is useful in many applications, it only works in completely observable environments. Furthermore, it requires a considerable amount of computation and is lack of generalization for continuous state and action spaces. In contrast, the policy search method such as REINFORCE algorithm aims to find an optimal policy directly, without the help of a value function. It represents a policy by a parametric approximator, and seeks an optimal parameter vector based on the gradient descent of the policy. Such a policy gradient method typically finds only local optima of the expected reward and converges slowly if the information based on which it acts is noisy, but it accommodates a partially observable Markov decision processes (POMDPs) very well.

In the policy gradient method, a policy can be represented using an independent function approximator with its own differentiable parameters (Sutton et al., 2000; Busoniu et al., 2011). The parameterized function approximator is suitable for these RL problems that require continuous or large discrete state spaces or action spaces. The function approximator is usually represented by a neural network whose weights are considered as the policy parameters. The input and the output of the neural network are regarded as the state and a distribution probability function for action selection, respectively. Thus the represented policy is stochastic and has the ability for exploration. The gradient updates are performed on the policy parameters. Assume θ and ρ are the vector of policy parameters and the performance of the policy (e.g., the average reward received in each step). The policy parameters can be updated approximately proportional to the gradient:

$$\Delta\theta \approx \alpha \frac{\partial \rho}{\partial \theta} \tag{7.1}$$

where α represents a positive step size. The method estimates the gradient of the average reward ρ with respect to the policy parameters θ, so as to adjust θ in a direction that maximizes the average reward. Unlike the value function approach in POMDPs that small changes in the estimated value of an action can cause big changes in the policy, small changes of θ in the policy gradient method can cause only small changes in the policy. There are two main advantages for this method. One is that the use of a function approximator for policy representation solves the generalization problems. The other is that the method can be implemented online,

which only requires low computational or memory complexity.

7.1.2 Related algorithms and applications

As far, many theoretical variants of policy gradient methods have been developed. The earliest algorithm for stochastic policies working with gradient method was Williams' REINFORCE algorithm (Williams, 1992). This algorithm brings ideas and mathematical concepts for the unbiased estimate of the gradient in policy space. These policy parameters is updated in a form as follows:

$$\Delta w_{ij} = \alpha_{ij}(r - b_{ij})e_{ij} \tag{7.2}$$

$$e_{ij} = \frac{\partial ln(g_i)}{\partial w_{ij}} \tag{7.3}$$

where α_{ij} is a learning rate factor; r is the reinforcement signal; b_{ij} is a reinforcement baseline; e_{ij} is the characteristic eligibility of the weight w_{ij}; and g_i is the probability density function determining the random generated action.

Kimura et al. (1995) extended the REINFORCE algorithm to the infinite horizon setting. The technique of discounting future rewards was introduced in the algorithm based on a stochastic gradient ascent (SGA). Baxter and Bartlett proposed REINFORCE-like algorithms, called the GPOMDP and OLPOMDP algorithms, for estimating an approximation to the gradient of the average reward (Bartlett and Baxter, 2000; Baxter and Bartlett, 2000). These algorithms remove the reliance on both a system model and the knowledge of the underlying state. The convergence of the method is proven with probability of 1. Besides, a number of similar algorithms follows, such as the GARB algorithm (Weaver and Tao, 2001) and the Meuleau's method (Meuleau et al., 2001).

The policy gradient methods have been applied to real applications and they often yield good results. Kimura et al. (1997) used the SGA algorithm to solve robot control problems on a two-link manipulator. The goal is to enable the body of the manipulator to move forward as fast as possible. They took a simple two-layer artificial neural network (ANN) for policy representation. The ANN has two input nodes whose values are the sensor-reading joint angles, and two output nodes corresponding to the turning directions of the motors in the two joints. The immediate reward is defined as the length that the manipulator moved forward in the current step. Compared to Jaakkola's method and Q-learning, the SGA method achieved good results in terms of handling hidden state and function approximation, as well as performance sensitivity when increasing the observation space.

Kohl and Stone (2004) proposed using a form of policy gradient RL to automatically search the set of possible parameters with the goal of finding the fastest quadruped gait.They chose twelve parameters that can represent a gait as a policy and adopted forward speed as the reward function. Their approach starts from an initial policy. Then several randomly generated policies near the initial one are proceeded to estimate the partial derivative of the policy's reward with respect to each parameter. After that, the increment of each parameters is calculated and thus a

superior policy is obtained. They tested the approach on the Aibo robot and got a fastest gait that significantly outperforms a variety of existing gaits for the Aibo.

Tomoyuki et al. (2009) introduced a method for acquiring energy-efficient CPG-based biped walking for a robot with knees and feet. The idea was to find a torque-free period in the swing leg control during which no torque was applied to the hip joint controller. In their method, a CPG controller was applied at the hip joint and a GARB algorithm was used to optimize the start and the end time for the torque-free period. The reward is designed as a function that it is proportional to the ratio between the walking distance and the consumed energy. Simulation results showed that after the RL, the energy consumed in the CPG-based torque-free walking was reduced by 40% compared with pure CPG-based walking.

El-Fakdi et al. proposed a field application of RL control system on an autonomous underwater robot for solving the action selection problem in cable tracking task (El-Fakdi et al., 2006; El-Fakdi and Carreras, 2008). In the application, the OLPOMDP algorithm was selected to carry out the RL of policy. A three-layer ANN with 6 input nodes, 3 hidden nodes and 5 output nodes was generated for the representation of the stochastic policy. The six input nodes correspond to the normalized states, namely the x and y positions, the rotating angle and their corresponding changing rate, while the five output nodes represent five control actions. The reward is a piecewise function that depends on the position of the image tracking. They verified the approach on the real robot $ICTINEU^{AUV}$ and got good performance by the learned policy.

Besides the aforementioned examples, the policy gradient method has applied to a variety of robot learning problems, ranging from simple control tasks (e.g. pole balancing (Riedmiller et al., 2007)) to complex learning tasks such as the applications on helicopter flying (Bagnell and Schneider, 2001), baseball swing (Peters and Schaal, 2008) and biped locomotion (Matsubara et al., 2005; Cherubini et al., 2009). Although there exist some applications about locomotion control, the policy gradient method is seldom seen in the application of adaptive locomotion. The following introduces the development of adaptive locomotion using the policy gradient method.

7.2 Design of the adaptive control system

A caterpillar-like robotic configuration which is the same to the one in Section 6.2.1 is used here as the test robot for the application of adaptive locomotion. Since the main purpose of this chapter is to remove the human intervention, the control system in this chapter is almost the same to the one used in the previous chapter 6.2, except for the substitution of the RL method for the reaction maker component.

Figure 7.1 shows the adaptive control system of the robot. It consists of the sensor processor component, the RL component, the parameter modulator component and the locomotion control component. The control system works as follows: First, the sensor processor component plays the same role as the one described in

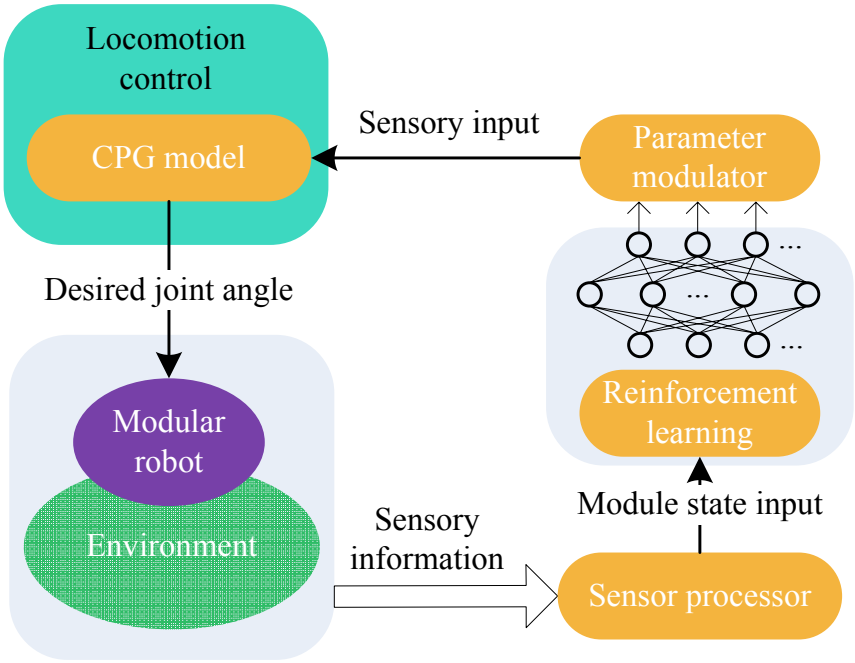


Figure 7.1: Adaptive control architecture using policy gradient reinforcement learning.

Section 6.3 in gathering the raw sensory data and converting it into module states. Then, these module states are introduced into the RL component.The RL component learns the mapping from the modules states to a certain amount of stimuli. After that, the stimuli are transmitted to the parameter modulator component for generating the sensory input. Finally, the sensory input is integrated into the CPG model in the locomotion control component. It affects the CPG output and takes effects on the joints of the robot, resulting in the adaptation of the robot to the environmental change.

Note that in the RL component the RL only focus on mapping between the module state and the amount of external stimuli. But as described in Section 6.5, the generation of sensory input in the parameter modulator component is not only related to the amount of external stimuli λ, but also involving the time variables δ_m and δ_r (see equations 6.18 and 6.19). In order to compensate for this difference, here we rewrite the equation of the sensory neuron and assume the time variables are fixed with values of $\delta_m = 6$ and $\delta_r = 12$:

$$\delta_m \dot{x}_{\{SN\}k} = -x_{\{SN\}k} + A \cdot \lambda_k \qquad \text{if} \quad \lambda_k \neq 0 \tag{7.4}$$

$$\delta_r \dot{x}_{\{SN\}k} = -x_{\{SN\}k} \qquad \text{otherwise} \tag{7.5}$$

The choose of δ_m and δ_r is to ensure fast response to external stimuli and slow recovery when there are no external stimuli afferent.

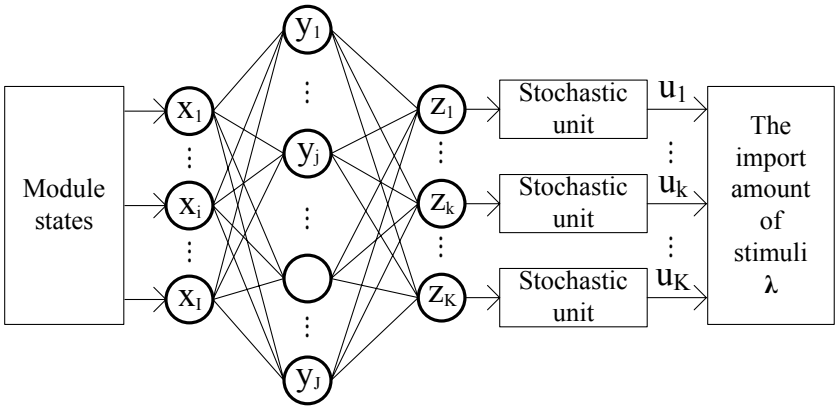


Figure 7.2: A three-layer ANN for stochastic policy representation.

7.3 Implementation of the reinforcement learning

Since the sensor processor component, the parameter modulator component and the locomotion control component have been already introduced in the previous chapter, here we only present how to implement the policy gradient RL within the control structure.

7.3.1 Problem statement

As described above, the RL component is responsible for mapping the module states into a certain amount of external stimuli. On one side, as the input of the RL component, the module states forms a discrete state space. They only contain two discrete values 1 and 0, which indicates whether the module is in "periodic touch" or "hanging in the air", respectively. On the other side, as the output of the RL component, the external stimuli are continuous and as defined in the previous chapter bounded in the range of [-1, 1]. Therefore, the RL component has a continuous action space.

Now the RL problem in our control system is clear that given a certain discrete state, the RL component is required to quickly output an approximation of the optimal action with continous values. From the introduction of the RL in the first section, it has been mentioned that a value function based RL is memory and time consuming to deal with continous state or action problems. Instead, the policy gradient method is suitable for the RL problems which require the generalization of the state or action space. Considering this advantage, we choose the policy gradient method as the basis of the RL component.

7.3.2 Neural network construction

A three-layer ANN is used here to represent the stochastic policy, as shown in Figure 7.2. In the network, the notation we used is as follows: Let x, y and z denote the units derived from the input layer, the hidden layer and the output

layer, respectively; let I, J and K be the number of unit in the three layers; and let $\mathbf{W}$ be the weight matrix consisting of all weights w in the network. For the input layer, each input unit x_i corresponds to one module state from the robot. For the hidden layer, the activation function used for the hidden unit y_j is represented in a linear form:

$$y_j = \sum_{i=1}^{I} x_i w_{ij} \tag{7.6}$$

For the output layer, the activation function used for the output unit z_k is designed as a hyperbolic tangent type:

$$z_k = \frac{1 - e^{net_k}}{1 + e^{net_k}} \tag{7.7}$$

where net_k is the linear representation of the units in the hidden layer:

$$net_k = \sum_{j=1}^{J} y_j w_{jk} \tag{7.8}$$

In addition, the network is designed to have deterministic hidden units but stochastic output units. Here we assume each output unit is a Gaussian unit (Williams, 1992). To determine the output of such a unit, it would first calculate the mean and the standard deviation deterministically and then draw the output from the normal distribution. Use of these stochastic output units makes sense because their randomness allows any necessary exploration to take place. Especially for the Gaussian unit, control over the standard deviation is equal to control over the unit's exploration behavior.

In the kth Gaussian unit, let the original output z_k denote the mean, let another variable σ_k denote the standard deviation, let u_k denote the actual output value (namely u_k is λ_k in equation 7.4, representing the import amount of stimuli), and let g_k denote the probability mass function that determines u_k as a function of the input vector $\mathbf{x}$ and the weight matrix $\mathbf{W}$:

$$\begin{aligned} g_k &= Pr\{u_k | z_k = f(\mathbf{x}, \mathbf{W}), \sigma_\mathbf{k}\} \\ &= \frac{1}{\sqrt{2\pi}\sigma_k} e^{-(u_k - z_k)^2 / 2\sigma_k^2} \end{aligned} \tag{7.9}$$

Thus the real output u_k can be sampled from the Gaussian distribution g_k:

$$u_k = z_k + \sigma_k \cdot n \tag{7.10}$$

where $n \sim N(0, 1)$. $N(0, 1)$ is a Gaussian distribution which has a mean of 0 and a variance of 1.

7.3.3 Weights update

Since the randomness of all the Gaussian units is independent and identically distributed, the overall probability mass function determining the input-output behavior of the whole network is dependent on the production of individual probability

mass function, namely:

$$g = \prod_{k=1}^{K} g_k \tag{7.11}$$

According to the definition of characteristic eligibility in 7.3, the characteristic eligibility for every weight of the network has the form:

$$e = \frac{\partial ln(g)}{\partial w} = \sum_{k=1}^{K} \frac{\partial ln(g_k)}{\partial w} \tag{7.12}$$

The chain rule is used to compute the characteristic eligibility for a particular weight in the network. For the weights between the hidden layer and the output layer, the characteristic eligibility is given by:

$$\begin{aligned} e_{jk} &= \sum_{k'=1}^{K} \frac{\partial ln(g_{k'})}{\partial w_{jk}} = \frac{\partial ln(g_k)}{\partial w_{jk}} \\ &= \frac{\partial ln(g_k)}{\partial z_k} \cdot \frac{\partial z_k}{\partial net_k} \cdot \frac{\partial net_k}{\partial w_{jk}} \end{aligned} \tag{7.13}$$

By differentiating the equations 7.9, 7.7 and 7.8 and substituting the results in equation 7.13, the characteristic eligibility is got:

$$e_{jk} = \frac{(u_k - z_k) \cdot (1 - {z_k}^2) \cdot y_j}{2{\sigma_k}^2} \tag{7.14}$$

For the weights between the input layer and the hidden layer, due to the deterministic portions of them, their characteristic eligibility can be calculated based on the previous obtained characteristic eligibility:

$$\begin{aligned} e_{ij} &= \sum_{k'=1}^{K} \frac{\partial ln(g_{k'})}{\partial w_{ij}} \\ &= \sum_{k'=1}^{K} \frac{\partial ln(g_{k'})}{\partial z_{k'}} \cdot \frac{\partial {z_k}'}{\partial net_{k'}} \cdot \frac{\partial net_{k'}}{\partial y_j} \cdot \frac{\partial y_j}{\partial w_{ij}} \\ &= \sum_{k'=1}^{K} e_{jk'} \cdot \frac{\partial net_{k'}}{\partial y_j} \cdot \frac{\partial y_j}{\partial w_{ij}} / (\frac{\partial {net_k}'}{\partial w_{jk'}}) \end{aligned} \tag{7.15}$$

Likewise, differentiating the equations 7.6 and 7.8 yields:

$$e_{ij} = \frac{x_i}{y_j} \sum_{k'=1}^{K} e_{jk'} \cdot w_{jk'} \tag{7.16}$$

Finally, according to the equation 7.2, all the weights can be updated.

7.3.4 Exploration

As mentioned in the previous section, the variable σ contributes to the exploration behavior for each Gaussian unit. It is able to narrow or broaden the search around the mean of the Gaussian unit. Let r denote the reward and let $\overline{r}$ denote the averaged reward. The searching behavior of σ in each Gaussian unit is associated with the reward and the sampled output. According to equation 7.9 and 7.11, the characteristic eligibility of σ_k for the kth Gaussian unit is given by:

$$e_{\sigma_k} = \frac{\partial ln(g)}{\partial \sigma_k} = \frac{(u_k - z_k)^2 - {\sigma_k}^2}{{\sigma_k}^3} \tag{7.17}$$

Then the update of σ_k has the form:

$$\Delta\sigma_k = \alpha(r - \overline{r})\frac{(u_k - z_k)^2 - {\sigma_k}^2}{{\sigma_k}^3} \tag{7.18}$$

Since the discrepancy of r and $\overline{r}$ is not normalized, σ may change dramatically or even become negative. An alternative is to use a small fixed increment $\Delta\sigma$ for the update of σ. For example, for the kth Gaussian unit, σ_k is updated according to four situations:

- If the real output u_k sampled by equation 7.10 leads to an increase of the reward (e.g., $r > \overline{r}$), and u_k lies within one standard deviation of the mean z_k (namely, $|u_k - z_k| < \sigma_k$, which is equal to $e_{\sigma_k} < 0$), then σ_k decreases by $\Delta\sigma$ ($\Delta\sigma > 0$);
- If $r > \overline{r}$ and $e_{\sigma_k} \geq 0$, then σ_k is updated with an increment of $\Delta\sigma$;
- Similarly, if $r < \overline{r}$ and $e_{\sigma_k} < 0$, then σ_k decreases by $\Delta\sigma$;
- If $r < \overline{r}$ and $e_{\sigma_k} \geq 0$, then σ_k increases by $\Delta\sigma$.

In general, the updates of both the mean and the standard deviation determine the searching direction of the Gaussian unit. One on hand, the update of the mean that is equal to the update of the weight in the network depends on the change of the reward. An increasing reward leads to the update of the mean moving toward to the sampled output. In contrary, an decreasing reward causes the update of the mean moving away from the sampled output. On the other hand, as described above, the change of the reward and the sampled output together have a definite relationship for the update of the standard deviation, which determines whether narrowing or broadening the search around the mean. Thus, the Gaussian unit can converge the local maximum by means of the two types of updates.

7.4 Simulation

In this section, we not only investigate the feasibility of the RL based control system in realizing adaptive locomotion through simulation experiment, but also compare its performance with the genetic algorithm (GA) based adaptive locomotion (see details in Section 6.7.1).

Algorithm 1 Episodic REINFORCE algorithm

Given:

The weights $\mathbf{W}$ in the ANN;

The policy probability function g with the standard deviation $\boldsymbol{\sigma}$;

Algorithm parameters:

The characteristic eligibility $\mathbf{e_w}$ for the weights in the ANN;

The characteristic eligibility $\mathbf{e}_\sigma$ for the standard deviation $\boldsymbol{\sigma}$;

The number of output units K in the ANN;

The number of iterations n in one episode;

1: Initialize the variables $\mathbf{e_w}$ and $\mathbf{e}_\sigma$ to zero;
2: **for** $i = 1 \rightarrow n$ **do**
3: Receive the state $\mathbf{x_i}$;
4: Generate the external stimuli according to the ANN with a probability $g(\cdot|\mathbf{x_i}, \mathbf{W}, \boldsymbol{\sigma})$;
5: Accumulate the characteristic eligibility:
$\mathbf{e_w} \leftarrow \mathbf{e_w} + \frac{\partial ln(g)}{\partial \mathbf{W}}$
$\mathbf{e}_\sigma \leftarrow \mathbf{e}_\sigma + \frac{\partial ln(g)}{\partial \boldsymbol{\sigma}}$
6: **end for**
7: Receive reward R;
8: Calculate the reinforcement baseline:
$B \leftarrow B + (R' - B)/j$
where R' is the reward in the last episode and j is the episode number;
9: Update the weights $\mathbf{W}$ as:
$\Delta \mathbf{W} \leftarrow \alpha(R - B)\mathbf{e_w}$
10: Update the standard deviation $\boldsymbol{\sigma}$ as:
for $j = 1 \rightarrow K$
if$((R - B)e_{\sigma j} > 0)$ **then**
$\sigma_j \leftarrow \sigma_j + \Delta\sigma$;
else
$\sigma_j \leftarrow \sigma_j - \Delta\sigma$;
end if
end for
11: Return the policy parameters $\mathbf{W}$ and $\boldsymbol{\sigma}$.

7.4.1 Episodic learning

In the ODE environment, we create a caterpillar-like robot with 6 pitch-pitch connected modules and build the same environment as the one described in Figure 6.7. In addition, the caterpillar-like robot uses the same linear gait with an amplitude of 20° and a phase difference of 120° for forward motion.

The learning task is to find an optimal policy that guides the robot to adaptively climb over these slopes. In our learning task, a three-layer ANN with 7 input units,

Table 7.1: Parameters in the simulation

Parameters	Value	Description
α	0.001	Learning rate
$\Delta\sigma$	0.01	The increment of the standard deviation
n	100000	Time steps in one episode

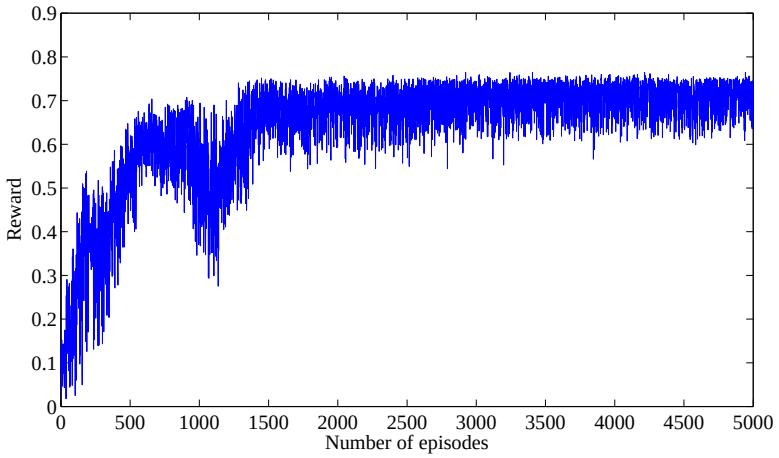


Figure 7.3: The learning curve of the episodic REINFORCE algorithm.

10 hidden units and 6 output units is used to generate the stochastic policy. The policy is trained in an episodic manner. One learning episode is defined as a simulation running under the current policy with a fixed amount of time steps. When the episode ends, a reward is obtained and the current policy is updated according to the policy gradient method. This process will be repeated until the predefined number of episodes is reached. Algorithm 1 shows the episodic REINFORCE algorithm based on the constructed ANN in Section 7.3.2, and Table 7.1 shows the corresponding variables used in the algorithm.

In the simulation, the weights $\mathbf{W}$ are initialized to random values between ± 0.3 and the standard deviation σ for each Gaussian unit is unified initialized to 0.1. For ease of comparison between the RL based locomotion and the GA based adaptive locomotion, the reward function is designed the same as the fitness function in the GA based adaptive locomotion (see details in Section 6.6), which rewards both the average climbing velocity and the average touch on the terrain in parallel with a proportional variable $\eta = 0.85$ (see equation 6.20).

The number of episodes to be done is set to 5000. Figure 7.3 shows the reward for every episode when trained by the episodic REINFORCE algorithm. It is clear from the figure that the performance of the robot under the trained policy is increased with the growth of the number of episode. After about 1500 episodes, an appropriate

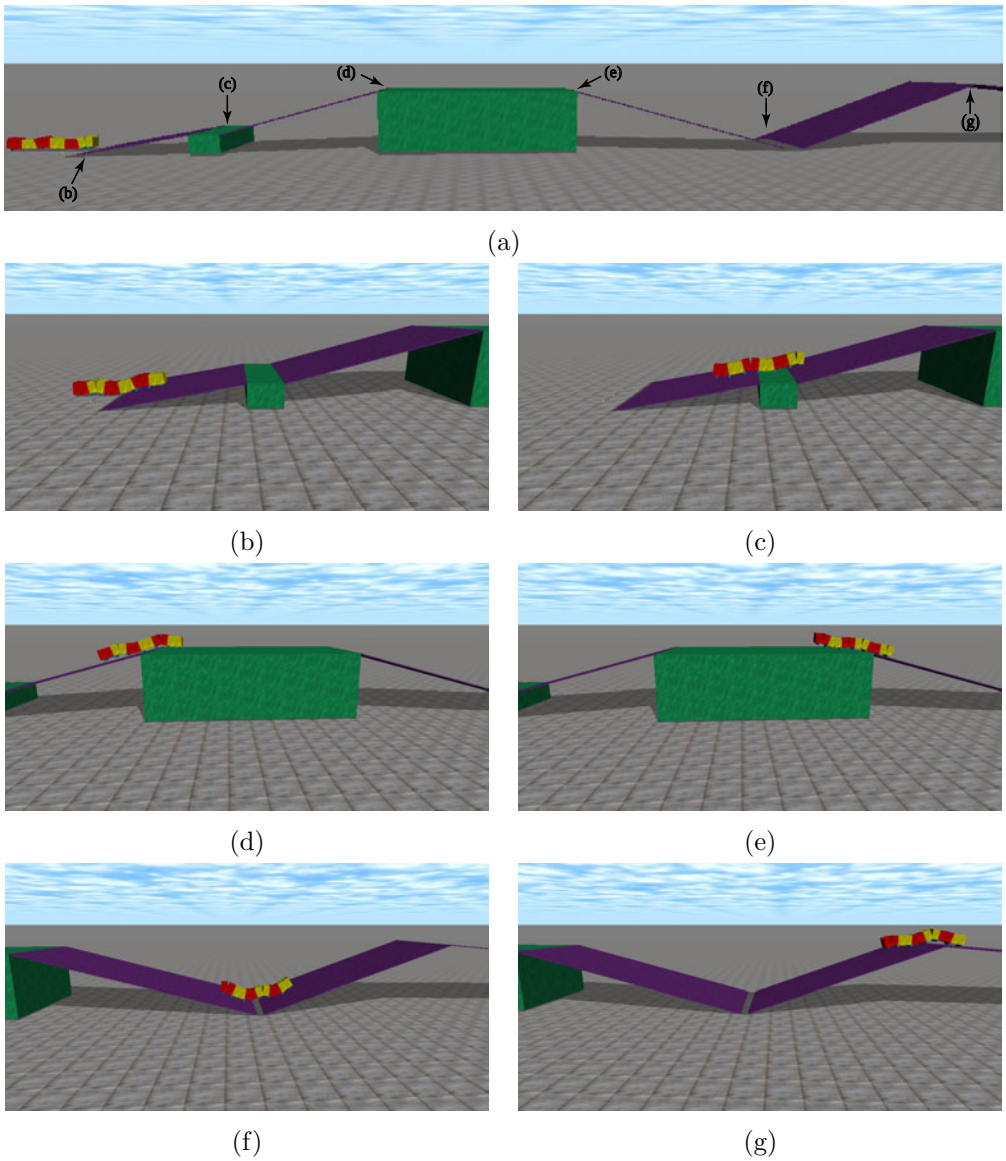


Figure 7.4: Simulation for RL based adaptive locomotion. (a) The overall environment which is the same to the one constructed in the simulation of GA based adaptive locomotion (see Figure 6.9). (b)–(g) Scenes of adaptative caterpillar-like locomotion. The robot succeeds to climb over these slopes by using the learned policy.

Table 7.2: Performance comparison

Performance	GA based control system	RL based control system
Reward	0.89	0.76
Average speed (m/s)	0.089	0.074
Average touch (%)	91.01	84.17

policy which enable the robot to climb over all the slopes is acquired. Figure 7.4 represents the climbing process of the robot using the final obtained policy. The simulation result shows that the robot climbs well at every slope except the slopes in 7.4(d) and 7.4(g), where the robot climbs unstable and wastes much time balancing the climbing behavior. Even though the robot performs inefficient on the climbing of the two slopes, it finally succeeds to climb over the whole uneven environment. All the simulation data of the slope climbing experiment can be found in Appendix B.

From the simulation, we conclude that the RL based control system is feasible in realizing the adaptive locomotion for limbless robots.

7.4.2 Performance comparison

As mentioned above, for ease of comparison, this simulation almost copies the environmental setting from the simulation in Section 6.7.1), including the robot, the scenario, the basic locomotive pattern and the reward function. The main difference for the two simulations is their control systems. In the GA based simulation in Section 6.7.1, since the control system contains some predefined reaction rules based on the module states, it only uses GA to optimize the speed and the amount of external stimuli. While in this simulation, the control system removes these predefined reaction rules and uses RL method to learn the mapping between the module states and the external stimuli.

Here we compare the best performance between the two control systems. For the GA based control system, the best performance is achieved when the number of neighboring module states is equal to 6 ($l = 6$). The robot with the GA based control system can get a reward of 0.89. For the RL based control system, the robot aquires a best reward of 0.76 when using the finally obtained policy. Talble 7.2 lists the performance comparison of the two simulations. Compared with the robot in the GA based simulation, the robot in the RL based simulation performs an inferior average speed as well as an inferior average touch. It is observed in the RL based simulation that the robot works inefficiently when it climbs from an upslope to a relative flat terrain, e.g., the position in Figure 7.4(d) and 7.4(g). The robot does not obtain an optimal response but learns an sub-optimal response to deal with such a situation. Thus, the robot does not adapt to the terrain very well

and wastes much time climbing over it.

The following analizes why the performance of RL based control system is inferior to that of GA based control system. First, the learning parameters in the two control system are different. On the one hand, the RL based control system needs to learn the stochastic policy, i.e., 130 ($= 7 \times 10 + 10 \times 6$) weights parameters in the three-layer ANN. On the other hand, the GA based control system only needs to optimize 5 parameters for each module, i.e., 30 parameters in total. Thus, the parameter space in the RL based control system is much more bigger than that in the GA based control system. Second, the number of trials is different. For the RL based control system, the simulation takes 5000 episodes for policy learning. While for the GA based control system, the simulation runs 126 generations. Each generation contains 100 candidates. The simulation is equal to take 12600 episodes in total. This means the GA based control system learns much longer than the RL based control system does and thus gets more information in the parameter space. Third, the GA based control system itself contains predefined reaction rules, which reduces the whole searching space for optimal solutions. Fourth, from the algorithm point of view, policy gradient RL method learns slower than normal RL methods and often converge to local optimum of the expected reward. In contrast, since GA searches parallel from a population of candidates, it has the ability to avoid being trapped in the local optimal solution. Fifth, the initial values of the policy and the learning rate may also affect the final result of the RL based control system. In summary, the above is the five possible reasons resulting in the performance difference.

Although the performance of the RL based control system is not as good as that of the GA based control system, the original intention of this chapter is to remove these predefined reaction rules. From the analysis of the performance comparison, it has found that there are still possible ways for improving the performace of the RL based control system, such as decreasing the number of learning parameters, increasing the number of trials and trying to seach multiple policies in parallel. The furture work will investigate these methods and try to find a solution for improving the performance of the RL based control system.

7.5 Summary

This chapter presents the development of adaptive locomotion using policy gradient RL method. The aim of the development is to remove the human intervention in the control system developed in last chapter. First, the policy gradient RL method as well as the related applications is investigated. The policy gradient RL method is found to be suitable for continuous RL problems and often yields good result. Then, the RL based control system is proposed. The original reaction maker component is replaced by a RL component. Next, the implementation of the policy gradient RL method via an ANN is introduced in detail. After that, a simulation is carried out using the same enviromental setting of simulation in last chapter. The feasibility

of the RL based control system is verified in realizing the adaptive locomotion for limbless robots. Finally, the performance is compared between the RL based control system and the GA based control system. The reasons for the performance difference are also analyzed, which provides possible solutions for improving the performance of the RL based control system in future work.

Chapter 8

Conclusions and Future Work

Contents

8.1 Conclusions

This thesis presents a hierarchical control architecture as steps toward developing limbless robots capable of 3D locomotion, fast reflex responses, as well as sophisticated reaction to environmental stimuli. The three main functionalities developed in the control system all serve important roles. First, the control system in limbless robots has the ability to generate 3D locomotive patterns so that the robots can move in environments. Second, the control system allows the limbless robots to fast respond to external stimuli via a reflex mechanism. And third, the control system provides the limbless robots with a mean for achieving deliberate goal-oriented behavior. Through simulations and experiments, the hierarchical control architecture is validated useful to ensure the limbless robots to intelligently and autonomously move in complex environments.

As a highlight of the thesis, the hierarchical control architecture is designed to have two levels of control scheme. On the fine-scale level, a CPG controller is applied for gait generation and a reflex mechanism designed as a special extended pathway of the CPG model is used for fast response to external stimuli. On the large-scale level, learning algorithms are applied to study the mapping between the parameters of the CPG model and the sensory information gathered from the environment, so as to generate deliberate response behaviors to environmental changes. From the control point of view, the fine-scale level provides the large-scale level with the basic mobile ability, while the large-scale level in turn modifies the behavior generated by the fine-scale level by means of learned reaction rules. Therefore, the two levels of the control architecture are closely coupled with each other. It is noted that the CPG model in the control architecture plays such a key role in organically connecting the two levels taking advantage of its biological properties. This is the main reason we choose CPG based method rather than other control methods in the hierarchical control architecture.

The main contribution of this thesis is the design of the CPG model inspired by the neuronal circuit diagram in the spinal cord of lampreys. We design the oscillator

of the CPG model at the connectionist level and describe these interneurons with a set of sigmoid functions and leaky integrators. Based on the oscillator model, we further design two types of CPG circuits, called the chained inhibitory CPG circuit and the cyclic inhibitory CPG circuit. As the kernel of the hierarchical control architecture, the CPG model possesses four interesting characteristics:

- First, traditional connectionist CPG models such as the Matsuoka's model are usually lack of independent control parameters for online modulation. But through numerical simulations, the proposed CPG model is verified to have explicit and uncoupled control parameters for output signal modulation, including the modulation of amplitude, period, phase difference and offset.

- Second, the proposed CPG model has rich dynamics of oscillatory activities. It can not only generate phase fixed oscillatory activity, but also expand the oscillatory activity to synchronization and maintenance activities. Even though some other oscillators, such as Hopf oscillators and phase oscillators could realize synchronization activity, they fail to achieve maintenance activity due to the limit cycle behavior. The expansion of the oscillatory activity would be beneficial to further develop limbless gaits.

- Third, the proposed CPG model is easy to add the sensory reflex mechanism. Although other CPG models can also realize the same mechanism, our method is more simple and natural. This is because the proposed CPG model is developed at the neuronal level and the concept of reflex arcs used for realizing the reflex mechanism takes effect also on the neuronal level.

- Fourth, the proposed CPG model allows the integration of sensory feedback for achieving adaptive limbless locomotion. Taking advantage of theoretical support from biological findings in lampreys, additional sensory neurons are added into the CPG model, so that sensory information can be fed back into the CPG model via the sensory neurons.

Because of the above properties of the CPG model, the development of 3D locomotion, fast reflex responses and deliberate response behavior becomes possible.

The second contribution is the design of four types of 3D limbless gaits, namely side winding, rolling, turning and flapping. Even though these limbless gaits have been investigated by researchers using kinematics or sinusoidal function based methods, it is rarely seen to use CPG based method to imitate the 3D limbless gaits. Hence, we emphasis the design of CPG circuits used for generating the limbless gaits. For each locomotion pattern, two CPG circuits are required to control the pitch and yaw joints on the robot, respectively. In order to generate cooperative locomotion patterns, the necessary conditions for the cooperation between the two CPG circuits are analyzed in detail. Since the proposed CPG model provides explicit control parameters for output modulation, the corresponding gait circuit has the ability to modulate the resulting motion. Through numerous simulations, we investigate the parameters that are necessary for achieving fast limbless locomotion.

Furthermore, we implement the four types of limbless gaits and verify the effectiveness of the proposed CPG circuits in generating limbless locomotion patterns.

Another contribution of this thesis is the realization of integrating the sensory reflex mechanism. Biological research shows that sensory neurons exist in lampreys and play a role in bridging external stimuli to interneurons. Since our CPG model is based on the lamprey's neural circuit, to design the reflex mechanism in a natural fashion, our CPG model is extended by adding sensory neurons into the model. Moreover, to integrate short pathways to make quick responses to external stimuli, reflex arcs that involve the sensory neurons are established. The feasibility is confirmed by a ball hitting experiment and a corridor passing experiment. The robot embedded with the sensory reflex mechanism is able to respond actively and correctly towards external stimuli, which further endows the robot with the ability to behave adaptively to different environmental conditions.

The last contribution lies in the development of adaptive limbless locomotion. We refine the large-scale level of the hierarchical control architecture and propose a framework of closed-loop control for achieving adaptive limbless locomotion. The framework is of course based on our CPG model. On the one hand, the CPG model is used for gait generation. On the other hand, the sensory neurons attached to the CPG model serve as a bridge between external stimuli and the output of the CPG model. Thus, the main difficulty of the framework is to find a mapping between the external stimuli and the desired sensory input, so as to shape the output of the CPG model to realize adaptive behavior. Two types of learning methods are tested for resolving the problem. The first method is based on the genetic algorithm (GA). By quantifying the neighboring module states on the limbless robot and predefining some sketchy reaction strategies, the GA can evolve the parameters in these reaction strategies and finally obtain the specific reaction rules. In order to remove the human intervention, i.e. those predefined reaction strategies, an alternative is to use policy gradient reinforcement learning (RL) method. An artificial neural network is designed as the policy of the RL based method. The policy directly learns the mapping between the module states and the desired sensory input, eliminating the need for the predefined reaction strategies. The method updates the policy along the gradient direction with respect to the expected reward and finally obtains an optimal policy that maximizes the expected reward. The feasibility for both of the two methods is confirmed by a slope climbing experiment. Furthermore, the performance of adaptive limbless locomotion under the same experiment is compared between the two methods. The reasons of performance difference are also analyzed, which would be helpful to improve the control system in further work.

8.2 Future work

The work presented in this thesis has been finished. Nevertheless, it does not mean the end of the research on limbless robots. Actually, there are some limitations in the proposed hierarchical control architecture. First, the CPG model has some

shortages itself. For example, the ranges of relevant characteristics of the CPG output are not as wide as those generated by mathematical CPG models. Second, the reflex mechanism is only applied to the yaw modules of the limbless robots, which limits the reactive behaviors. Third, the adaptive limbless locomotion is only realized on limbless robots with pitch modules. If it is applied to another limbless robot with pitch-yaw connection, the responses to environmental changes will become strange and the robot will fail to cross the terrain.

The future work is not only to overcome those limitations mentioned above, but also to promote the performance of the control system. First, the proposed CPG model may have more interesting properties than those we have found. More research on the connection between oscillators, the integration of sensory neurons, the connection between CPG circuits as well as the robustness of the CPG circuits should be conducted. Second, In this thesis we only emphasize how fast and correct the response is, but do not describe what the reflex response is when emergency situation happens to the robot. Therefore, the reflex mechanism should be further developed to guarantee the locomotion safety and reliability for emergency cases. Third, the study of adaptive limbless locomotion should be more in depth. For example, energy consumption should be considered during the limbless locomotion, which would be helpful to make the locomotion not only fast and stable, but also efficient.

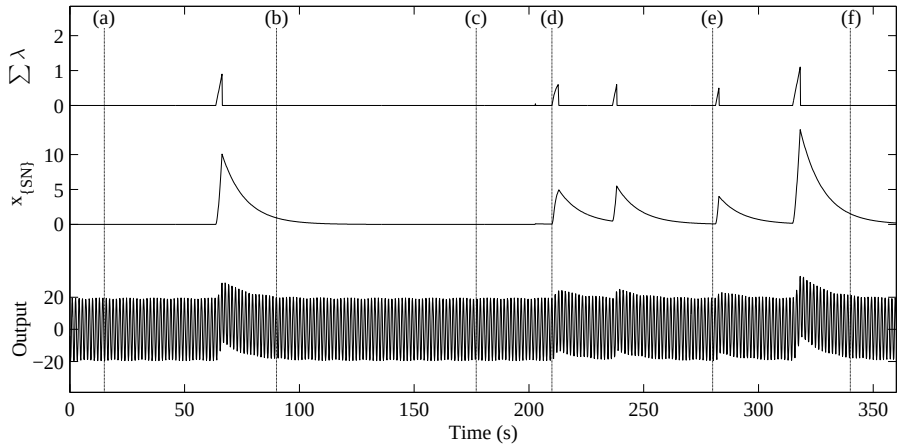
Besides the control system, limbless robots are also expected to expand the ability to interact with more unstructured outdoor environments, such as obstacles, pipes and ditches. Currently, we are designing new modular robot called CUBO robot which can not only add auxiliary equipments, such as wheels, suckers, lights and cameras, but also integrate different onboard sensors for measuring tactile force, torque, posture, voltage, etc. As the next phase of the research, we plan to use the limbless robot to perform more complicated tasks such as grasping manipulation with locomotion capability in unstructured outdoor environments.

The locomotion control of limbless robots is a fascinating topic. There is still a long way to go before the free movement of limbless robots in natural environments comes true. It is hoped that this thesis is helpful to further research on the topic of limbless locomotion.

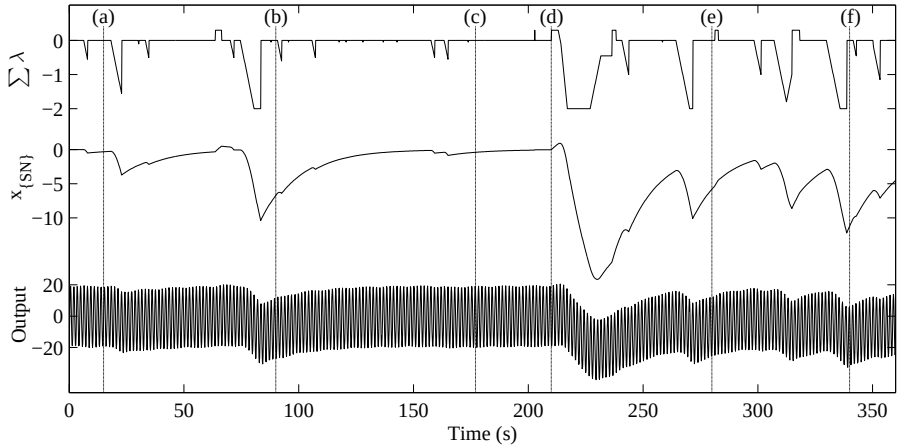
Appendix A

On-site experimental data

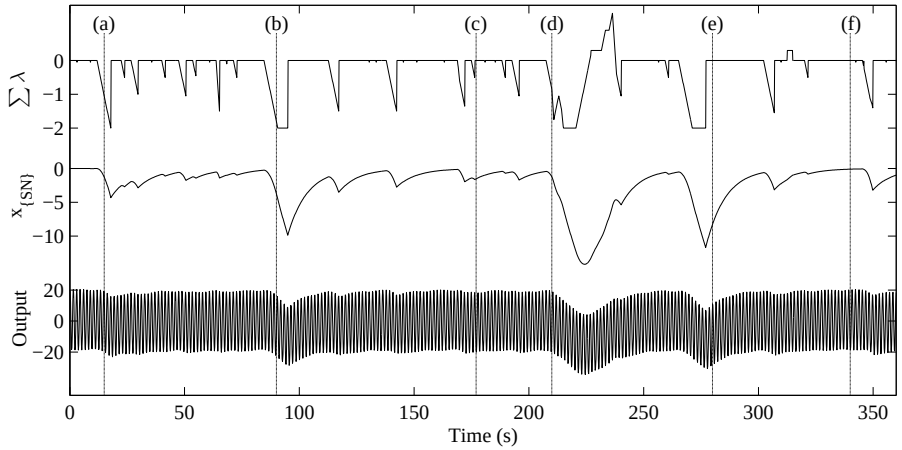
This appendix provides all the tracked data of the on-site experiment in Chapter 6. The following figures illustrate the variation of each module of the limbless robot. Each figure contains three curves from the top to the bottom, representing the afferent stimuli, the sensory input and the joint output, respectively. Furthermore, the vertical lines in each figure represent the time when the robot is climbing over slopes, with whose labels (a)–(f) corresponding to Figure 6.10 (a)–(f).



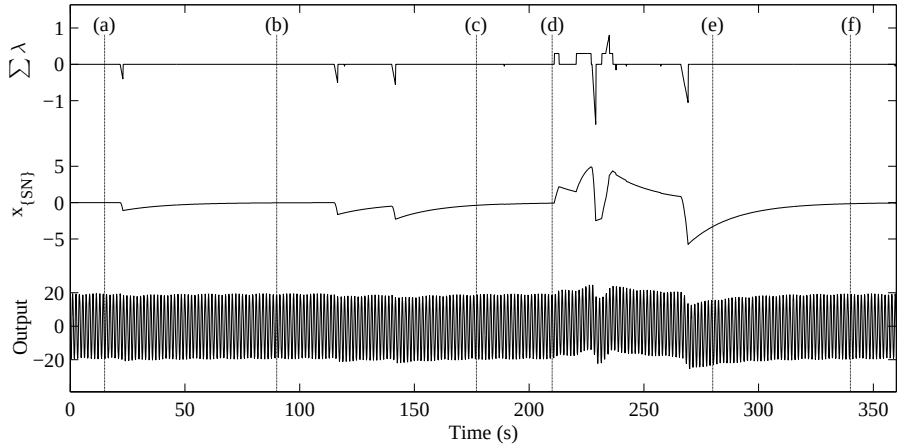
(a) Module 1



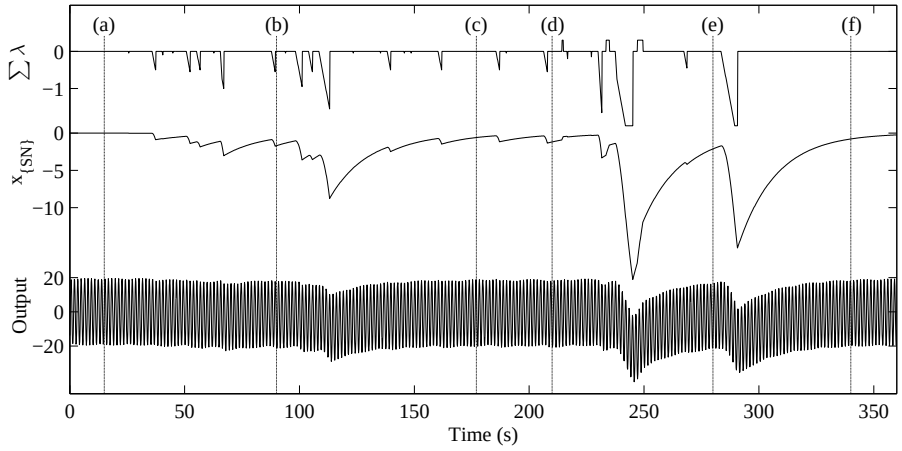
(b) Module 2



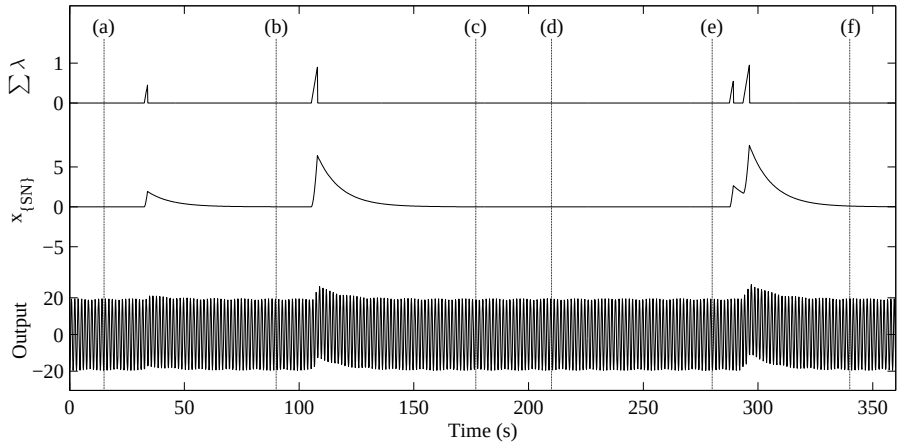
(c) Module 3



(d) Module 4



(e) Module 5



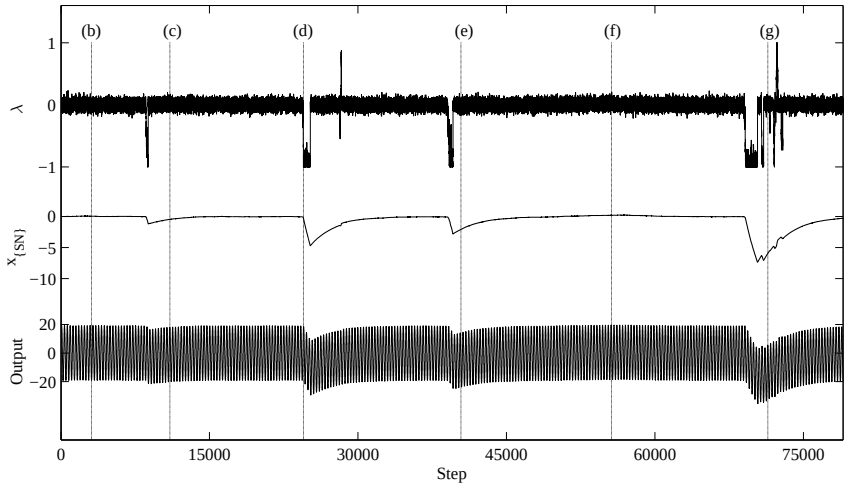
(f) Module 6

Figure A.1: On-site experimental data of the limbless robot from the head to the tail in the slope climbing experiment.

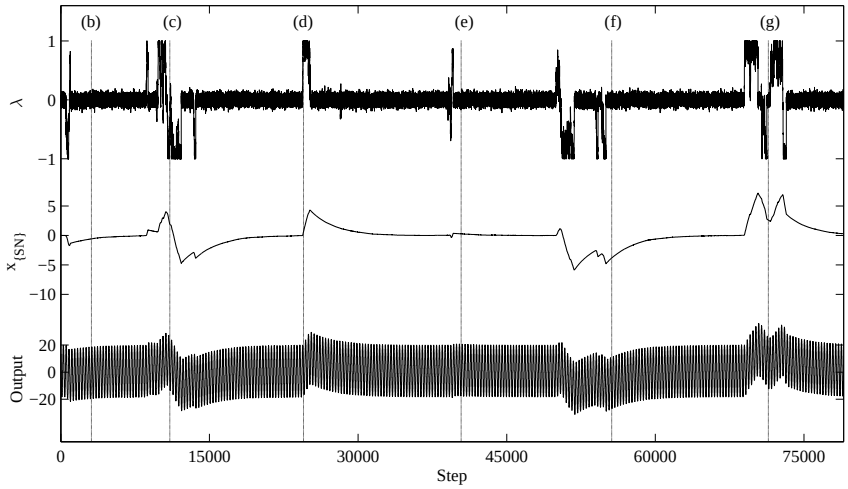
Appendix B

Simulation data

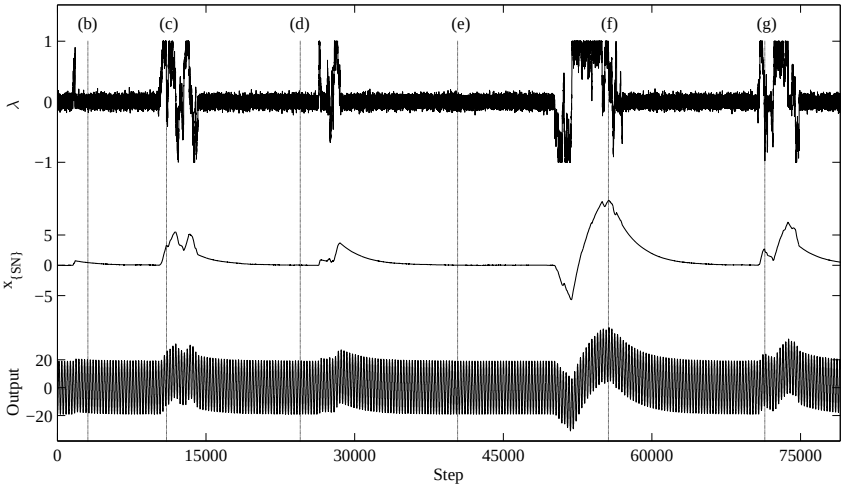
The following is the simulation data for the slope climbing experiment in Chapter 7. Each figure records the variation of the module of the limbless robot. The three curves in each figure from the top to the bottom represent the afferent stimuli, the sensory input and the joint output, respectively. The vertical lines in each figure represent the time when the robot is climbing over slopes, with whose labels (b)–(g) corresponding to Figure 7.4(b)–(g).



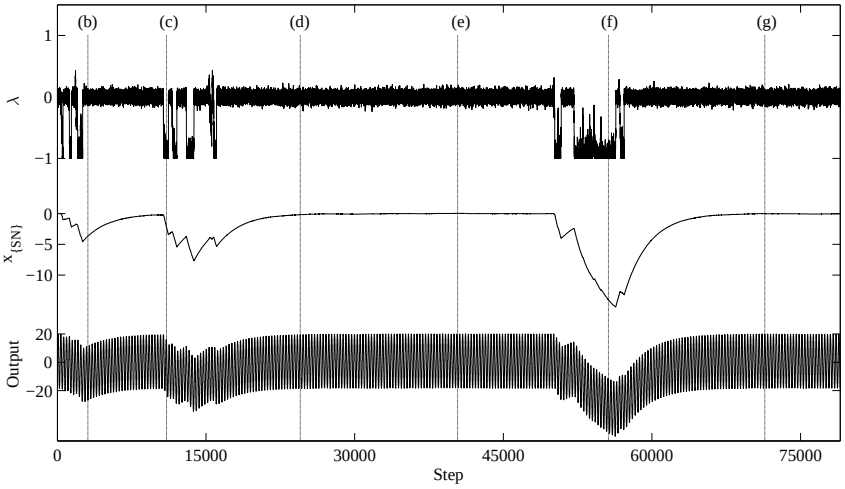
(a) Module 1



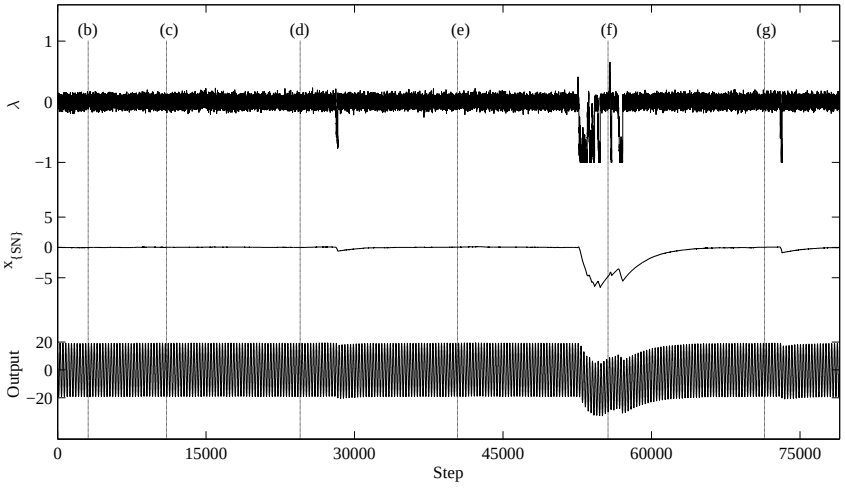
(b) Module 2



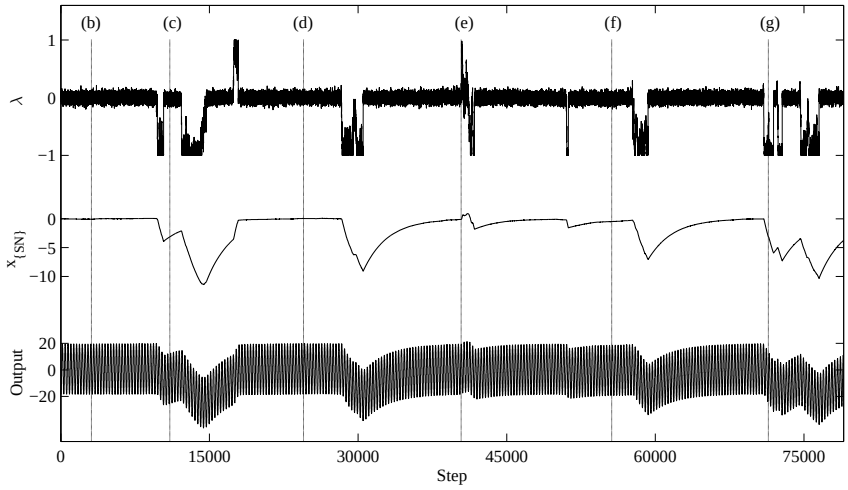
(c) Module 3



(d) Module 4



(e) Module 5



(f) Module 6

Figure B.1: Simulation data of the limbless robot from the head to the tail in the slope climbing experiment.

Bibliography

Aoi, S. and Tsuchiya, K. (2005). Locomotion control of a biped robot using nonlinear oscillators. *Autonomous Robots*, 19(3):219–232.

Aoi, S. and Tsuchiya, K. (2006). Stability analysis of a simple walking model driven by an oscillator with a phase reset using sensory feedback. *IEEE Transactions on Robotics*, 22(2):391–397.

Arai, M., Tanaka, Y., Hirose, S., Kuwahara, H., and Tsukui, S. (2008). Development of "souryu-iv" and "souryu-v": serially connected crawler vehicles for in-rubble searching operations. *J. Field Robot.*, 25(1-2):31–65.

Arena, P., Fortuna, L., Frasca, M., and Patane, L. (2005a). A cnn-based chip for robot locomotion control. *IEEE Transactions on Circuits and Systems*, 52(9):1862–1871.

Arena, P., Fortuna, L., Frasca, M., Patane, L., and Vagliasindi, G. (2005b). Cpg-mta implementation for locomotion control. In *Proceedings of IEEE Int. Symp. Circuits Syst.*, volume 4, pages 4102 – 4105, Kobe, Japan.

Bagnell, J. and Schneider, J. G. (2001). Autonomous helicopter control using reinforcement learning policy search methods. In *Proceeding of 2001 IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1615–1620.

Bailey, S. (2004). *Biomimetic control with a feedback coupled nonlinear oscillator: insect experiments, design tools, and hexapedal robot adaptation results.* PhD thesis, Dept. of Mechanical Engineering, Stanford University.

Bartlett, P. L. and Baxter, J. (2000). Stochastic optimization of controlled partially observable markov decision processes. In *Proceedings of the 39th IEEE Conference on Decision and Control*, volume 1, pages 124–129.

Baxter, J. and Bartlett, P. L. (2000). Reinforcement learning in pomdp's via direct gradient ascent. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, pages 41–48. Morgan Kaufmann.

Bay, J. S. and Hemami, H. (1987). Modeling of a neural pattern generator with coupled nonlinear oscillators. *IEEE Transactions on Biomedical Engineering*, 34(4):297–306.

Borenstein, J., Granosik, G., and Hansen, M. (2005). The omnitread serpentine robot–design and field performance. In *Proc. SPIE Defense and Security Conference, Unmanned Ground Vehicle Technology VII*, pages 324–332, Orlando, FL.

Borenstein, J. and Hansen, M. (2007). Omnitread ot-4 serpentine robot–new features and experiments. In *Proc. SPIE Defense and Security Conference, Unmanned Systems Technology IX*, Orlando, FL.

Borenstein, J., Hansen, M., and Borrell, A. (2007). The omnitread ot-4 serpentine robot-design and performance: Field reports. *J. Field Robot.*, 24(7):601–621.

Brodal, P. (1998). *The central nervous system: structure and function.* Oxford University Press, USA, second edition edition.

Buchanan, J. T. (1992). Neural network simulations of coupled locomotor oscillators in the lamprey spinal cord. *Biological Cybernetics*, 66:367–374.

Buchanan, J. T. and Grillner, S. (1987). Newly identified 'glutamate interneurons' and their role in locomotion in the lamprey spinal cord. *Science*, 236(4799):312–314.

Busoniu, L., Ernst, D., De Schutter, B., and Babuska, R. (2011). Approximate reinforcement learning: an overview. In *Proceedings of the 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL 2011)*, pages 1–8.

Castano, A., Shen, W.-M., and Will, P. (2000). Conro: towards deployable robots with inter-robots metamorphic capabilities. *Auton. Robots*, 8(3):309–324.

Chatterjee, R. and Matsuno, F. (2001). Use of single side reflex for autonomous navigation of mobile robots in unknown environments. *Robotics and Autonomous Systems*, 35(2):77 – 96.

Cherubini, A., Giannone, F., Iocchi, L., Lombardo, M., and Oriolo, G. (2009). Policy gradient learning for a humanoid soccer robot. *Robotics and Autonomous Systems*, 57(8):808–818.

Chirikjian, G. and Burdick, J. (1995). The kinematics of hyper-redundant robot locomotion. *IEEE Transactions on Robotics and Automation*, 11(6):781–793.

Collins, J. and Richmond, S. (1994). Hard-wired central pattern generators for quadrupedal locomotion. *Biological Cybernetics*, 71:375–385.

Creed, R. S. (1972). *Reflex activity of the spinal cord.* Clarendon Press.

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

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

Crespi, A., Badertscher, A., Guignard, A., and Ijspeert, A. J. (2005b). Swimming and crawling with an amphibious snake robot. In *Proceedings of the 2005 IEEE international conference on robotics and automation (ICRA)*, pages 3024–3028, Barcelona, Spain.

Crespi, A. and Ijspeert, A. J. (2006). 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)*, pages 19–27, Brussels, Belgium.

Crespi, A. and Ijspeert, A. J. (2008). Online optimization of swimming and vrawling in an smphibious dnake robot. *Trans. Rob.*, 24(1):75–87.

Dowling, K. J. (1997). *Limbless locomotion: learning to crawl with a snake robot.* PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Duff, D. G., Yim, M. H., and Roufas, K. D. (2001). Evolution of polybot: a modular reconfigurable robot. In *Proceedings of COE/Super-Mechano-Systems Workshop*, Tokyo, Japan.

Ekeberg, O. (1993). A combined neuronal and mechanical model of fish swimming. *Biological Cybernetics*, 69:363–374.

El-Fakdi, A. and Carreras, M. (2008). Policy gradient based reinforcement learning for real autonomous underwater cable tracking. In *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3635–3640.

El-Fakdi, A., Carreras, M., and Ridao, P. (2006). Towards direct policy search reinforcement learning for robot control. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3178–3183.

Endo, G., Togawa, K., and Hirose, S. (1999). Study on self-contained and terrain adaptive active cord mechanism. In *Proceedings of the 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 1399–1405.

Espenschied, K. S., Quinn, R. D., Beer, R. D., and Chiel, H. J. (1996). Biologically based distributed control and local reflexes improve rough terrain locomotion in a hexapod robot. *Robotics and Autonomous Systems*, 18(1-2):59 – 64.

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

Ganong, W. F. (2009). *Review of medical physiology.* McGraw-Hill Medical, 23th edition edition.

Gonzalez-Gomez, J., Zhang, H., and Boemo, E. (2007). Locomotion principles of 1d topology pitch and pitch-yaw-connecting modular robots. In *Bioinspiration and Robotics: Walking and Climbing Robots*, pages 403–428. Advanced Robotic Systems International and I-Tech Education and Publishing, Vienna, Austria.

Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., and Maisonnier, B. (2009). Mechatronic design of nao humanoid. In *Proceeding of 2009 IEEE International Conference on Robotics and Automation (ICRA)*, pages 769 –774, Kobe, Japan.

Granosik, G., Hansen, M. G., and Borenstein, J. (2005). The omnitread serpentine robot for industrial inspection and surveillance. *Industrial Robot-an International Journal*, 32:139–148.

Gray, J. (1946). The mechanism of locomotion in snakes. *Journal of Experimental Biology*, 23(2):101–120.

Grillner, S. (1985). Neurobiological bases of rhythmic motor acts in vertebrates. *Science*, 228(4696):143–149.

Grillner, S., Wallen, P., Brodin, L., and Lansner, A. (1991). Neuronal network generating locomotor behavior in lamprey: circuitry, transmitters, membrane properties, and simulation. *Annual Review of Neuroscience*, 14:169–199.

Hatton, R. L. and Choset, H. (2010). Generating gaits for snake robots: annealed chain fitting and keyframe wave extraction. *Auton. Robots*, 28(3):271–281.

Heliot, R. and Espiau, B. (2008). Multisensor input for cpg-based sensory–motor coordination. *Robotics, IEEE Transactions on*, 24(1):191–195.

Herrero-Carrón, F., Rodríguez, F. B., and Varona, P. (2011). Bio-inspired design strategies for central pattern generator control in modular robotics. *Bioinspiration & Biomimetics*, 6(1):1–16.

Hirai, K., Hirose, M., Haikawa, Y., and Takenaka, T. (1998). The development of honda humanoid robot. In *Proceeding of 1998 IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1321–1326.

Hirose, S. (1993). *Biologically inspired robots: snake-like locomotors and manipulators*. Oxford science publications. Oxford University Press.

Hirose, S. and Endo, G. (1997). Development of autonomous snake-like robot acm r-1. In *Proc. of the 1997 Annual Conf. Robotics & Mechatronics*, pages 309–310 (in Japanese).

Hirose, S. and Morishima, A. (1990). Design and control of a mobil robot with an articulated body. *Int. J. Rob. Res.*, 9(2):99–114.

Hirose, S., Morishima, A., Tukagosi, S., Tsumaki, T., and Monobe, H. (1991). Design of practical snake vehicle: articulated body mobile robot kr-ii. In *Proc. 5th Int. Conf. Advanced Robotics*, volume 1, pages 833–838.

Hirose, S. and Yamada, H. (2009). Snake-like robots. *Robotics Automation Magazine, IEEE*, 16(1):88–98.

Hooper, S. L. (2000). Central pattern generators. *Current biology*, 10(5).

Hopkins, J. K., Spranklin, B. W., and Gupta, S. K. (2009). A survey of snake-inspired robot designs. *Bioinspiration & Biomimetics*, 4(2):021001.

Huang, Q. and Nakamura, Y. (2005). Sensory reflex control for humanoid walking. *IEEE Transactions on Robotics*, 21(5):977–984.

Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21(4):642 – 653.

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

Ijspeert, A. J., Crespi, A., Ryczko, D., and Cabelguen, J.-M. (2007). From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315(5817):1416–1420.

Ijspeert, A. J., Hallam, J., and Willshaw, D. (1997). Artificial lampreys: comparing naturally and artificially evolved swimming controllers. In *Proceeding of 4th European Conference on Artificial Life*, pages 256–265.

Ijspeert, A. J., Hallam, J., and Willshaw, D. (1998). Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology. *Adaptive Behavior*, 7(2):151 – 172.

Inoue, K., Sumi, T., and Ma, S. (2007). Cpg-based control of a simulated snake-like robot adaptable to changing ground friction. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1957–1962.

Johnson, A., Wright, C., Tesch, M., Lipkin, K., and Choset, H. (2011). A novel architecture for modular snake robots. Technical report, Robotics Institute.

Kamata, Y., Ming, A., and Shimojo, M. (2007). Motion control of a manipulator with mechanical joint stops. In *Proceedings of 2007 IEEE International Conference on Robotics and Biomimetics(ROBIO)*, pages 1451 –1456.

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

Kamimura, A., Kurokawa, H., Yoshida, E., Tomita, K., Murata, S., and Kokaji, S. (2003). Automatic locomotion pattern generation for modular robots. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, pages 714–720.

Khunnithiwarawat, T. and Maneewarn, T. (2011). A study of active-wheel snake robot locomotion gaits. In *Proceedings of 2011 IEEE International Conference on the Robotics and Biomimetics (ROBIO)*, pages 2805–2809.

Kim, B., Lee, M. G., Lee, Y. P., Kim, Y., and Lee, G. (2006). An earthworm-like micro robot using shape memory alloy actuator. *Sensors and Actuators A: Physical*, 125(2):429 –437.

Kimura, H., Fukuoka, Y., and Cohen, A. H. (2007). Adaptive dynamic walking of a quadruped robot on natural ground based on biological concepts. *The International Journal of Robotics Research*, 26(5):475–490.

Kimura, H. and Hirose, S. (2002). Development of genbu : active wheel passive joint articulated mobile robot. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 823–828.

Kimura, H., Miyazaki, K., and Kobayashi, S. (1997). Reinforcement learning in pomdps with function approximation. In *Proceedings of the 14th International Conference on Machine Learning (ICML)*, pages 152–160.

Kimura, H., Yamamura, M., and Kobayashi, S. (1995). Reinforcement learning by stochastic hill climbing on discounted reward. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*, pages 295–303.

Klaassen, B. and Paap, K. L. (1999). Gmd-snake2: a snake-like robot driven by wheels and a method for motion control. In *Proceeding of 1999 IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3014–3019.

Kohl, N. and Stone, P. (2004). Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceeding of 2004 IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2619–2624.

Kolter, J., Rodgers, M., and Ng, A. (2008). A control architecture for quadruped locomotion over rough terrain. In *Proceeding of 2008 IEEE International Conference on Robotics and Automation (ICRA)*, pages 811–818.

Kurokawa, H., Kamimura, A., Yoshida, E., Tomita, K., and Kokaji, S. (2003). M-tran ii: metamorphosis from a four-legged walker to a caterpillar. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2454–2459.

Kurokawa, H., Tomita, K., Kamimura, A., Kokaji, S., Hasuo, T., and Murata, S. (2008). Distributed self-reconfiguration of m-tran iii modular robotic system. *Int. J. Rob. Res.*, 27(3-4):373–386.

Liljeback, P., Pettersen, K. Y., Stavdahl, O., and Gravdahl, J. T. (2011). Experimental investigation of obstacle-aided locomotion with a snake robot. *IEEE Transactions on Robotics*, 27(4):792–800.

Liljeback, P., Pettersen, K. Y., Stavdahl, O., and Gravdahl, J. T. (2012a). Snake robot locomotion in environments with obstacles. *IEEE/ASME Transactions on Mechatronics*, 17(6):1158 –1169.

Liljeback, P., Pettersen, K. Y., Stavdahl, O., and Gravdahl, J. T. (2012b). *Snake Robots: modelling, mechatronics, and control*. Advances in Industrial Control. Springer.

Liu, C., Chen, Q., and Wang, D. (2011). Cpg-inspired workspace trajectory generation and adaptive locomotion control for quadruped robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 41(3):867–880.

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

Ma, S. (1999). Analysis of snake movement forms for realization of snake-like robots. In *Proceeding of 1999 IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3007–3013.

Ma, S., Araya, H., and Li, L. (2001). Development of a creeping snake-robot. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 77 – 82.

Ma, S., Ohmameuda, Y., Inoue, K., and Li, B. (2003). Control of a 3-dimensional snake-like robot. In *Proceeding of 2003 IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 2067–2072.

MacKay-Lyons, M. (2002). Central pattern generation of locomotion: a review of the evidence. *Physical Therapy*, 82(1):69–83.

Manoonpong, P., Pasemann, F., and Wörgötter, F. (2008). Sensor-driven neural control for omnidirectional locomotion and versatile reactive behaviors of walking machines. *Robotics and Autonomous Systems*, 56(3):265–288.

Marder, E. and Bucher, D. (2001). Central pattern generators and the control of rhythmic movements. *Current Biology*, 11(23):986–996.

Masayuki, A., Takayama, T., and Hirose, S. (2004). Development of "souryu-iii": connected crawler vehicle for inspection inside narrow and winding spaces. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 52–57.

Matsubara, T., Morimoto, J., Nakanishi, J., Sato, M.-a., and Doya, K. (2005). Learning sensory feedback to cpg with policy gradient for biped locomotion. In *Proceeding of 2005 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4164–4169.

Matsuoka, K. (1985). Sustained oscillations generated by mutually inhibiting neurons with adaptation. *Biological Cybernetics*, 52:367–376.

Matsuoka, K. (1987). Mechanisms of frequency and pattern control in the neural rhythm generators. *Biological Cybernetics*, 56:345–353.

Matthews, G. G. (2001). *Neurobiology: Molecules, Cells and Systems*. Wiley-Blackwell, second edition edition.

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

Menciassi, A., Gorini, S., Pernorio, G., Weiting, L., Valvo, F., and Dario, P. (2004). Design, fabrication and performances of a biomimetic robotic earthworm. In *Proceedings of 2004 IEEE International Conference on the Robotics and Biomimetics (ROBIO)*, pages 274–278.

Meuleau, N., Peshkin, L., and Kim, K.-E. (2001). Exploration in gradient-based reinforcement learning. Technical report, MIT.

Mezoff, S., Papastathis, N., Takesian, A., and Trimmer, B. A. (2004). The biomechanical and neural control of hydrostatic limb movements in manduca sexta. *Journal of experimental biology*, 207(17):3043–3053.

Miller, G. (2002). Snake robots for search and rescue. *Neurotechnology for Biomimetic Robots*, pages 271–284.

Mondada, F., Guignard, A., Bonani, M., Bar, D., Lauria, M., and Floreano, D. (2003). Swarm-bot: from concept to implementation. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1626–1631.

Mondada, F., Pettinaro, G. C., Guignard, A., Kwee, I. W., Floreano, D., Deneubourg, J.-L., Nolfi, S., Gambardella, L. M., and Dorigo, M. (2004). Swarm-bot: a new distributed robotic concept. *Auton. Robots*, 17(2-3):193–221.

Mori, M. and Hirose, S. (2002). Three-dimensional serpentine motion and lateral rolling by active cord mechanism acm-r3. In *Proceedings of the 2002 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), volume 1, pages 829–834.

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

Ohno, H. and Hirose, S. (2001). Design of slim slime robot and its gait of locomotion. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 707–715.

Osuka, K. (2003). Development of four-crawler multilink mobile robot moira for searching debris. *J. Robotics Mechatron.*, 15:561–570.

Patel, L. N., Murray, A., and Hallam, J. (2006a). Super-lampreys and wave energy: Optimised control of artificially-evolved, simulated swimming lamprey. *Neurocomputing*, 70(7-9):1139 – 1154.

Patel, L. N., Murray, A. F., and Hallam, J. (2006b). Evolving multi-segment 'super-lamprey' cpg's for increased swimming control. In *Proceeding of 2006 European Symposium on Artificial Neural Networks*, pages 461–466, Bruges, Belgium.

Peters, J. and Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697.

Raibert, M., Blankespoor, K., Nelson, G., Playter, R., and et.al. (2008). BigDog, the rough-terrain quadruped robot. In *Proceedings of the 17th IFAC World Congress*, pages 10822–10825, Seoul, Korea.

Riedmiller, M., Peters, J., and Schaal, S. (2007). Evaluation of policy gradient methods and variants on the cart-pole benchmark. In *Proceedings of the 2007 IEEE international symposium on approximate dynamic programming and reinforcement learning*, pages 254–261.

Righetti, L. and Ijspeert, A. J. (2008). Pattern generators with sensory feedback for the control of quadruped locomotion. In *Proceeding of 2008 IEEE International Conference on Robotics and Automation (ICRA)*, pages 819–824.

Rome, E., Hertzberg, J., Kirchner, F., Licht, U., and Christaller, T. (1999). Towards autonomous sewer robots: the makro project. *Urban Water*, 1(1):57 – 70.

Rus, D. and Vona, M. (2000). A basis for self-reconfiguring robots using crystal modules. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2194 –2202.

Rus, D. and Vona, M. (2001). Crystalline robots: self-reconfiguration with compressible unit modules. *Autonomous Robots*, 10(1):107–124.

Ryu, J.-K., Chong, N., You, B., and Christensen, H. (2010). Locomotion of snake-like robots using adaptive neural oscillators. *Intelligent Service Robotics*, 3(1):1–10.

Saito, M., Fukaya, M., and Iwasaki, T. (2002). Modeling, analysis, and synthesis of serpentine locomotion with a multilink robotic snake. *IEEE Contr. Syst. Mag.*, 22:64 –81.

Sakurai, T., Okamoto, S., Konyo, M., and Tadokoro, S. (2010). Research of conditions of stimulus for inducing grasping force control reflex. In *IEEE/SICE International Symposium on System Integration*, pages 408 –413.

Sato, T., Kano, T., and Ishiguro, A. (2011). A snake-like robot driven by a decentralized control that enables both phasic and tonic control. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1881–1886.

Seo, K., Chung, S.-J., and Slotine, J.-J. E. (2010). Cpg-based control of a turtle-like underwater vehicle. *Autonomous Robots*, 28(3):247–269.

Seok, S., Onal, C. D., Cho, K.-J., Wood, R. J., Rus, D., and Kim, S. (2012). Meshworm: a peristaltic soft robot with antagonistic nickel titanium coil actuators. *IEEE/ASME Transactions on Mechatronics*, PP(99):1–13.

Shan, J. and Nagashima, F. (2002). Neural locomotion controller design and implementation for humanoid robot hoap-1. In *Proceedings of 20th Annual Conference of the Robotics Society of Japan*.

Smith, R. (2008). Open dynamics engine. http://www.ode.org/.

Spenneberg, D. and Kirchner, F. (2007). The bio-inspired scorpion robot: design, control & lessons learned. *Climbing and Walking Robots, towards New Application, I-Tech Education and Publishing, Vienna*, pages 197–218.

Stefanini, C., Orofino, S., Manfredi, L., Mintchev, S., Marrazza, S., Assaf, T., Capantini, L., Sinibaldi, E., Grillner, S., Wallen, P., and Dario, P. (2012). A novel autonomous, bioinspired swimming robot developed by neuroscientists and bioengineers. *Bioinspir Biomim*, 7(2):1–8.

Streich, H. and Adria, O. (2004). Software approach for the autonomous inspection robot makro. In *Proceeding of 2004 IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3411–3416.

Suh, J., Homans, S., and Yim, M. (2002). Telecubes: mechanical design of a module for self-reconfigurable robotics. In *Proceeding of 2002 IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 4095–4101.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge Univ Press.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12(22).

Takayama, T. and Hirose, S. (2000). Development of souryu-i connected crawler vehicle for inspection of narrow and winding space. In *Proceeding of the 26th Annual Conference of the IEEE Industrial Electronics Society (IECON)*, volume 1, pages 143 –148.

Takayama, T. and Hirose, S. (2003). Development of "souryu i and ii"–connected crawler vehicle for inspection of narrow and winding space. *Journal of Experimental Biology*, 15(1):61–69.

Tesch, M., Lipkin, K., Brown, I., Hatton, R. L., Peck, A., Rembisz, J., and Choset, H. (2009). Parameterized and scripted gaits for modular snake robots. *Advanced Robotics*, 23(9):1131–1158.

Togawa, K., Mori, M., and Hirose, S. (2000). Study on three-dimensional active cord mechanism: development of acm-r2. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2242–2247, Takamatsu, Japan.

Tomoyuki, T., Azuma, Y., and Shibata, T. (2009). Acquisition of energy-efficient bipedal walking using cpg-based reinforcement learning. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 827–832.

Transeth, A., Leine, R., Glocker, C., and Pettersen, K. (2008a). 3d snake robot motion: nonsmooth modeling, simulations, and experiments. *IEEE Transactions on Robotics*, 24(2):361–376.

Transeth, A. A., Leine, R. I., Glocker, C., Pettersen, K. Y., and Liljeback, P. (2008b). Snake robot obstacle-aided locomotion: modeling, simulations, and experiments. *IEEE Transactions on Robotics*, 24(1):88–104.

Vaidyanathan, R., Chen, C.-T., Jeong, C.-D., Williams, C., Endo, Y., Ritzmann, R. E., and Quinn, R. D. (2012). A reflexive vehicle control architecture based on a neural model of the cockroach escape response. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 226(5):699–718.

Wall, M. (1996). Galib: A c++ library of genetic algorithm components. *Mechanical Engineering Department, Massachusetts Institute of Technology*, 87.

Weaver, L. and Tao, N. (2001). The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the 17th conference on Uncertainty in artificial intelligence*, pages 538–545.

Weiner, W. J. and Shin, R. K. (2010). *Neurology for the non-neurologist*. Lippincott Williams & Wilkins.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256.

Williamson, M. M. (1998). Neural control of rhythmic arm movements. *Neural Networks*, 11:1379–1394.

Wright, C., Buchan, A., Brown, B., Geist, J., Schwerin, M., Rollinson, D., Tesch, M., and Choset, H. (2012). Design and architecture of the unified modular snake robot. In *Proceeding of 2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4347 –4354.

Wright, C., Johnson, A., Peck, A., Z.McCord, Naaktgeboren, A., Gianfortoni, P., Gonzalez-Rivero, M., Hatton, R., and Choset, H. (2007). Design of a modular snake robot. In *Proceedings of the 2007 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA.

Wu, X. and Ma, S. (2010). Adaptive creeping locomotion of a cpg-controlled snake-like robot to environment change. *Autonomous Robots*, 28(3):283–294.

Yamada, H., Chigisaki, S., Mori, M., Takita, K., Ogami, K., and Hirose, S. (2005). Development of amphibious snake-like robot acm-r5. In *Proceedings of 36th Int. Symposium on Robotics*, volume 1.

Yamada, H. and Hirose, S. (2006). Development of practical 3-dimensional active cord mechanism acm-r4. *J. of Robotics and Mechatronics*, 18(3):305–311.

Yim, M. (1994). *Locomotion with unit-modular reconfigurable robot*. PhD thesis, Stanford, CA, USA.

Yim, M., Duff, D., and Roufas, K. (2000). Polybot: a modular reconfigurable robot. In *Proceeding of 2000 IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 514–520.

Yim, M., Homans, S., and Roufas, K. (2001). Climbing with snake-like robots. In *Proceedings of the IFAC Workshop on Mobile Robot Technology*, Jejudo, Korea.

Yim, M., Roufas, K., Duff, D., Zhang, Y., Eldershaw, C., and Homans, S. (2003). Modular reconfigurable robots in space applications. *Auton. Robots*, 14(2-3):225–237.

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

Yu, J., Wang, M., Tan, M., and Zhang, J. (2011). Three-dimensional swimming. *IEEE Robot. Automat. Mag.*, 18(4):47–58.

Zhang, H., Gonzalez-Gomez, J., Me, Z., Cheng, S., and Zhang, J. (2008). Development of a low-cost flexible modular robot gz-i. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronlics*, pages 223 –228.

Zhang, H., Gonzalez-Gomez, J., and Zhang, J. (2009). A new application of modular robots on analysis of caterpillar-like locomotion. In *Proceedings of the 2009 IEEE International Conference on Mechatronics*, pages 1–6.

Zhang, H., Wang, W., Deng, Z., Zong, G., and Zhang, J. (2006). A novel reconfigurable robot for urban search and rescue. *International Journal of Advanced Robotic Systems*, 3(4):359–366.