A Biologically Inspired Four Legged Walking Robot

This thesis is presented for the degree of
Doctor of Philosophy in Engineering,
Murdoch University, Perth, Western Australia, Australia

2006

By Shiqi Peng

B.E. (Honours)

Supervisors:

Assoc. Prof. Graeme R. Cole
School of Electrical, Energy and Process Engineering, Murdoch University,
Murdoch WA 6150, Australia

Assoc. Prof. Chiou Peng Lam
School of Computer and Information Science, Edith Cowan University, Mt. Lawley
WA 6050, Australia
Acknowledgment

First of all, I would like to take this opportunity to thank my two supervisors: Assoc. Prof. Graeme R. Cole and Assoc. Prof. C. Peng Lam. It was with their enormous encouragement, guidance, support and care during my Ph.D. research that has enabled me to reach this point. Specifically, I would like to thank Graeme for his enormous help in assisting me to build the experimental robot, in helping me to understand the Subsumption Architecture and the Forth language. In addition, I would like to thank Graeme and Peng for helping me to improve my English. It is my pleasure and honour to say that I have been lucky to have two very good supervisors. Without their support, this thesis would not have eventuated.

I would also like to thank Murdoch University and the Australian Government for providing me with the opportunity to come to Australia to undertake my Ph.D. study and for providing me with the scholarship in the first three years of my research. In addition, I would like to thank the School of Engineering (now Electrical, Energy and Process Engineering) which provided me with all the necessary equipment and appropriate environment to carry out this research.

I would like to thank Kelly very much for her assistance in helping me to settle down and adjust to the new environment in Australia when I first arrived. Also, I would like to thank my friend Wei for all of his assistance in helping me to start my life in Australia. I would like to thank Jo who provided considerable help in improving my English. Thanks to Gary and John in helping me to build the robot. I would like to thank all other staff and friends in the School that helped me either in my daily life
or in my research.

I would like to take the opportunity to thank all my colleagues and managers of Woodside, for their supports and cares of me. It is a long list of names for me to write down here: Tu-uyen, David, Garvin, Susan, Tanya, Christine, John, Chris, Antonio, Robin, Barry, Simon, Gwen, Tom, Fiona, Liz, Mi, Swee, Helene, Ron, Bob …. For those I haven’t been able to list here, my thanks to you are the same. In addition, I would also like to thank Woodside itself. The extraordinary working experiences in Woodside significantly expand my visions and skills for the Ph.D. research, particularly in processing the data of experimental results.

I would like to take this opportunity to thank all my friends and relatives that have supported and helped me in the past and during the time of my Ph.D. research, too numerous to list here, thanks. Finally and most importantly, I would like to thank my parents and my wife, Meiqian, for their enormous love, support, encouragement and care to allow me to carry out this Ph.D. research, especially in those difficult times when I lost my way or was sidetracked. Were it not for them, I am sure that I could not have come this far in finishing this research. This thesis is especially for all of you.

Shiqi
Eight years ago far away in a small town in China, I had the idea of starting a Ph.D. in Australia. At the beginning of the research, I was a little lost, as I knew little about the field of robotics. It was with the help of my supervisors, Graeme and Peng, that I started on a journey of developing a greater understanding of robotics. After several months of reviewing and studying the literature (which I can not deny was a little boring), I found it necessary to build an experimental robot. This would provide a research platform to investigate issues associated with both the static and dynamic balance of a walking legged robot, with a strong emphasis on a biological perspective. The design and construction of the robot provided valuable experience and insight, even though at times, it was very frustrating. The robot “easily” fell down when walking due to a range of factors: narrow base, high centre of gravity, multiple degrees of freedom of limb movements, loose joints, small feet, etc. However this was a deliberate part of the design. In attempting to have a strong biological perspective the robot was modeled loosely on the structure of a dog and as such easily fell over. This was in order to facilitate and more thoroughly investigate the balance issue. In the first stage of the research, Reinforcement Learning alone was used by the robot to learn to keep its balance. The results were encouraging and led to a published paper “Using Reinforcement Learning to Achieve Balance for a Four Legged Walking Robot”, presented at the ISTIA 2000 international conference held at Canberra, Australia (2000). However, this work only allowed the robot to keep its balance but was not successful in making the robot walk. I am now in a position to be able to explain why it was difficult to solely use Reinforcement
Learning for getting the robot to walk. The learning space was simply too big for the limited capability of the onboard microcontroller. Given this lack of success in achieving “walking”, a parallel Subsumption Architecture approach was developed. It was found that each major element of the robot’s walking could be designed as a specific behaviour with the interaction of all the behaviours enabling the robot to walk - an “Emergent Walking Behaviour”. A four-phase walking strategy was developed. It was based upon other walking related research and observing videos of how horses and dogs walk. To implement the four-phase walking strategy, it was necessary to develop a central control unit to coordinate the movement of individual legs. This so-called Central Pattern Producer (CPP) coordinated the movements of all four legs of the robot by generating rhythmical walking phase signal sets. Many experiments were conducted to “tune” the entire system and obtain an operationally walking robot including the ability to balance. This work led to two published papers: "A Biologically Inspired Four Legged Robot that Exhibits some Natural Walking Behaviours", presented at the IAT 2001 International Conference (Maebashi City in Japan, 2001), and "A Biologically Inspired Four Legged Walking Robot", presented at the IEEE ICRA 2003 international conference (Taiwan, 2003).

In the process of implementing the proposed four-phase walking strategy, it was found that it was not simple to implement a Subsumption Architecture (SA) from scratch, especially given that there were several parallel SAs in the system. With regards to the robot, the control of each leg was implemented as an independent SA. There were four SAs running in parallel in the system. This precipitated the idea of providing a recipe or implementation framework for implementing such parallel SA
systems. This work led to two other published papers: "An implementation methodology of Subsumption Architecture for Robotics", presented at the ISA 2000 international conference (Wollongong of Australia, 2000), and "A generic framework for implementing Subsumption Architecture", presented at RA2000 international conference (Honolulu, 2000).

In the early stages of the experiments, an empirical value for the CPP was used to generate rhythmical walking phase signals. It would be better for the robot to learn this value through real-time interaction with the environment. This resulted in the idea of again using machine learning, specifically reinforcement learning. Experiments associated with the learning algorithm demonstrated that the robot can successfully learn an “optimal” value associated with a specific terrain.

Although I have reached an end point of my research, there are still many interesting experiments to be carried out to investigate further aspects of the robot's walking, e.g. trotting, pacing, and more complicated learning tasks. These future investigations are left to my successors to carry out. For them, I would like to say “good luck”.

This thesis collects together all of the individual components of my research work contained in my published papers. However, it is presented here as a single piece of consolidated work and thus it is hoped that it reads in this way and is not simply an assembly of papers.
Table of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>Assisting Behaviour</td>
</tr>
<tr>
<td>AC</td>
<td>Action Component</td>
</tr>
<tr>
<td>BSM</td>
<td>Behaviour Suppression Mask</td>
</tr>
<tr>
<td>CPG</td>
<td>Central Pattern Generator</td>
</tr>
<tr>
<td>CPP</td>
<td>Central Pattern Producer</td>
</tr>
<tr>
<td>DOF</td>
<td>Degrees of Freedom</td>
</tr>
<tr>
<td>DSM</td>
<td>Dynamic Stability Margin</td>
</tr>
<tr>
<td>EC</td>
<td>Executor Component</td>
</tr>
<tr>
<td>IW</td>
<td>Ideal Walking</td>
</tr>
<tr>
<td>IWB</td>
<td>Ideal Walking Behaviour</td>
</tr>
<tr>
<td>LB</td>
<td>Left Back</td>
</tr>
<tr>
<td>LF</td>
<td>Left Front</td>
</tr>
<tr>
<td>NIW</td>
<td>Non-ideal Walking</td>
</tr>
<tr>
<td>NIWB</td>
<td>Non-ideal Walking Behaviour</td>
</tr>
<tr>
<td>OO</td>
<td>Object Oriented</td>
</tr>
<tr>
<td>OWCI</td>
<td>Optimal Walking Cycle Interval</td>
</tr>
<tr>
<td>PSS</td>
<td>Phase Signal Set</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulated</td>
</tr>
<tr>
<td>RB</td>
<td>Right Back</td>
</tr>
<tr>
<td>RF</td>
<td>Right Front</td>
</tr>
<tr>
<td>RL</td>
<td>Reinforcement Learning</td>
</tr>
<tr>
<td>SA</td>
<td>Subsumption Architecture</td>
</tr>
<tr>
<td>SM</td>
<td>Suppression Mask</td>
</tr>
<tr>
<td>SSM</td>
<td>Static Stability Margin</td>
</tr>
<tr>
<td>TC</td>
<td>Trigger Component</td>
</tr>
<tr>
<td>WCI</td>
<td>Walking Cycle Interval</td>
</tr>
<tr>
<td>ZMP</td>
<td>Zero Moment Point</td>
</tr>
</tbody>
</table>
Abstract

This Ph.D. thesis presents the design and implementation of a biologically inspired four-phase walking strategy using behaviours for a four legged walking robot. In particular, the walking strategy addresses the balance issue, including both static and dynamic balance that were triggered non-deterministically based on the robot’s real-time interaction with the environment. Four parallel Subsumption Architectures (SA) and a simple Central Pattern Producer (CPP) are employed in the physical implementation of the walking strategy. An implementation framework for such a parallel Subsumption Architecture is also proposed to facilitate the reusability of the system. A Reinforcement Learning (RL) method was integrated into the CPP to allow the robot to learn the optimal walking cycle interval (OWCI), appropriate for the robot walking on various terrain conditions. Experimental results demonstrate that the robot employs the proposed walking strategy and can successfully carry out its walking behaviours under various experimental terrain conditions, such as flat ground, incline, decline and uneven ground. Interactions of all the behaviours of the robot enable it to exhibit a combination of both preset and emergent walking behaviours.
List of Published Papers

Below is a list of papers that have been published. These papers essentially represented the results of various stages of the research work carried out during the Ph.D. project and are described within this thesis as a consolidated piece of work.


Table of Contents

CHAPTER 1: INTRODUCTION ........................................................................................................ 12
1.1. About Robotics and Its Architectures .............................................................................. 12
1.2. The Problem in Context ................................................................................................. 13
1.3. The approach and contribution of this thesis ................................................................. 18
1.4. Structure of the thesis ................................................................................................... 21
1.5. Chapter Summary ......................................................................................................... 23

CHAPTER 2: LITERATURE REVIEW AND EXISTING WORK ................................................. 24
2.1. Review of existing work on legged robot locomotion ...................................................... 24
  2.1.1. The Issue of Balance ............................................................................................. 24
  2.1.2. Brooks and his six legged walking robots ............................................................. 27
  2.1.3. Raibert and his running robot ................................................................................. 31
  2.1.4. Hirose and his walking robots – the TITAN series ................................................ 34
  2.1.5. Kimura and his walking robots .............................................................................. 39
  2.1.6. Rodney, the walking robot and the evolutionary approach (Genetic Programming) .. 43
  2.1.7. SCAMPER, the running robot from the Furusho Laboratory .................................. 45
  2.1.8. BISAM and the mammal-like locomotion robots ................................................... 47
  2.1.9. Bowling and Harmeyer’s legged robots for agile locomotion .................................. 49
  2.1.10. The latest works on bipedal robots .................................................................... 51
2.2. Review of techniques employed in this thesis ............................................................... 54
  2.2.1. The biological inspired approach and the Subsumption Architecture ................. 55
  2.2.2. Object Oriented Programming .............................................................................. 56
  2.2.3. Reinforcement Learning ...................................................................................... 58
2.3. Chapter Summary ......................................................................................................... 65

CHAPTER 3: THE EXPERIMENTAL ROBOT ...................................................................... 66
3.1. Introduction ................................................................................................................... 66
3.2. Mechanical Design ....................................................................................................... 67
3.3. Sensors ............................................................................................................................ 71
3.4. Onboard Microcontroller .............................................................................................. 72
3.5. Descriptions of motions of the robot associated with its designed behaviours ............ 73
  3.5.1. Introduction ....................................................................................................... 73
  3.5.2. Descriptions of behaviour motions designed for the proposed walking strategy .... 76
3.6. Chapter Summary ......................................................................................................... 80

CHAPTER 4: THE DESIGN OF THE FOUR-PHASE WALKING STRATEGY USING BEHAVIOURS .............................................................. 81
4.1. Introduction ................................................................................................................... 81
4.2. The four phase walking strategy using behaviours ...................................................... 82
4.3. The Central Pattern Producer (CPP) and walking ...................................................... 88
4.4. Setting of walking cycle intervals and integration of reinforcement learning ............... 91
  4.4.1. About the walking cycle interval ........................................................................... 91
  4.4.2. Design of the learning algorithm ........................................................................ 93
4.5. Chapter Summary ......................................................................................................... 95

CHAPTER 5: IMPLEMENTATION ....................................................................................... 96
5.1. Introduction ................................................................................................................... 96
5.2. Implementation of the four phase walking strategy ...................................................... 99
  5.2.1. Implementation framework ............................................................................... 99
  5.2.2. Applying the Implementation Framework to the Robot to Implement the Proposed Four-phase Walking Strategy ......................................................... 107
5.3. Implementation of the CPP and learning ................................................................... 114
CHAPTER 6: EXPERIMENT RESULTS AND DISCUSSION .................................................... 118

6.1. Introduction ............................................................................................................. 118
6.2. Experimental results and discussion for the case of an empirical value for the OWCI .......... 121
   6.2.1. Flat ground ...................................................................................................... 121
   6.2.2. Experiments on Other terrains: Incline, Decline and Uneven Ground ................ 128
6.3. Experimental results and discussion of the robot’s learning ........................................ 134
   6.3.1. Learning results for the robot walking on flat ground ........................................ 134
   6.3.2. Learning results for the robot walking on other terrains .................................... 138
6.5. Chapter Summary .................................................................................................. 143

CHAPTER 7: CONCLUSIONS AND FUTURE WORK .................................................... 144

REFERENCE: ............................................................................................................... 147
Chapter 1: Introduction

1.1. About Robotics and Its Architectures

In the robotics community, there are two basic robotic architectures: analytical approaches (e.g. Donner, 1987) and biologically inspired approaches (e.g. Brooks, 1986b). The merging of these two types of architectures generates a third group of approaches - hybrid approaches that integrate the characteristics of both analytical and biologically inspired architectures (e.g. Nicolescu and Mataric, 2002). The analytical approaches generally require a mathematical model of the system and are computationally intensive. On the other hand, the biological approaches do not involve a mathematical model of the system and are relatively less computationally intensive. Mataric (2002) stated that this type of approach employs general and bottom-up philosophies that support a certain degree of freedom from interpretation and computations. It can also tolerate a certain extent of false sensor information. Bekey (2000) stated that robotic control mechanisms could get more inspiration from the biological world as it exhibits greater reliability and efficiency despite the complexity of the world itself.

Robots can be categorised into two groups based on their method of locomotion: wheeled robots and legged robots. Legged robots have received increasing interest as they have some advantages that are less easily achieved by wheeled robots, such as
navigating in an uneven terrain. Legged robots can further be sub-classified into three major groups:

- Biped robots that have two legs (e.g. the bipedal walking robot of Collins and Ruina, 2005).
- Quadruped robots that have four legs (e.g. BISAM by Ilg and Berns, 1998).
- Insectoid robots that have more than four legs (e.g. Boadicea by Binnard, 1998).

1.2. The Problem in Context

In spite of research achievements in legged robots, there is still a tremendous amount of work required before legged robots can be used from a practical perspective (Horvath, 2000). The main reason is due to the complex dynamics of legged robots during locomotion. Both analytical (Raibert, 1989) and biologically inspired (Ilg, 1998) methods have been employed on legged robots to address this problem. In the context used in this thesis, there are six key issues associated with a legged robot’s locomotion:

1. To maintain its balance during locomotion;

2. Phases and action sequencing of the legs;

3. Coordination of the movements of the legs;

4. Ability to interact with different environmental conditions;

5. Development and implementation of a system composed of various behaviours;
6. Ability to carry out certain types of machine learning.

The first issue is associated with the topic of balance for a legged robot. Raibert (1985; 1987; 1989) had investigated this area in many of his papers. Analytical approaches were employed in his investigations, in which a leg is modeled as an inverted pendulum system (Raibert, 1989). Raibert stated that there are two types of balance strategies employed during the locomotion of legged robots: dynamic balance and static balance. With static balance, the centre of gravity of the robot is kept within its supporting area. For a four legged robot, this area is a three or four sided polygon where the vertices are the ground contact points of the legs. The robot can statically keep its posture and not fall down regardless of whether the robot is in motion or not. With dynamic balance, the robot's centre of gravity at times, momentarily falls outside its supporting area while the robot is in motion or there is no supporting area (e.g. two legs or less are in contact with the ground). The robot must then use its movements, which generates momentum, to compensate for its temporary instability. In the real world, animals employ both balance strategies during their locomotion. The faster an animal moves, the more often the dynamic strategy is employed. These two types of balance strategies have been implemented in various implementations, although in some cases it has not been explicitly stated. For example, static balance is employed in the work of Brooks' six legged robots (Brooks, 1989) using a biologically inspired approach and dynamic balance is employed in the work of Sardain, Rostami and Bessonnet (1999) using an analytical approach. Meanwhile, Fukuoka, Kimura and Cohen (2003) have applied a biological approach to their robot Tekken-I to achieve dynamic walking. Hirose et al. (1989)
had also investigated the problem of both dynamic and static walking in his robots (e.g. TITAN III & TITAN IV) using a central planning sway compensation schema. In his implementation, the robot can change its walking from static status to dynamic status as the locomotion speed increased. However, the transfer between static and dynamic balance in this group of robots is largely dependent on the robot’s locomotion speed, which can thus be viewed as a pre-determined approach. In this thesis, the balance issue is implemented using behaviours within four parallel SAs. The switch between static and dynamic balance is non-deterministic. The activation of a balance strategy is dependent on the real-time situation of the robot and its interaction with the environment.

The second issue is about the phasing and sequencing of a legged robot’s locomotion. This aspect of the project is indeed inspired from the real biological world. This thesis provides a brief discussion on the essential “phase and sequencing” characteristics associated with how animals carry out their locomotion. Biological studies on animal locomotion (e.g. Grillner, 1981) revealed that a natural rhythmic cycle associated with four legged animal locomotion is composed of several different phases (status), interpreted as *duty factor* in many biophysics documents (Alexander, 1984; Alexander, 1992; Grillner, 1981). Different gaits are employed by different animals and different gaits also involve different numbers of phases (Hoffman, 1982). For example, a walking gait has four-phases while trotting, pacing and bounding have two phases (Alexander, 1984). The reason why there are different gaits in animal locomotion is that a specific gait is more energy efficient at a certain speed (Alexander, 1992). For instance, the walking gait is suitable for low-
speed locomotion while pacing (or trotting) is suitable for high-speed locomotion (Howell, 1965). In the investigation carried out by Alexander (1984), the walking gait had four different phases. Each of the legs undergoes these four-phases during walking. However, at any one point in time, each of the legs will be in a different phase with respect to each other. With regards to the implementation of the phase issue in legged robots’ locomotion, much of the existing work either uses an analytical methodology (Collins and Stewart, 1994) or CPG based approach (Bazhenov et al., 1996; Golubitsky, 1998; Lewis, 2000; Reeve and Hallam, 1995; Kimura and Fukuoka, 2000). In this thesis, phases of the robot’s walking are implemented using specific behaviours. Each phase in a legged robot’s walking is implemented as a behaviour in a SA.

With regards to the coordination issue (third issue), a legged robot needs coordination of its different legs to generate sufficient force to move forward. Much of the existing work has addressed this issue by implementing a Central Pattern Generator (CPG) (Bazhenov et al., 1996; Golubitsky, 1998; Lewis, 2000; Reeve and Hallam, 1995), which emphasises or incorporates a central planning/control schema. Kimura and Fukuoka (2000) implemented this strategy using the model of a Neuron Oscillator (Matsuoka, 1985). In this thesis, the coordination of legged movements of a legged robot is implemented by a simple Central Pattern Producer (CPP), which functions like a CPG but works differently. The CPP only generates a periodic phase signal set (PSS) for all four legs which results in the activation and interactions of specific behaviours to physically implement the coordination of legged movements.

In terms of interaction with different environmental conditions (fourth issue), a
legged robot must be able to react in real-time to changes in the environment, both internally and externally, and to be able to recover from undesirable circumstances when they occur. Much of the existing works (Connel, 1990; Francis, 1993; Peasegood et al., 1998; Robinson and Jenkin, 1995; Scott, 1997) addresses this issue using the Subsumption Architecture (SA) proposed by Brooks (1986b). In this thesis, the “reaction” issue was also implemented using SA via specific behaviours, for example the Balance behaviour (see explanation in chapter 4). The activation of certain sensor conditions (either physical or virtual) will activate its associated behaviours to take over control of the robot’s walking.

For the behaviour issue (fifth issue), Brooks (1989) has implemented a six legged walking robot using specific behaviours. The walking robot employed in his implementation had six legs and only dealt with static balance. After Brooks, other research projects (e.g. Horvath et al., 1997; Peasegood et al., 1998; Robertson, 1996; Scott, 1997) took a similar approach to implement robot walking by using behaviours. In this thesis, behaviours are implemented to deal with both static and dynamic balance. It is also used to define and to implement various phases of a legged robot’s walking.

Learning, the last and sixth issue, has been implemented in legged robot locomotion in a variety of situations (e.g. Berns, 1994; Huber and Grupen, 1998; Ilg et al., 1997; Nicolescu and Mataric, 2001). The learning methodology in these implementations addressed various aspects of a legged robot’s walking, for example such as walking gait, locomotion trajectory and locomotion planning. In this thesis, it was implemented in a way that uses reinforcement learning to allow the robot to learn the
optimal walking cycle interval (OWCI) for the CPP to output the walking phase pattern for the “legged” behaviours.

1.3. The approach and contribution of this thesis

This thesis is an attempt to address each of the six key issues (introduced and briefly discussed in the previous section) associated with a four legged walking robot using some biologically inspired approaches. More specifically, this thesis attempts to investigate an area that involves the employment of both static and dynamic balancing, which can be triggered non-deterministically based on the robot’s real-time interaction with the environment. Implementation was achieved via a four phase walking strategy through the use of specific behaviours within four parallel SAs. An RL algorithm was also integrated into the system to enable the robot to learn the value of the OWCI via real-time interaction with the environment during locomotion.

The robot used in this research was an experimental platform having similar complexities and characteristics to some biological counterparts (e.g. a high centre of gravity, narrow feet, limbs with multiple degrees of freedom and a narrow supporting base). A four-phase walking strategy was designed and implemented in the robot via a simple Central Pattern Producer (CPP) which integrated an RL algorithm and four parallel SAs (one for each leg), using a reusable implementation framework. Experimental results were recorded and investigated via various analysis methods (e.g. Behaviour plots and data correlation).
The major contributions of this thesis are listed below:

1. **Usage of behaviours to implement the walking phases:**

   Although behaviours have been used in legged robots, this thesis did not find any existing work that employs behaviours to implement walking phases. In the implementation of Brooks (1989), behaviours were employed to deal with limb movements only. They were not used to implement various phases in legged robot's locomotion. In addition, most of the existing works that have implemented phases have used either analytical methodologies (Collins and Stewart, 1994) or CPG based methodologies (Bazhenov et al., 1996; Golubitsky, 1998; Lewis, 2000; Reeve and Hallam, 1995; Kimura and Fukuoka, 2000).

2. **Implementation through the usage of four parallel SAs, to deal with non-deterministic dynamic and static balance:**

   The balance issue is addressed in this thesis by employing four parallel SAs to implement a system that allows the switch between static and dynamic balance in a non-deterministic way. This is in contrast to the work of Brooks (1989), where only static balance has been addressed using a biologically inspired approach, and the works of Raibert (1985; 1987; 1989), where only dynamic balance has been addressed using an analytical approach. The approach of this thesis is also different to that of Hirose et al. (1989), where the switch between dynamic and static balance is pre-determined.

3. **An experimental robot which is able to exhibit a combination of preset (e.g.
phase signal sequences output by the CPP) and emergent (i.e. the real-time interaction of the four parallel SAs with the environment) behaviours:

A contribution of this thesis is to use a combination of preset and emergent behaviours resulting from interactions of four SAs to produce a non-deterministic transition between static and dynamic balance as the robot interacts with the environment. Most existing work only dealt with either static (e.g. Brooks (1989)) or dynamic (e.g. Raibert (1989)), or the transition between static and dynamic balance were pre-deterministic (Hirose et al., 1989). This thesis also uses RL to learn the OWCI which is used by the CPP to output phase signal sets to the four legs of the robot.

4. Creation of a generic framework for implementation of parallel SAs. The generic framework employs an Object Oriented (OO) methodology to implement SA to facilitate the reuse of components and behaviours:

This is another contribution of the thesis. It is unique in that no similar existing work was found at the time the thesis was completed. It provides a generic framework for the implementation of parallel SAs using some principles of Object Orientation (OO), which allows other researchers and system developers to concentrate on the design of behaviours and suppression rules of the system, rather than being concerned with the details of implementation.

5. Successful demonstration of an entire approach which enables a “relatively unstable” robot (i.e. one with a narrow base, high centre of gravity, multiple
degrees of limb movements, loose joints, and small feet) to walk:

It is a deliberate attempt in this thesis to base the design and implementation of the walking strategy on a legged robot with a strong biological perspective, which in turn was modeled loosely on the structure of a dog and as such, will fall over and/or collapse easily without any appropriate control. A major difference between the experimental robot and many other existing robots is that it has very small feet with a narrow supporting width. The ground contact area varied approximately between 0.5 cm\(^2\) to 2 cm\(^2\), depending on the real-time situation of the robot's interaction with the ground, and at the same time the robot weighed about 16 kg. This thesis did not find other existing robots that had such small feet/ground contact area and a narrow supporting width, at the time of completion of this thesis. The approach was successfully evaluated using four different terrains, including flat ground, incline, decline and uneven ground.

In summary, the main contribution of this thesis was to design and implement a four phase walking strategy using specific behaviours that emphasise the balance issue in locomotion. This design was developed through a consideration of a biological perspective and implemented as specific behaviours via four parallel SAs and a CPP, in a generic reusable framework.

1.4. Structure of the thesis

This thesis is composed of 7 chapters, in which a four-phase walking strategy implemented using specific behaviours is proposed and implemented within an
This first chapter provided a general introduction to robotics and its architectures. It also discussed some key issues in locomotion of legged robots and some of the existing work that has been carried out to address some of these issues. Based on the discussion, the idea and overall approach of the thesis was proposed and discussed. Chapter 2 is a literature review, providing a discussion of some of the major existing work which is of particular relevance to this thesis. This includes the discussion of work by Brooks, Hirose, Kimura, Ilg, and Raibert, etc. It also provides a brief survey on all background information and techniques that were used in the thesis. Chapter 3 is concerned with the experimental platform. It presents a discussion of the important aspects of the design and construction of the four legged walking robot, which was employed and used as the experimental platform for all the experimental work discussed in this thesis. The chapter also discusses some of the dynamic and kinematic aspects associated with some of the proposed behaviours of the robot. Chapter 4 is concerned with the concept and design of the robot’s walking strategy. In particular, it addresses the design of the four-phase walking strategy, the CPP and the learning algorithm. The four-phase walking strategy was interpreted and implemented as specific behaviours via four parallel SAs and a CPP. This chapter also discusses the relationship between CPP and walking. It illustrates how the CPP affects the robot's behaviours and how the robot walks. Design of the learning algorithm that employs the one step Sarsa learning (Sutton and Barto, 1998) is also presented in this chapter. Following in Chapter 5, implementation details for the four-phase walking strategy, the CPP and the learning are presented and discussed. Specifically, it provides a general implementation framework that can be re-used in other applications. Chapter 6 is
concerned with the experimental results and a discussion involving the robot successfully carrying out real-time walking and learning in various terrains. Behaviour plots, data correlation and learning curves were employed to analyse the experiment results. The conclusion and future work are discussed in Chapter 7.

A point to note here is that when the term “biological” is used in this thesis, it only applies to the following components:

1. CPP which functions like a CPG – only from the perspective of generating a periodic phase signal set;
2. Behaviours and SA;
3. Walking phases and associated duty factor;
5. A physical robot which has “mechanically architectural” features and proportions which are similar to a mid-sized dog.

1.5. Chapter Summary

This chapter provided a general introduction to robotics and its architectures, specifically, legged robot walking from a biological perceptive. It discussed six key issues regarding legged robot walking and a discussion of some existing works that have been carried out to investigate these issues. This chapter also describes the major contributions and structure of this thesis.
Chapter 2: Literature Review and Existing Work

2.1. Review of existing work on legged robot locomotion

This section reviews existing work on legged robots and briefly compares that work to the approach proposed in this thesis. Given the enormous range and quantity of research concerning legged robots, it is impossible to give an all encompassing review. Instead, the intention here is to try to cover the major work which has made important contributions to the “art” of legged robot locomotion. In particular, a discussion of previous work which allows the approach presented in this thesis, to be placed in context.

As mentioned in the previous chapter, one of the fundamental issues in legged robot locomotion is balance, which could be classified into two categories: static balance and dynamic balance. This thesis gives a detailed discussion on the balance issue in the first section (2.1.1.), followed by 9 sections reviewing some of the major works and legged robots that have made important contributions to the community of legged robots.

2.1.1. The Issue of Balance

Like human beings and animals, a legged robot needs to be able to balance itself before it can walk. There are two approaches to achieve balance: statically or
dynamically. Legged robots employ either or both approaches in achieving locomotion.

The first approach is normally referred to as static balance, also known as static stability (McGhee and Frank, 1968). Static balance is achieved by keeping the ground projected point of the centre of mass of a robot within the supporting polygon formed by the legs of the robot that are making contact with the ground. Therefore, the minimum number of legs required for a legged robot to achieve static balance in walking is four as the robot is required to keep at least three legs on the ground at any time. Ridderström (2003) used the term of \( P_{CM} \) to denote the ground projecting point of the centre of the mass of a legged robot, and the term \( A_{sup} \) to denote the supporting area formed by the legs of a legged robot that are on the ground. Thus, the requirement for static balance could be represented as \( P_{CM} \in A_{sup} \) (Ridderström, 2003), whereby \( \in \) stands for “within”. McGhee and Frank (1968) also defined an important factor in static balance, the Static Stability Margin (SSM), which is defined as the shortest distance from \( P_{CM} \) to the boundary of \( A_{sup} \). In order for a legged robot to maintain static balance in locomotion, \( SSM \) must be greater than zero for all the times the robot is in motion. Ridderström (2003) argued that the locomotion of a legged robot could be assumed as static balance if at any time of the locomotion, \( SSM \) of the robot remains positive, as illustrated by equation 2.1.

\[
P_{CM}(t) \in A_{sup}(t) \quad \forall t \quad \text{equation 2.1 (Ridderström, 2003)}
\]

For legged robots that have four or more legs, static balance is commonly employed
to achieve balance in locomotion. However, because of the requirement of $P_{CM} \in A_{SUP}$, static balance has a limitation on the locomotion speed that the legged robot could achieve. Static balance is normally used for low-speed locomotion. For high-speed locomotion, dynamic balance, which will be discussed next, is a more efficient and practical approach to achieve balance.

Dynamic balance is also commonly known as dynamic stability or active balance (Raibert, 1986). A legged robot that employs dynamic balance in motion does not require to have its $P_{CM}$ kept within its $A_{SUP}$ at all the locomotion times, which could be illustrated as $P_{CM}(t) \notin A_{SUP}(t) \forall t$ (Ridderström, 2003), where $\notin$ stands for “not-within” and $\forall t$ stands for “at any given time”. Dynamic balance emphasizes the effects of the dynamics of motions. Hirose and Yoneda (1993) illustrated dynamic walking of a legged robot using dynamic balance as: a legged robot that employs dynamic balance in walking will start to fall and will not be able to carry out the planned walking when its locomotion speed is reduced to such a level whereby the effects of the dynamics of motions are lost. In other terms, a legged robot that employs dynamic balance will result in changing its locomotion gaits when leg motion of the robot has been changed to certain level. This is in contrast to static balance, where a legged robot that employs static balance will result in changing its locomotion speed but not its locomotion gaits when speed of the leg movement of the robot has been changed to a certain level (i.e. the legged robot could still keep the same locomotion gait regardless of the speed of the motion of its legs, providing leg motion pattern is not changed).
The concept of *Static Stability Margin* (*SSM*) of static balance could be extended to *Dynamic Stability Margin* (Lin and Song, 1993), *DSM*, by replacing $P_{CM}$ with $P_{CP}$, whereby $P_{CP}$ stands for the projecting point of the centre of pressure of a robot. Lin and Song (1993) defined $P_{CP}$ as the ground projecting point of the centre of mass of a legged robot and all other forces acting on the robot. Therefore *DSM* could be defined as the shortest distance from the $P_{CP}$ to the boundary of the supporting area formed by all legs of the robot that are on the ground (Ridderström, 2003). In order for a legged robot to maintain dynamic balance in motion, it is required that the robot keep *DSM* positive at any time during its locomotion, which could be illustrated by equation 2.2, a modification of equation 2.1 (Ridderström, 2003).

$$P_{CP} (t) \in A_{SUP} (t) \forall t \quad \text{equation 2.2}$$

An important term commonly referenced in dynamic balance is the Zero Moment Point (ZMP), which is similar to $P_{CM}$ as defined above. It was introduced by Vukobratovic and Stepanenko (1972, 1973). ZMP is defined as a ground projecting point at which the net force and net moment that are acting on a robot are zero (Yagi and Lumelsky, 2000).

### 2.1.2 Brooks and his six legged walking robots

Brooks (1986) of MIT is one of the pioneers in the area of investigation and implementation of robotic architectures using a biologically inspired methodology. He introduced the idea of the Subsumption Architecture (SA) in the mid 1980s. SA is a biologically inspired approach that emphasizes the principle of “Sensing –
Reacting”. It is composed of a set of pre-defined simple behaviours, for example move forward, stop, move backward, turn left, etc. Each behaviour is activated by certain physical or virtual sensor information. For example, the behaviour stop could be activated if an infrared sensor of a robot senses an obstacle in front of it. Even though each behaviour might be simple, the interaction of these simple behaviours could generate quite complex behaviours, known as emergent behaviours. For instance, the interaction of the above mentioned simple behaviours for a robot might generate some complicated emergent behaviours such as flocking behaviours.

Since the inception of SA, Brooks has applied it to many of his robots, including both legged and wheel robots. Of those, Genghis, Attila and Hannibal were the three of his legged robots that he had successfully implemented SA into and enabled them to walk. After Brooks, many other researchers have effectively applied and expanded his idea of SA in real applications, for example the Behaviour-Based architecture proposed by Arkin(1998) and Mataric(1992). SA (and those behavior-based architectures derived from SA) is now one of the popular robotic architectures used in the robotics community.

2.1.2.1 Genghis, the six legged walking robot

As mentioned previously, Genghis was a legged robot in which Brooks (1989) implemented the idea of SA to enable the robot to walk. Genghis is a six legged robot, built to walk on uneven terrain. Each leg of Genghis was manipulated by two motors. The first one was used to move a leg in a forward or backward motion, while the second one was used to move a leg in a up or down motion. This resulted
in each leg having two degrees of freedom (DOF), referred to as "Forward - Backward" and "Up -down". A picture of Genghis is shown in figure 2.1. Genghis has a wide base and low centre of gravity. This is to be contrasted with the experimental robot (Figure 3.1) employed in this thesis which had four legs, a narrow base, high centre of gravity and multi-jointed legs, each with 3 degrees of freedom (i.e. "ankle", "knee" and "hip" joints).

Only static balance was implemented in Genghis, which is to be contrasted with the experimental robot that deals with both static and dynamic balance. Behaviours of Genghis were composed of: Stand up, Simple walk, Force balancing, Leg lifting, Whiskers, Pitch stabilization, Prowling, and Steered prowling. Behaviour Stand up has the lowest priority, which means that this behaviour will be activated if no other behaviour is triggered. Behaviour Simple walk enables the robot to carry out a “tripod” type walking, i.e. front and back legs of one side and the middle leg of the other side being on the ground to support the robot while the rest of other legs are off ground and moving forward. Behaviour Force balancing and Pitch stabilization will be activated when the robot needs to move leg(s) up or down to adjust the balance of the robot. Behaviour Leg lifting will be activated when the robot needs to move its leg over an obstruction. Behaviours Whiskers, Prowling and Steered prowling were used to deal with obstacles and path following. Overall, interactions of these behaviours generated emergent behaviours of an insect-like motion.
2.1.2.2. Attila (or Hannibal), A Six Legged Autonomous Walking Robot

Attila and Hannibal (Angle and Brooks, 1990) are similar in physical structure, differing only in colour (Hannibal was red in colour and Attila was gold in colour). A picture of Attila is shown in Figure 2.2. It is also a six legged walking robot and is about 41 cm square and 20 cm high when standing up. It was designed and built in the MIT AI Lab by Colin Angle in the early 1990s to serve as an experimental platform for autonomous planetary exploration.

Similar to Genghis, it has a wide base and low centre of gravity. Only static balance was well studied for Attila (Ferrell, 1994). This is also to be contrasted with the
experimental robot employed in this thesis which deals with both static and dynamic balance. As for implementation, Attila employed multiple microcontrollers (i.e. each leg had an independent microcontroller that communicated with a master microcontroller). There were two microcontrollers for each of the six legs, one for sensor information processing and the one for behaviour controlling. Both microcontrollers were linked to the main processor via an onboard local network. Each leg had a ground contact sensor, joint position sensors, force and proximity sensors. These sensors worked together to provide sensing information for activation of various behaviours. With regards to the design of SA and behaviours, Attila was much more complicated than Genghis. For example, in order to fit in the role as an experimental platform for autonomous planetary exploration, Attila had a behaviour known as *sleep*, which is used to save energy so that it could survive in unknown environment.

### 2.1.3. Raibert and his running robot

Like Brooks who started the idea of SA, Raibert is one of the pioneers who investigated the balance issue of legged robots. The studies and investigations of Raibert make valuable contributions to the art of legged robot locomotion, especially in the area of balance. Koditschek and Buehler (1991) stated that the robots and control methodology of Raibert and his colleagues changed the outlook of the robotic community for legged robots to carry out high speed and dynamic locomotion (i.e. hopping, running, etc.). Even at the present time, Raibert’s control methodology still remains a popular mechanism for legged robots to achieve high
speed locomotion.

As mentioned previously in the first chapter, balance, particularly dynamic balance, is the central topic in the research undertaken by Raibert in many of his studies (1985; 1987; 1989). His investigations started with a one legged robot, as illustrated in figure 2.3 (Fiorini et al., 1999). It was easier to study the balance issue with only one leg. In addition, a robot with one leg has no other choice but to employ dynamic balance to prevent it from falling over. Raibert used a model of an inverted pendulum as the main controller mechanism for the robot in implementing dynamic balance. As illustrated in the figure, the robot has a thrust leg which generated the $X$ direction movement and a joint to connect the leg to the hip which generated $\theta$ direction movement, resulting in a 2 DOF system.

![Figure 2.3: Raibert's one leg hopping machine](referenced from Fiorini et al., 1999)

The controller consists of three modules, which control the speed of running, attitude of the body and height of hopping. The control mechanisms focused on the stabilization of bouncing height of the robot, dynamic balance of the body, tension control of the leg and manipulation of the locomotion speed of the robot. As the robot is “static unstable” (i.e. the robot will fall down if taking no action), it must
use constant movement to compensate for the instability. The basic idea of the inverted pendulum control mechanism is to adjust the $X$ movement of the leg and the $\theta$ motion of the hip so that the $ZMP$ could be kept as close as possible to the ground contacting area of the leg of the robot. A bigger $\theta$ value will create a larger horizontal force, resulting in a faster locomotion speed of the robot. At the same time, a bigger $\theta$ value will reduce the vertical force of the leg motion, resulting in a lower hopping height. On the other hand, a smaller $\theta$ value will result in a slower locomotion speed and a higher hopping height. Similarly, a bigger/faster $X$ movement will result in faster motions (both horizontal and vertical) of the robot. However, too big/fast $X$ movement of the robot will increase its dynamic instability, as it is hard to keep the $ZMP$ within its ground contacting area if motions of the robot are too fast. The one legged robot hopped, using a gyroscope as the balance sensor which functions like the inner ears of animals. It could hop at a constant speed, maintain balance in motion and follow certain paths.

The central idea of Raibert’s methodology is to use the movements of a robot to achieve dynamic balance. Movements of a robot will generate dynamic momentum, which in turn will compensate for any temporary instability of the robot. This principle was also employed by this thesis in designing some of the behaviours. As stated by Herr and McMahon (2000), colleagues of Raibert in the Leg Laboratory, “walking is harder than running”. To support their argument, Herr and McMahon used an example of what we usually do when we stumble in motion. In most circumstances, we will run a few more steps forward to prevent ourselves from falling over, rather than walking or stopping.
As a result, Raibert’s robots ran rather than walked. He applied this running (hopping) mechanism into his one, two and four legged robots. His investigations found that the differences in the number of legs did not make a significant difference in designing the control mechanism. All the legs were carrying out the same actions at the same time. As a result of the hopping/running nature of his robots, only dynamic balance was studied comprehensively in his investigations. A picture of one of his two legged robots is given in figure 2.4. It can be seen from the picture that its legs are telescopic, which is not a biological feature and is more easily controlled from the perspective of this viewpoint. This is to be contrasted with the biologically inspired multi-jointed leg of the experimental robot (Figure 3.1) employed in this thesis, which is more difficult to control because of the multiple degrees of freedom of limb movements.

Figure 2.4: Raibert’s two legged hopping robot (Raibert, 1985)

2.1.4. Hirose and his walking robots – the TITAN series

As one of the world-leading groups in research and development of autonomous
robots in the robotic community, Hirose and his colleagues of the Tokyo Institute of Technology and the University of Tokyo have developed a range of various robots, including walking robots, snake robots, wheeled & crawler robots, colony robots, tethered robots, service/inspection robots, medical robots, and planetary exploration robots, etc (Hirose et al., 1986; Hirose and Kunieda, 1991; Hirose et al., 1985; Hirose et al., 1984; Hirose and Yokoi, 1998; Hirose et al., 1989). Of all these robots, walking (legged) robots is one of their main areas of research interest and has won this group some of the world-renown acknowledgements in legged robot research and application. Hirose (2002) stated that legged robots have some advantages that are less achievable by wheeled or tracked robots. Legged robots can walk on very coarse environments on which wheeled or tracked robots would be unable to move. In addition, legged robots can astutely choose optimal paths without the requirements for constant roads/tracks as otherwise normally required by wheeled or tracked robots. As a bonus, legged robots can even utilize their legs as hands to carry out various operations that could not be achieved by wheeled or tracked robots, for example grabbing.

Biologically inspired approaches to control motion of robots were employed in their research. One of Hirose's typical robots is the TITAN IV (Yoneda and Hirose, 1992), as shown in figure 2.5. It weights about 160 kg, with four legs each approximately 1.2 metre in length. The name TITAN comes from the acronym of Tokyo Institute of Technology Aruku Norimono (the last two words stand for Walking Robot in Japanese).
After TITAN IV was built, it was displayed at the Science Exhibition at Tsukuba in 1985. In the exhibition, it showed the capability of walking on a stage with three different step levels and carrying out various locomotion gaits, including static walking, crawling, dynamic walking and even trotting, in which it trotted through the usage of two diagonal legs alternatively. A noteworthy aspect of TITAN IV is that it could transfer its locomotion gait from static walking to dynamic trotting as the locomotion speed increased. Yoneda and Hirose (1992) call this transfer a "dynamic and static fusion gait". Dynamic balance was achieved by using a methodology known as “sway compensation”. The main idea of sway compensation is to keep projection of all the forces (e.g. inertia and gravity) acting on the body of the robot within the supporting area formed by the supporting legs (Yoneda and Hirose, 1995).
A typical control mechanism of TITAN IV is given in figure 2.6 (Ridderström, 2003). As illustrated in the figure, the control mechanism consists of three hierarchy levels, from level A to C. The Level A control module is involved in global locomotion path planning. It creates path commands and passes those commands to the level B control module. Level B control module is called Intelligent Gait Generation module. It receives global path commands from Level A module and modifies these commands based on real-time terrain conditions that the robot is interacting with. Another task carried out by the level B module to is generate the locomotion gait (both horizontal and vertical) of the robot, for example which leg is to be on the ground, which leg is being swung forward, when a leg is to be lifted off ground, when a leg is to be put on the ground, etc. The level C control module is also known as the “Emergent Motions (behaviours)” module. It includes the balance (static and/or dynamic) module, body posture control and servo controls. It manages emergent motions (behaviours) using reflexes. For example, when a leg is trapped in
a hole, this module will trigger a reflex to lift that trapped leg from the hole. When a reflex of this module is activated, all other actions incurred by other modules will be suspended until actions of this reflex have been finished.

Hirose's latest TITAN series walking robot is the TITAN VIII (Kurazume et al., 2002), as shown in figure 2.7. Its walking control mechanism is similar to TITAN IV. However with TITAN VIII, an improvement in the ordinary sway compensation strategy, known as the 3D sway compensation mechanism (Kurazume et al., 2002) was employed in dealing with dynamic balance. The 3D sway compensation trajectory includes the calculations of the longitudinal and latitudinal (vertical) direction forces, besides the lateral force. In addition, feedback mechanisms that make use of sensor information were also applied within the robot in order to let the robot walk on uneven ground. Compared to the experimental robot in this thesis, TITAN VIII has a wider base, lower centre of gravity and bigger feet which make it easier to achieve balance during walking.

Figure 2.7: TITAN-VIII (Kurazume et al., 2002)
2.1.5. Kimura and his walking robots

Kimura and his colleagues of the National University of Electro-Communications in Japan is another robotic research group that has worldwide recognition. His research areas include autonomous walking robots, cooperation of a group of robots and cooperation between human and robots. Of which, biologically inspired dynamic walking and running legged robots is one of the focal areas that Kimura is interested in and has investigated this topic in many of his papers (Kimura et al., 1999a; Kimura and Fukuoka, 2000; Kimura et al., 2003a; Kimura et al., 2001; Kimura et al., 2003b; Kimura et al., 1999b; Kimura et al., 2000). One of his typical robots is Patrush (Kimura and Fukuoka, 2000), as shown in figure 2.8. Its legs are driven by servo motors and has 3 joints for each leg and two micro switches for each foot. It can walk up and down slopes and steps, walk through ground with humps and walk over obstacles. The walking strategy of Patrush is implemented through the use of a neural oscillator integrated with some feedback mechanisms (reflexes), to create a control mechanism known as “coupled-dynamics-based motion generation” (Kimura et al., 2003a), as illustrated in figure 2.9. The neural oscillator model was developed by Matsuoka (1985) and previously used in simulation by Taga (1995). The interactions of the neural and mechanical systems with the environment create emergent walking behaviours of the robot.
A typical CPG based on the neural oscillator concept employed by Kimura is given in figure 2.10 (Kimura et al., 2003a). There are two types of neurons in the oscillator: the Extensor Neuron and the Flexor Neuron. The output of the Extensor Neuron will result in the robot being able to extend its corresponding leg and the output of the Flexor Neuron will result in the robot being able to flex its corresponding leg. In the real implementation, output signals of both neurons in
most circumstances will be mixed with the signals (extend or reflex) generated by various reflexes before they were physically passed to the actuation system. There is a CPG for each leg of the robot. Outputs of neurons of all the CPGs result in either extending or flexing of the individual leg of the robot, in other words, generating various locomotion gaits.

Figure 2.10. A typical CPG based on neural oscillator

(Kimura et al., 2003a)

It could be seen from the above discussions that Kimura's approach is to combine a CPG (Central Pattern Generator) with reflex. The CPG is built upon the neural oscillator, as mentioned above. It responds to sensing inputs by changing the state and output of its active/inactive phase. The reflexes include stretch reflex, spinal reflex, extensor reflex and flexor reflex. They receive sensing inputs and outputs controlling torques as responses. They work together to enable the robot to exhibit
dynamic walking behaviours through interaction with the walking environment. This approach has some similarity to the approach proposed in this thesis. However, the implementations of both the CPG (loosely related to the CPP in this thesis) and the reflex in Kimura's model, are significantly different to what was done in this thesis. In Kimura's methodology, the CPG was built upon the neural oscillator model that is composed of extensor neurons and flexor neurons. Output of these neurons is either directly applied to or mixed with signals created by reflexes and then applied to the actuation system (e.g. a PID controller). There is one CPG for each leg. This is to be contrasted with the experimental robot employed in this thesis with only one CPP, and is not based upon any neural oscillator. The CPP was programmed to periodically output preset PSS associated with the various “walking” behaviours built for the robot. With regards to the reflex in Kimura's implementation, there are only two types of reflexes in Kimura's model: the stretch reflex and the flexor reflex. Activation signals of these two reflexes were used to modify the outputs of a CPG. This is to be contrasted with the behaviours built for the experimental robot employed in this thesis, which could process input signals and in reaction to these signals, appropriate actions for controlling the robot were executed.

Kimura's latest robot is the Tekken-II (Fukuoka et al., 2003), as shown in figure 2.11. Tekken-II employs a similar walking control strategy as that used in Patrush. A significant difference to Patrush is that Tekken-II has spring mechanisms around the hip knee joint to achieve better energy efficiency. It weights about 4.3 kg (including onboard power batteries). It can walk both indoor and outdoor. When walking

\[1\] For definitions of parameters and details of the neural oscillator, please refer to the paper Kimura et al., 2003a.
outdoors, Tekken-II can walk over ordinary ground with spotted grasses and pebbles. It can also walk on paved tracks.

Figure 2.11: Tekken-II (Fukuoka et al., 2003)

2.1.6. Rodney, the walking robot and the evolutionary approach (Genetic Programming)

Evolutionary approaches for robotics can be viewed as a branch of the biologically inspired approach. It is based upon the genetic programming (GP) technique (Holland, 1975), which implemented the "fitness for survival" principle, following Darwin’s theory of evolution. The basic idea of GP is that given a target goal, which is a problem to be solved, the system defines a population, which is the combination of individuals. In a generation, each individual performs its task and then the performance of each individual is valued and scored. The “good performers” will receive a good rank and those not so good performers will receive a lower rank.
Next, each individual will reproduce itself with the others, based on the rule that individuals with a higher rank will be given more opportunity to reproduce. As a result, the overall performance of the new generation will be better than the previous generation. Such a process will continue until the reproduction reaches a so called “sufficient generation”, in which the best performer of that generation will represent the best solution for the problem. GP can be used in many applications, providing that individuals of a system can be evaluated, compared and ranked.

One of the typical robots that employ the evolutionary approach is Rodney (Lewis et al., 1992), a six legged, motor controlled, Brooks-style robot that can walk forward and backward. A picture of Rodney is shown in figure 2.12. It is about 36 cm long and 13 cm wide. Its legs have two degrees of freedom and are actuated by servo motors, which can provide swing and elevation motions for the limbs. Similar to Brooks' walking robot (e.g. Genghis (Brooks, 1989)), it has a wide base and low centre of gravity. Only static balance is employed in Rodney. With regards to its walking strategy, it employs a trajectory planning schema using neural oscillators, which constitutes a network of neurons. This is to be contrasted with the SA approach employed in the experimental robot of this thesis which emphasises the “sensing-reflex” concept. Genetic Algorithm (GA), which is a branch of GP, was employed to train the robot to evolve two types of oscillator networks in a staged evaluation approach. The first type is an oscillator network for controlling the movement of each leg and the second type is an oscillator network for coordinating individual legs.

As Rodney has limited onboard computing capability, GA computing was performed
by a separate simulator known as GENESIS (Grefenstette, 1987). Neural signals, outputs of the GA computing, will then be downloaded into Rodney for execution. Results of the execution, which are the walking behaviours of Rodney were evaluated and input as feedbacks for the GA computing.

The controller of Rodney pioneered some of the works in applying GA to legged robots for walking. This approach has the advantage of not requiring a complete understanding of the dynamics between the robot and the environment for constructing a walking controller. Lewis et al. (1994) stated such implementations, if developed sufficiently, could provide a generic methodology for building and controlling complex robots that could be applied in the real world.

![Figure 2.12: Rodney(Lewis et al., 1992)](image)

2.1.7. SCAMPER, the running robot from the Furusho Laboratory

Furusho and Sano, as well as their colleagues in their laboratory of Osaka University, have investigated the topic of robotics for more than 20 years. Their research areas include walking robots, robot motion control, robot arms, robot force
control, etc. Similar to other robotic research groups in Japan, walking robots, including biped and quadruped robots, are one of their primary research areas. They have developed various walking strategies for those legged robots by carefully studying the walking characteristics of human beings and animals, such as dogs, cats and horses.

One of their typical walkers, SCAMPER (Furusho et al., 1995), as shown in figure 2.13, is a quadruped robot which was developed to analyze dynamic walking behaviours of legged robots. It had 2 degrees of freedom per leg and was able to run using a gallop and bound gait. Talebi et al. (2001) stated that SCAMPER is one of the quadruped robots which had been reported in the literature that could gallop and bound. Walking and running strategies of SCAMPER involved a hierarchical controller that was made up of a high-level motion trajectory planner, which is based upon the model of an inverted pendulum, and low-level feedback from leg sensors (e.g. joint position sensor). This is to be contrasted to the experimental robot in this thesis where the fundamental principle is based upon the concept of SA. Leg controlling of SCAMPER involves three types of controls: free rotation, position control and velocity control. It splits a complete walking/running cycle into 8 states and switched the two joints per leg between the three types of controls mentioned above. SCAMPER achieved running (galloping) via electric motors located at the joints of each leg, employing a speed control mechanism. Estremera and Waldron (2000) called such control mechanism the “Leg Thrust Control Method”. It includes Leg Lift off Speed Estimation, Closed Loop Leg Thrust Control and Open Loop Leg Thrust Control.
Other robots built and investigated by Furusho et al employed a similar controller to that of Scamper. For example, in the robot COLT (Sano et al., 2000), a similar trajectory planner was implemented with an enhancement to subdivide the planner into sub-states. States are decided by the situations (known as phases in their paper) of legs.

2.1.8. BISAM and the mammal-like locomotion robots

Ilg and his colleagues have investigated mammal-like locomotive robots for many years. The terminology of “mammal-like locomotion robots” refers to robots designed to have a body and leg structure resembling that of a mammal and is also designed to exhibit walking behaviours similar to that of a mammal. Their research emphasized applicable walking strategies that could apply to complex walking robots with a high level of complexity in mechanical and body/leg structure, which had a certain level of similarity to their biological counter parts. To achieve this goal,
Ilg et al. has been investigating mammal locomotion and applied various approaches to their walking robots, for example the walking strategy based upon neural oscillators for gait generation (Ilg et al., 1998b), leg trajectory learning (Ilg and Scholl 1998) and trotting posture control using Reinforcement Learning (RL) (Albiez et al. 2001b).

BISAM (Ilg et al., 2000), which got its name from the acronym of “Biologically InSpired wAlking Machine”, is the main experimental robot built by Ilg and his colleagues for their investigations. A picture of BISAM is shown in figure 2.14. It was built to investigate mammal-like locomotion and gaits. It weighted about 23 kg and was about 70cm in height. It consists of the body, four legs and a head. Unlike most quadruped robots that have a rigid body, the body of BISAM has four segments that connect via five rotary joints, resulting in 5 degree of freedom of the body. This characteristic gives BISAM very good freedom of body posture in motion. In addition, BISAM was able to rotate its shoulder and hip. Each leg has three degrees of freedom with limbs connected by rotary joints. All joints of BISAM are driven by DC geared motors. With regards to the control strategy employed in BISAM, it is based on "reactive posture control" (Albiez et al., 2001). Body and leg posture was controlled by a set of well designed reflexes, coordinated by a centre reflex coordination unit. This is to be contrasted with the experimental robot employed in this thesis which distributes the coordination into individual behaviours and implements it as various suppression rules for each individual behaviour. Nevertheless, the experimental robot does have a CPP which in context can be viewed as having certain coordination functionality. However, the CPP in this thesis
is actually used to generate rhythmical PSS only and does not deal with the details of behaviour coordination.

There are two parts in each reflex of BISAM, which were described as a sensor processing unit (part 1) and a controller (part 2) implemented using fuzzy control. The sensor processing unit was responsible for processing, filtering and conditioning sensor input information to standardize them as input signals for the controller. The controller was implemented using a classic PD fuzzy logic, consisting of five fuzzy sets: negative large, negative small, zero, positive large and positive small. Outputs of the fuzzy controller were used to adjust leg and body postures and movements.

![Figure 2.14: The BISAM, a mammal-like four legged walking robot (Ilg et al., 2000)](image)

### 2.1.9. Bowling and Harmeyer’s legged robots for agile locomotion

Bowling and Harmeyer of the Robotics and Dynamic Systems Laboratory of University of Notre Dame, Indiana, USA, have investigated legged robots to achieve so called agile locomotion (Bowling and Khatib, 2005; Bowling and Kim, 2003; Bowling and Kim, 2006; Harmeyer and Bowling, 2005). Their work concentrates on how robots could utilise ground contact and impact forces to accelerate the
locomotion speed, based on performance analysis of legged locomotion. An important characteristic of such locomotion is that performance of the locomotion is non-linear in relation to input signals. As a result, statistical based optimization methodologies were employed in their approach to deal with such non-linear problems rather than linear analysis. Bowling et al. (2006) classified locomotion into two groups: periodic and non-periodic. In the former group, locomotion exhibits a periodic pattern, which is commonly known as gait locomotion. In the latter group, locomotion is more to concerned with force and acceleration, and therefore referred to as agile locomotion in terminology by Bowling et al.. As claimed by Harmeyer and Bowling (2005a), a more thorough investigation into agile locomotion could lead to some novel design principles for the locomotion of legged robots. Analytical approaches were employed in their research, in which intensive computation was used to deal with dynamic linear and angular motion, force and moments. This is to be contrasted with the biologically inspired approach for the experimental robot employed in this thesis, which does not have nor rely on any intensive computations.

Figure 2.15 presents a photo of the latest legged robot of the Robotics and Dynamic Systems Laboratory, the Hexapod, which was used as an experimental platform for many of the investigations of Bowling and his colleagues. It weights about 1.3 kg and around 22 cm in height. As shown by the figure, the robot has a much larger ground support area formed by its supporting legs that are on the ground, comparing to that of the experimental robot of this thesis.
2.1.10. The latest works on bipedal robots

Recently there has been an increase in the amount of published work on bipedal (humanoid) robots. Research on bipedal robots has a broad scope, including design, dynamics analysis, balance control, motion control, motion planning and machine learning, etc.

One of the most popular known bipedal robots is the Honda ASIMO (Honda, 2006), as shown in figure 2.16. It weights about 52 kg, with dimension of 120 cm high, 45 cm wide. It is driven by servomotors and can walk/run up to 1.6 km per hour. It has 6 DOF for each leg, 5 DOF for each arm, 2 DOF for the head and 1 DOF for each finger. ASIMO has two control units, one for walking/operating control and one for wireless transmission control. The walking /operating control unit consists of floor reaction control, target ZMP (zero momentum point) control and foot planting location control. The walking strategy of ASIMO was developed through carefully
studying and investigating the walking behaviours of human beings. ASIMO is one of the bipedal robots that could exhibit stable locomotion behaviours like human. From the early stage of stumble walking, ASIMO gradually improves its balance and movement control techniques and is now able to exhibit stable walking and running behaviours similar to that of humans.

Figure 2.16: Honda ASIMO (Honda, 2006)

Another important area of work on bipedal robot is through the usage of the technique called “Passive-Dynamic” (McGeer, 1990), which emphasizes energy efficiency. In particular, the research work of Ruina of Cornell University, Collins of University of Michigan, Tedrake of the Massachusetts Institute of Technology (MIT) and Wisse of Delft University of Technology of Holland all of which were presented in the February issue of the journal Science (Collins et al., 2005). The walking strategy known as “Passive-Dynamic” (McGeer, 1990) was employed by each of
these researchers. Walking robots using this walking strategy maximise usage of the mass of the leg and body in the leg swing actions to create a simple, easily controlled and energy-efficient walking. This is in contrast to ASIMO, for which energy consumption is not its primary concern. Even though ASIMO can walk and run like human beings, it is not energy efficient and consumes as much as twenty times the energy used by humans for the same activity, as stated by Ruina in the report written by Williams (2005).

With regards to the control mechanism, Ruina and Collin employ a hierarchical finite state machine (FSM), which consists of a group of simple behaviors. These simple behaviours when receiving satisfactory input signals can be activated between each other to enable the robot to walk. For example, in the first state *Left Leg Swing*, actuators of the left leg are switched off to enable the left leg to passively swing forward by utilizing the weight of the leg. The swinging left leg is locked at the knee strike, where the sensor at the knee detects the locking action and signals the FSM to move to the next state, *Right Toe-Off*. In this state, the actuator acting at the right foot is switched on to extend the foot. When the right foot is fully extended, the FSM changes its state to *Right Toe-Return*, where the associated actuator is activated to retract the foot and signals the FSM to shift to the next state, *Right Leg Swing*. This state will enable the right leg to passively swing forward, which then activates the foot and toe action of the left leg. Such an iteration through the states of the FSM forms a full walking cycle of the robot and will continue until the robot stops walking.

A picture of the experimental robot used by Ruina and Collin is given in Figure 2.17.
One significant difference of this robot as compared to the experimental robot employed in this thesis, besides the number of legs and actuation system, is that the feet of Ruina and Collin’s robot are much larger than those of the experiment robot used in this thesis.

2.2. Review of techniques employed in this thesis.

This section is concerned with providing background information of the techniques employed in the thesis in both the design and implementation phases of the research.
This includes the SA, Object Oriented (OO) programming and Reinforcement Learning.

2.2.1. The biological inspired approach and the Subsumption Architecture

A significant principle of the biologically inspired approach is to employ distributed and component (behaviour) based control mechanisms. Dating from the early works of Rygg (1893), it is commonly accepted that the motions of individual legs in legged robots are manipulated by independent control systems (Cruse et al., 1995). Further investigation into locomotion of animals found that there is a high level locomotion control centre (Shik and Orlovsky, 1976), the Central Pattern Generator (CPG), that coordinates the actions of individual legs to produce coordinating locomotion behaviours.

One of the better known biologically inspired architectures is probably SA (Brooks, 1986), where a system is made up of a hierarchical set of predefined basic behaviours, all of which operate in parallel and are triggered by certain sensor conditions. A behaviour is defined as a set of actions activated by certain sensor (physical or virtual) conditions for achieving a certain goal that will eventually facilitate the achievement of the final system target goal. There is a set of suppression rules that enable higher level behaviours, when triggered, to suppress lower level behaviours. Figure 2.18 below illustrates a typical SA. As shown in the figure, condition \( n \) is associated with the activation of behaviour \( n \). After a behaviour has been activated, and if there is no higher level behaviour being activated at the same time, it will take control of the actuation system. However, if a higher level
behaviour has been activated at the same time, it will suppress any lower level behaviour and will thus take control of the actuation system. The real-time interaction of these behaviours can potentially generate some complex emergent behaviours (Brooks and Flynn, 1989) which result in an intelligent system (Brooks, 1991). Other biological approaches include Behaviour-Based architectures (Arkin, 1998; Mataric, 1992) which are an extension of the SA, Neuron architectures (Brown, 2000), Neuro-muscular architectures (Cruse et al., 1998), Neural Oscillator (Kimura et al., 2000; Matsuoka, 1985), Genetic Programming (Lewis et al., 1992), Imitation (Nicolescu and Mataric, 2001) and Articulated Control (Mataric et al., 1999).

![Subsumption Architecture](image)

*Figure 2.18: The Subsumption Architecture*

### 2.2.2. Objection Oriented Programming

The Object Oriented (OO) approach has become a popular methodology for system design and implementation. It emphasises the reusability, integrity, extensibility and maintainability of a design and implementation. Lewis (1995) stated that the term, OO, has existed for at least two or more decades. The first notion of an Object
appeared in a computer programming language known as Simula (Birtwistle et al., 1973). Since then, the OO design methodology has been greatly advanced to become one of the most popular system design solutions, replacing the traditional module (structure) approach. The essence of the OO design principle is to encapsulate data and its processing methods together, making integrated units that are simple, compact, and elegant (Lewis et al., 1995). Such Object units contain data and methods used to access and manipulate that data. Chorafas and Steinmann (1993) defined an object as an entity that encapsulates information elements, as well as operations on those elements. An abstract version of the same type of objects is called a Class, which defines a template for creation of concrete entities. The relationship between Object and Class can be illustrated as follows: an Object is the physical instance of a Class while a Class is the abstract template of an Object.

There are four essential characteristics in OO: encapsulation, inheritance, polymorphism and instantiation. Encapsulation is the most basic principle in OO, which is a technique used to reduce dependencies among individual units through defining strict external interfaces (Snyder, 1986). It packs data and operating methods into a single entity to give the user a generic reference interface to implement the integrity principle of OO. Inheritance is the main mechanism to implement the reusability principle. Through inheritance, new objects (and classes) can be built based upon the existing one, with new characteristics (data and/or methods) being added to existing ones. Polymorphism is another important aspect of OO. It enables specific objects to employ the same interface to perform different functionality, depending on the real-time circumstance of the object entities.
Instantiation is also a mechanism to implement reusability, and is used to create specific objects (instances) from their respective class (template). Rumbaugh et al. (1991) stated that the OO approach is good for problem understanding, information exchange, system modeling, policy abstracting and documentation.

This thesis adopts some of these OO principles in implementing the proposed four-phase walking strategy. Specifically, it is applied in the implementation of the SA. As such, a generic implementation methodology for SA is proposed, for which behaviours are implemented as objects, instantiated from the behaviour class and composed of reusable components.

2.2.3. Reinforcement Learning

Reinforcement learning (RL) (Sutton and Barto, 1998) is a machine learning method that learns by trial-and-error via interaction with an environment. In the history of RL, there are three primary threads: learning by “try and error”, optimal control and temporal-difference (TD) method. The “try and error” thread can be tracked back into the early 1910s, which started in the field of psychology (e.g. Thorndike, 1911). It formed some of the early works of artificial intelligence. As implied by the words, “Try and error” means that the agent employing the methodology will try different actions, which might lead to a success or failure (error), in various circumstances. Based on the results (success or fail), the agent will gradually learn to avoid those “fail” actions and to choose the “success” action. This is the most preliminary form of RL. The second thread, optimal control, started in the early 1950s. It was later better known as dynamic programming (DP) (Bellman, 1957) and Markovian
decision processes (MDP) (Bellman, 1957; Howard, 1960), in solving equations associated with control. Since its inception, DP and MDP have been extensively studied and expanded upon in the past half century into areas such as partially observable MDPs (surveyed by Monahan, 1982; Lovejoy, 1991), modified discounted MDPs (Puterman and Shin, 1978), infinite horizon DP (Bertsekas and Castañon, 1989), asynchronous stochastic approximation (Tsitsiklis, 1994), etc. The third thread, Temporal Difference (TD), although not as large a development as the other two threads, brings new and unique characteristics to the RL community and is widely adopted in modern reinforcement learning. As implied by the name, TD learns from the difference of temporally successive evaluations of the same instance. It was first implemented by Arthur Samuel (1959) and published in the IBM Journal of Research and Development. Since then TD has attracted increasing interest in the RL community and had achieved marvelous growth. A milestone in the RL history was that Watkins (1989) developed Q learning, which brought together aspects of all the above three threads, Try and Error, DP and TD. Another landmark in RL was the work of Tesauro (1992, 1994) which added the new feature of TD-Gammon in modern reinforcement learning in his famous self-teaching backgammon program. From the early stage of the preliminary try and error attempts, RL has now evolved into one of the most popular machine learning methodologies in the area of modern artificial intelligence.

Kaelbling et al. (2000) described a standard reinforcement learning model, shown in figure 2.19. As illustrated in the figure, a standard reinforcement learning system will consist of the following elements:
Figure 2.19: The standard reinforcement-learning model (Kaelbling, et al., 2000)

- $s$: a set of states of the environment (the environment might be known or unknown to the agent);
- $a$: a set of actions of the learning agent;
- $i$: Input function of the environment, i.e. how the learning agent perceives and interprets the environment;
- $i$: input signal to the system;
- $R$: Reinforcement function, i.e. how the system scores results of the actions taken by the agent;
- $r$: reinforcement signal;
- $T$: Transitions of states;
- $B$: Behaviours of the agent.

An explanation of how reinforcement learning works is given below:

1. At state $s$, the agent perceives and interprets the environment ($I$) and receives the input signal $i$;
2. The behaviour ($B$) of the agent chooses action $a$ to perform in state $s$ based on its selection policy;
3. Execution of action $a$ results in a transition ($T$) of the state of the environment;
As a result, the agent will get a reinforcement signal $r$ (reward or punishment) based on its scoring principle, the Reinforcement function $R$.

Based on this reinforcement signal $r$ the agent received, the agent adjusts its selection policy and starts another iteration of the try and error learning process.

The process will continue until the agent feels confident with the learning. At that time, the agent should learn that at any state $s$, it feels confident to select an optimal action $a$ in that state which will give this agent the best chance (probability) to achieve its goal. In the other words, the ultimate purpose of a reinforcement learning system is to learn an optimal action selection policy for each of the states, where such an optimal selection policy will enable the agent to select a so called “best action” which will give the agent the best probability (directly or not directly) to achieve its goal.

As can be seen in the above discussion, reinforcement learning does not require a precise model of the environment and/or training data of input/output pairs. In most practical RL systems, the agent learns from experiences without a model of its environment. In theory, although each instance of an experience may differ, the overall integration, based on a certain probability distribution existing in the environment (which may be unknown to the agent), should converge to the true value if an unlimited number of experiences have been tried. Most RL methods are based on this assumption. Under a state, the agent of an RL system performs an action, which is selected by using a specific policy, and gets a reward for the execution of this action. If the action is good for this state, it will get a positive
reward otherwise a negative reward is received. Based on this reward, the agent will adjust the selection policy. In theory, after an unlimited number of attempts, the agent should have tried all the possible actions under all possible states. Each state/action pair should have a value that specifies the probability of achieving the desired target goal if the specific action is taken while in the specific state. At the end of the learning process, it is expected that the selection policy has converged to the optimal policy.

Figure 2.20 provides an “overview map” of the Reinforcement Learning area and is included to provide an indication of the complexity of the entire field. It can be seen from the map that RL is a complicated and extensive area. However, for practical application, Q learning (off-policy) and Sarsa learning (on-policy) are the two main learning algorithms commonly used in practice. This is due to the fact that most of the practical applications are non-deterministic systems with no model of the environments and are learning from experiences using temporal difference (TD).

The highlighted boxes in Figure 2.20 show a trace that has been used in this thesis in the classification tree. This is also a typical use for practical RL application. The on-policy Sarsa learning algorithm was employed in this thesis and applied to the robot to learn the optimal walking cycle interval (details in chapter 5). The Sarsa learning algorithm is shown in equation 1 (Sutton and Barto, 1998)².

---

² This equation is commonly used in many RL documentations (e.g. Sutton and Barto, 1998). The “=" in the equation actually means “assign back to” and not the normal mathematical meaning of “Equal”. This representation is quite normal in computer programming such as “n=n+1”, which doesn’t mean that n is equal to n+1. It means to increase the value of variable n by 1 and then assign the new value back to variable n.
whereby, \( s \) and \( a \) are referred to as the current state and action of the robot; while \( s' \) and \( a' \) are referred to as the next state and action. Parameter \( \alpha \) is called the learning rate, whereby the larger the value, the faster the learning proceeds. Parameter \( R \) is the reward (positive or negative) received after action \( a \) has been performed in state \( s \). Parameter \( \gamma \) is called the discount factor, whereby the smaller the value, the more immediate feedback is taken into account in the learning process.

In equation 1, parameter \( TD \) is the acronym for Temporary Difference. Its values are calculated as: the reward received after action \( a \) has been performed in state \( s \) and taking the state to \( s' \), plus the Q value of \((s', a')\) multiplied by the discount factor \( \gamma \), and minus the original Q value of \((s, a)\).

Conventionally equation 1 uses the elements of \((s, a, r, s', a')\), which forms the word Sarsa, the common name for this learning algorithm. The purpose of the updating of the updating process of the algorithm is to improve the probability of achieving the target goal when taking action \( a \) in state \( s \). That is, even though taking action \( a \) in state \( s \) might not directly enable the learning agent to achieve its target goal (i.e. as discussed previously, status was taken to state \( s' \) after taking action \( a \) in \( s \)), the updating process tries to use the iterated Q value to tell the agent, in the long term, the chance of taking action \( a \) in state \( s \) contributing to the achievement of its target goal. In the end when the learning process has finished, at any given state, the agent can simply take the action with the maximum Q value of the state which will give the agent the best chance to achieve the target goal. However for Sarsa learning, as

\[
Q(s,a) = Q(s,a) + \alpha * TD = Q(s,a) + \alpha * [R + \gamma Q(s',a') - Q(s,a)]
\]
mentioned before, it is an on-policy learning algorithm, which means that the learning process always provides for a certain percentage of exploration when selecting an action in a state, even though the learning has already achieved a satisfactory performance.

Figure 2.20: An overview map of Reinforcement Learning
2.3. Chapter Summary

This chapter provided a review of some of the relevant existing work on legged robots and associated control methodologies, either using analytical approaches and/or biological approaches. A brief review of background information and techniques employed in the thesis, as a part of both the design and implementation, was also discussed.
Chapter 3: The Experimental Robot

3.1. Introduction

As mentioned in the first chapter, a four legged walking robot was built as the experimental platform. The size and proportion of the limbs and body of the robot mimics a mid-sized dog (with no "head" or “tail”). A picture of the robot is shown in Figure 3.1. It is about 71 cm long, 28 cm wide, 73 cm high when standing upright and weighs about 16 kg. The contact area of the feet with the ground is quite small (approximately 0.5 cm\(^2\) to 2 cm\(^2\), depending on the real-time situation of the robot's interaction with the ground). As shown in figure 3.1, the robot has a narrow supporting base, high centre of gravity, multiple degrees of freedom of the legs and limbs, and “small feet” together implying that it is an experimental platform where maintaining balance is a major difficulty. The legs are composed of “limbs”, each of which consists of sections of aluminium connected via bolts which form a simple joint. The robot has a different structure and function for its front versus back legs, similar to that of a real-life dog. Pneumatic cylinders are attached between each two adjacent limbs on either side of a joint. These act like “muscles”, providing the actuation through usage of solenoid valves. A potentiometer placed at each joint provides position information (i.e. one limb relative to the other). A group of mercury switches, mounted on the body of the robot, act as balance sensors. Ground
contact switches at the bottom of each foot act as touch sensors. A Motorola 68HC11 microcontroller with 64K memory acts as the onboard data processing and control unit. Several interface boards provide the following functions: A/D, Digital I/O, LCD, Keypad and solenoid valve drivers. Data and code is cross-compiled on a PC and then downloaded to the onboard microcontroller of the robot via a connecting cable. All robotic operations are performed by the onboard controller, executing and using the downloaded code and data.

![Figure 3.1: The experimental robot](image)

3.2. Mechanical Design

The mechanical materials used in building the framework of the robot consist of aluminium U sections. Sections (i.e. limbs) are jointed by bolts. Pneumatic cylinders (“muscles”) are mounted across joints between two limbs. This project required an experimental robot with a significant degree of control difficulty in terms of balance.
(i.e high center of gravity, narrow supporting base, small contact area with the ground, loose joints and multiple degree of freedom limbs) in order to evaluate the robustness of the balance and walking strategy developed here.

The robot’s mechanical structure is composed of three parts:

- The body
- Front legs
- Back legs

The body of the robot is a rigid framework constructed from aluminium sections. This framework houses the microcontroller, interface boards and solenoid valves. The framework also provides a “connection” point (i.e. hip joint) for the top of each of the four legs, and also the attachment point for one end of the pneumatic cylinders across each hip joint (i.e. “muscles” between “thigh” and “body”). As mentioned previously, the robot has a different structure and function for the front and back legs. Ideally, the front legs are used mainly for “steering” while the back legs for “driving” the robot forward. By “steering” it is meant that a leg leads the direction of the robot’s movement. By “driving” it is meant that a leg provides the main force to move the body forward. Figures 3.2-1 and 3.2-2 show a schematic of the side elevation of a front and a back leg respectively and the associated nomenclature.
The robot’s pneumatic cylinders act as “muscles” providing the actuation mechanism. Four pneumatic cylinders, labeled $\alpha$, $\beta$, $\gamma$ and $\delta$, were used on each leg as indicated in the schematic of figure 3.3 below. Cylinder $\alpha$ is used to control the movements of the foot relative to the shank. Cylinder $\beta$ is used to control the
movements of the shank relative to the thigh. Cylinder $\gamma$ and $\delta$ are used to control movements of the thigh relative to the body (two degrees of freedom).

Each of the pneumatic cylinders is only able to generate, when required, an “extension” or “contraction” force. Depending on the directions of the airflow into and out of each end of the cylinders, two types of movements can thus be generated: contraction and extension. These two types of movements are used in the contracting and extending actions of a limb. The airflow into and out of each end of the cylinders is controlled by “three-way” electrically actuated solenoid valves which are able to generate the following actions in the cylinder: extend, contract, and lock. In the last case, the “openings” at each end of the cylinder are blocked totally, resulting in the position of the cylinder being locked in its current position. All of the solenoid valves are attached on top of a single manifold which in turn is secured within the bottom of the “body” frame of the robot. SMC (2002) cylinders and manifold were used in the robot. Pneumatic tubing is used to connect each solenoid valve to the
appropriate cylinder.

Electrical actuation of the pneumatic solenoid valves was controlled using conventional Pulse Width Modulated (PWM) signals, generated from the onboard microcontroller. These PWM signals could be used to either extend or alternatively contract a cylinder. That is, the solenoid valves were driven in either a PWM “extend-lock” mode or a PWM “contract-lock” mode. For each of these modes, the larger the duty factor of the PWM signal the greater the “amount” of air attempting to flow out of the solenoid valve into the related cylinder, thus generating a faster and stronger cylinder (i.e. limb) movement.

3.3. Sensors

The robot has three different types of sensors: a balance sensor, contact sensors and position sensors. The balance sensor actually consists of eight individual mercury switches. For each of the four “tilt” directions (left, right, front and back) relative to the “horizontal”, there are two different limiting levels of tilt, approximately 5 degrees and 10 degrees providing different balance sensing information. Although these empirical values were determined by trial and error, the precise values were not found to be too critical. Essentially, the larger tilt value represents an upper limit beyond which, it is difficult for the robot to recover from toppling over. One point to note here is that although the switches which constituted the balance sensor on the robot are “tilt” activated, the movement of the robot actually accounts for more of the triggering of the balance sensor during the robot’s motions. This characteristic is referred to as “movement-activation” in this thesis. It is not hard to explain this, as it
is simply a matter of imagining the case of a cup of water while in a moving car. The water in the cup may spill out when the car is accelerating, braking or making sharp turns, even though the cup is never tilted. This characteristic may be seen as “false sensor” information from a “tilt” perspective but in fact is ideal from a dynamic balance perspective. It is actually good for the robot as this characteristic facilitates the investigation of the architecture’s capability to deal with dynamic balance when movement of the robot is considered. For example, a sudden acceleration of the robot’s motion in the forward direction may result in the backward direction component of the balance sensor being activated even if the body of the robot is not actually tilted. The robot needs to respond to this type of situation in order to combat the momentum from forward motion, otherwise it might topple over.

The “feet” contact sensors are two-state momentary contact mechanical push switches. They are physically located at the bottom of each of the four feet to sense whether a foot is in contact with the ground. This type of information was used by the robot’s behaviours to check if a leg is on or off the ground. The “joint” position sensors were simple potentiometers, mounted at each of the joints to sense the positions of one limb relative to another (or the body). A point to note is that all experiments in this thesis are conducted on a relatively firm surface as the robot is not equipped with pressure sensors that could provide more accurate ground contact information on a soft surface.

3.4. Onboard Microcontroller

The robot’s onboard microcontroller was a Motorola MC68HC11 based controller,
manufactured by New Micro Inc. (2000). The microcontroller had onboard 64K of memory, consisting of 32K RAM and 32K ROM. The ROM contained the operating system and some other supporting code. RAM was used to contain user developed software (i.e. the software developed to implement the proposed four-phase walking strategy). The microcontroller had an 8 Mhz clock, resulting in an effective 2MHz instruction cycle. Thus the onboard computation capability was limited. A number of interconnected interface boards also provided the following functionality: local LCD for status display and diagnostic purposes; local keypad for operator interaction; 16 channel multiplexed A/D for potentiometer measurements (one channel for each joint); 12 transistorised digital input channels for the balance sensor and contact switches and 16 digital outputs for the pneumatic solenoid valves.

The onboard microcontroller was connected to a personal computer (PC) via a serial communication cable to exchange code & data. Programming and compilation was carried out on the PC. Once this was finished, the resultant executable code on the PC was then either directly downloaded to the microcontroller or burned into an EPROM for the microcontroller. Execution of all the code and data generation was performed by the robot onboard microcontroller.

3.5. Descriptions of motions of the robot associated with its designed behaviours

3.5.1. Introduction

This section provides some description of the actions of the robot which are carried
out while using the proposed walking strategy. The purpose is to understand the
design of the specific walking behaviours and provide some documentation so that
results from this work might be reproduced by others.

Development of a dynamic mathematical model of the walking robot and the
environment would be extremely complex and has not been done in this thesis. Even
if a model were available, due to the complexity, uncertainty and the limited
computational capability of the onboard microcontroller, it could not have been put
to effective use within the experimental robot. As a biologically inspired approach
is used here, a more descriptive approach rather than a precise mathematical
formulation is used to narrate aspects of the movements of the robot while in
motion. As previously mentioned, the robot employs PWM signals in controlling its
cylinder (muscle) movements. This enables the robot to carry out a simple close-
loop feedback mechanism to control the cylinders, which is illustrated in figure 3.4.
In the event that a fast motion and/or a large force was required, the pulse width was
set to the full 100%. This is referred to as a "full duty" action (extension or
contraction) in this thesis. In other circumstances, a pulse width of 50% was
employed in controlling the robot's movements.
The robot’s movements involve quite complicated interactions of ground forces and leg forces. As previously discussed, each leg has four associated pneumatic cylinders which perform the actuation. Three of these cylinders $\alpha, \beta$ and $\gamma$ are associated with moving (rotating) the limbs “forwards” and “backwards”, while the last cylinder $\delta$ moves the legs inwards or outwards relative to the central “vertical” plane of the robot. Six types of behaviour were defined in the walking strategy and implemented: *Stand*, *Forward*, *Backward*, *Balance*, *Leg Down* and *Protect*. Although the complete details of these behaviours will be discussed in chapters 4 and 5, it is convenient at this stage to consider the motions associated with these behaviours. Behaviour *Stand* is used to support the robot’s weight. Behaviour *Forward* is used to move a leg of the robot forward. Behaviour *Backward* is used to push a leg of the robot backward in order to generate a counteracting ground force to move the body of the robot forward. Behaviour *Balance* is used to balance the body of the robot.
Behaviour *Leg Down* is used to force a leg of the robot to make contact with the ground. Behaviour *Protect* is used to prevent the robot from tipping over. Some aspects of the kinematics and dynamics of each of these behaviours will shortly be considered and illustrated.

An important point to note here is that in designing behaviour motions for the robot, there has been various “manual tuning” of some of the parameters, such as the length of time for a limb to extend/contract, the positions of limbs for various behaviours and phases, the tolerance of leg/limb position sensing, the maximum tilt degree of the body of the robot, etc. Without a certain level of “manual tuning” of these parameters, the experimental robot could not even stand up and maintain its balance. Theoretically, this is of course not the best way. However, given the complexity of the robot's interaction with the environment and the limited capability of the onboard microcontroller of this robot, this was found to be the most practical way of achieving the objectives in a reasonable time frame. Otherwise it would involve intensive computation beyond the capability of the robot system used in this project. As claimed by Albiez, Luksch, Ilg and Berns (2001), the approach of using a certain level of "tuning" has the advantage of making good use of the experience gained through the experiments by the experimenters.

### 3.5.2. Descriptions of behaviour motions designed for the proposed walking strategy

In this section, we describe some aspects of the motions of all behaviours designed in this thesis: *Stand, Forward, Backward, Balance, Leg Down* and *Protect* will be
illustrated. Each behaviour will carry out its preset actions when it is activated and try to complete all its actions once activated. Activation of specific behaviours will depend on the real-time situation of the robot's interaction with the environment (both externally and internally) and the priority of the behaviour itself, relative to all the other behaviours.

Each of the following sections is only concerned with a discussion and illustration of the motions associated with each of the behaviours. A detailed discussion of the behaviours themselves follows in the next chapter. In the following discussion of leg motions, the terms of $\alpha$, $\beta$, $\delta$, $\gamma$ refer to various cylinders fixed as actuators to a leg of the robot, as described previously (see Figure 3.3).

3.5.2.1. Description of motions of the Forward behaviour

The Forward behaviour is used to swing a leg forward. The motions associated with the behaviour consist of three stages of actions as indicated below.

- Preparation stage:
  Move a leg downward and slightly outward (with reference to the central vertical plane of the body) in order to push toward the ground strongly to create a counter force from the ground to move the robot's body upwards and to push toward the side opposite to the leg which is performing the actions. This involves a full duty extension of cylinders $\alpha$ and $\beta$. At the same time, a full duty extension of cylinder $\delta$ is used to generate a Z direction force (see Figure 3.3) to assist moving the body of the robot to the opposite side of this leg.
• Swing stage:

Give a full duty contraction to cylinders $\alpha$ and $\beta$ to contract the foot and shank to make room for the swinging action of the leg so that it does not touch the ground when swung forward. At the same time, fully swing the leg forward by extending cylinder $\gamma$ for a front leg or fully contracting cylinder $\gamma$ for a back leg. Overall the actions generate a force to swing a leg forward.

• Finish stage:

Lower the leg to make contact with the ground by extending cylinders $\alpha$ and $\beta$ until the foot is in contact with the ground (this is achieved by checking the associated ground contact switch sensor at the bottom of the foot). At the same time, contract cylinder $\delta$ to make the body incline toward the corner associated with this leg so that it is ready for the next sequential leg to move forward.

3.5.2.2. Description of motions of the Backward behaviour

Motions of this behaviour require the cylinder $\gamma$ to be contracted for the front legs and extended for the back legs. At the same time, the ground contact sensor of the leg are continually monitored to make sure that the foot of the leg is always in contact with the ground, extending cylinders $\alpha$ and $\beta$ if required. The overall leg actions generate a counteractive ground force resulting in forward movement of the robot. A point to note here is that while one leg is carrying out the Forward behaviour, the other three legs are simultaneously carrying out the Backward behaviours, but each with a different leg position relative to the body (see details in chapter 4). Interaction of the behaviours of all the four legs generates an overall
force to move the robot forward.

3.5.2.3. Description of motions of the Balance behaviour

When the balance sensor associated with a leg is activated (this may be caused by the inclination of the body exceeding the preset limit, for example 5 degrees, or a strong lateral movement), extend a leg by extending cylinders $\alpha$ and $\beta$ if the body corner associated with the leg is lower than the horizontal level (or the relevant part of the balance sensor is triggered by momentum) or contract a leg by contracting cylinders $\alpha$ and $\beta$ if the body corner associated with the leg is higher than the horizontal level (or the relevant part of the balance sensor is triggered by momentum).

3.5.2.4. Description of motions of the Leg Down behaviour

The Leg Down behaviour is used to immediately move a leg downward in order for it to make contact with the ground. This is achieved by extending cylinders $\alpha$ and $\beta$ until the ground contact sensor of the leg is activated.

3.5.2.5. Description of motions of the Stand and Protect behaviour

The Stand behaviour is used to lock a leg in its current position by closing both outlets of all the cylinders ($\alpha$, $\beta$, $\gamma$ and $\delta$). The Protect behaviour is used to reset the robot to a pre-defined "safe" posture, a position with which the body of the robot is not tilted which corresponds to the approximate leg positions as shown in figure 3.3.
3.6. Chapter Summary

This chapter presented aspects about the design and construction of the four legged walking robot that was used as the experimental platform in this thesis. The robot mimics a mid-sized dog (with no "head" or "tail") and has some characteristics such as a high centre of gravity, narrow supporting base, 3-d freedom of movement of limbs and small feet which made it a good experimental platform for investigating the issue of balance and the walking strategy proposed in this thesis. This chapter also provided a general description of the motions of the robot’s behaviours in preparation for the discussion of the biologically inspired walking strategy in the next chapter.
Chapter 4: The Design of the Four-phase Walking Strategy Using Behaviours

4.1. Introduction

An important aspect in the design of the walking strategy proposed in this thesis was the employment of distributed control mechanisms, as used in many of the biologically inspired architectures mentioned previously. From the early works of Howell (1965), it is now commonly accepted that the motions of individual legs of animals are manipulated by independent control systems. Cruse et al. (1995) stated that each of these control system centers manage the actions associated with the movement of one leg. In this thesis, these control systems are implemented as independent parallel Subsumption Architectures (SA) in the proposed walking strategy. At the same time, there is a “center” at a higher level that coordinates the movement between individual legs to enable them to produce collaborating locomotion behaviours (Shik and Orlovskii, 1965). In this thesis, the higher level center for coordination of locomotion is implemented as a simple Central Pattern Producer (CPP). In addition, there is a machine learning algorithm (implemented using Reinforcement Learning) incorporated in the walking strategy to allow the robot to learn an optimal time interval for the CPP to generate the rhythmical PSS via real-time interaction with the environment. It is found in this thesis that the CPP requires an optimal time interval to generate the PSS in order that the robot can
achieve a better walking quality. Section 4.3 of this chapter provides a detailed discussion on this matter.

4.2. The four phase walking strategy using behaviours

The proposed four-phase walking strategy associated with each leg consists of phase 0 to phase 3, as shown in Figure 4.1 to Figure 4.4. In these figures, there are 4 pre-defined positions of a leg relative to the body, namely $a$, $b$, $c$, $d$. The arrow underneath the position labels represents the direction of movement of a leg.

![Figure 4.1: Step 1 of the Four Phase Walking Strategy (left front leg off the ground and moving forward, others on the ground moving backwards)](image-url)

<table>
<thead>
<tr>
<th>Right front leg at Phase 2</th>
<th>Right back leg at Phase 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>d c b a</td>
<td>d c b a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Left front leg at Phase 0</th>
<th>Left back leg at Phase 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>d c b a</td>
<td>d c b a</td>
</tr>
</tbody>
</table>
In the above figures, a solid black line (e.g. left front leg of Figure 4.1, position a) represents the current position of a leg before its motions have been carried out in a
phase (i.e. at the start of that phase). A line consisting of “bold larger dots” (e.g. left front leg of Figure 4.1, position d) represents the finished position of a leg after its motions have been carried out in a phase (i.e. end of that phase). The other lines consisting of “grey smaller dots” (e.g. left front leg of Figure 4.1, positions b and c) indicate the various preset positions of a leg that were not employed in that phase.

The complete walking cycle of the four legs of the robot is illustrated in the order of figure 4.1 to figure 4.4, starting from step 1 (Figure 4.1) through to step 4 (Figure 4.4) and then cycling back to step 1 to repeat the steps. As illustrated in the figures, at phase 0, a leg (referred to as the leading leg by Howell, 1965) is lifted and swung forward. For the remaining three phases, the leg will be pushed backward (relative to the body). All four legs co-operate to generate the force to move the robot forward. While one leg is moving forward in the air, the other three legs are on the ground pushing backwards. The difference between phases 1, 2 and 3 is that the leg is at a different position relative to the body. A leg cycles through the four different phases in the order of …Phase 0 - Phase 3 - Phase 2 - Phase 1 – Phase 0 … .

Suppose the full movement distance of a leg (relative to the body) is defined as 100%, then the four leg positions d, c, b, a are defined to represent the leg positions at 100%, 67%, 33% and 0% respectively. Movement details of the four-phases are listed as follows:

- **Phase 0:** Move from position a to position d (in the forward direction, e.g. left front leg of Figure 4.1).
- **Phase 3:** Move from position d to position c (in the backward direction, e.g. left front leg of Figure 4.2).
• Phase 2: Move from position \( c \) to position \( b \) (in the backward direction, e.g. left front leg of Figure 4.3).

• Phase 1: Move from position \( b \) to position \( a \) (in the backward direction, e.g. left front leg of Figure 4.4).

By implementing this cycle of leg motion for each leg, a walking behaviour for the robot can be achieved. The legs in turn enter phase 0 in the order of left front (LF), right back (RB), right front (RF) and left back (LB) which is the normal walking gait for four legged animals. An independent SA with its own action execution unit (Francis, 1993) has been developed for each leg of the robot, resulting in four SAs functioning in parallel in the system. The four SAs have no direct communications with each other. They co-ordinate via real-time physical positions of the robot, interactions of the robot with the environment, and a CPP that will be discussed in the next section.

Each SA consists of six behaviours: \textit{Stand, Forward, Backward, Balance, LegDown} and \textit{Protect}. In total, there are 24 behaviours (4 legs with 6 behaviours per leg) running in parallel in the system. However at any one point in time, only the actions of one behaviour (referred to as the active behaviour) associated with each leg are being carried out by the robot. These 24 behaviours co-operate together to generate an overall emergent walking behaviour. The behaviours \textit{Forward} and \textit{Backward} are the only two behaviours needed for an ideal walking cycle. They are known as the ideal walking behaviours (IWB). The \textit{Forward} behaviour swings a leg forward from position \( a \) to position \( d \). The \textit{Backward} behaviour moves a leg backward a unit distance (e.g. From position \( d \) to \( c \), \( c \) to \( b \), or \( b \) to \( a \)) to constantly generate a counter
ground force to push the body of the robot forward. Theoretically at any point in time, if the *Forward* behaviour of a leg is triggered, the *Backward* behaviours of the remaining three legs are also simultaneously triggered, with a different phase for each of these other legs. However, in practice, this may not be the case as other behaviours (e.g. *Stand*, *Balance*, etc.) may be triggered to suppress the ideal walking behaviours in reaction to the robot’s real-time interactions with the environment during motion. The other four behaviours: *Stand*, *Balance*, *Leg Down* and *Protect*, exist to assist the robot’s walking (e.g. the *Protect* behaviour is used to prevent the robot from tipping over) and recover from bad situations. These four behaviours are known as assisting behaviours (AB) in the walking strategy. The *Stand* behaviour is employed when a leg completes its behaviour actions earlier than the other legs and waits for signals from the CPP. It can also be activated when a leg in some situations can not carry out its normal walking behaviours (e.g. being stuck). The *Balance* or *LegDown* behaviour is activated when the associated body corner of the robot tilts beyond a limiting angle (e.g. 5 degrees) relative to the horizontal surface or has a movement that results in the activation of the balance sensor. The former is activated if the leg is on the ground and the latter is activated if the leg is off the ground. The *Protect* behaviour will be activated when the robot is about to fall down (e.g. the body of the robot is tilted 10 degrees or more relative to the horizontal). This behaviour resets the robot to a pre-defined “safe” position, as shown in figure 3.1.

Hierarchically on top of the four parallel SAs is a CPP which is used to generate a predetermined rhythmical walking phase signal set (PSS) and then pass this generated PSS to the four legs for carrying out their associated behaviours. In each
time interval, a new PSS is generated and passed to the four legs. A PSS is defined as a four digit number, with each digit representing one leg. Each digit represents the current phase of a leg. The digit order from left to right corresponds to LF leg, LB leg, RF leg and RB leg. An example of a PSS is 0321 (see figure 4.1), which represents the phase signal set of phase 0 for the LF leg, phase 3 for the LB leg, phase 2 for the RF leg and phase 1 for the RB leg. To increase the ability of the robot to adapt to different walking environments, an RL learning algorithm was also integrated into the CPP to allow the robot to learn the optimal walking time interval (OWCI) for the CPP to generate the rhythmical PSS for an environment and to output it to the legs periodically at the set time interval.

Both static and dynamic balance strategies were considered in designing the behaviours. There is no doubt that static balance is utilised by the robot to support its body. At the same time, the robot attempts to employ a dynamic strategy wherever applicable during locomotion. That is, movements of the robot are used to compensate for temporary instability. For example, when a leg is performing its Forward behaviour to move a leg forward (the leg is off the ground), it will try to finish all the forward actions even if the robot’s body is falling, providing this “falling” is within an acceptable “body tilt” range. The movement corresponding to such a behaviour will create momentum which in turn tends to counteract, or at least delay, the falling. Our experimental robot currently does not have specific sensors that can determine if the centre of gravity of the robot is within its supporting area. This information could be used to verify if the robot is dynamically or statically balanced. In practice, static balance can be assumed as occurring if the robot does
not fall when there is no relative motion of its legs with respect to its body. In some experiments during development of the system, there were some occasions when the robot’s leg motion was suddenly frozen while in a walking mode, resulting in the robot falling. As the robot did not walk at a very high speed (approximately 3 to 10 metres/minute, depending on the terrain condition and real-time interaction of the robot walking behaviours with the environment), the contribution of the movement of the robot’s motion to its falling was limited. The inference made here is that when the robot maintains its balance while it is in motion, then some type of dynamic balance behaviour must be occurring. The activation of static or dynamic balance is non-deterministic. It depends on the real-time situation of the robot's interaction with the environment. However in general, the faster the robot is moving, the more often dynamic balance is expected to be occurring.

4.3. The Central Pattern Producer (CPP) and walking

Although there is no unanimous agreement on animals’ locomotion and the specific roles their body systems play during motion, it is generally accepted that an animal’s locomotion results from a combination of the spinal cord which contains a rhythmical Central Pattern Generator (CPG), reflexes that are in charge of locally responding to the peripheral stimulus, and some parts of the brain (e.g. the cerebellum and brain stem) (Grillner, 1981; Kandel et al., 1991; Shik and Orlovsky, 1976). The brain functionality was not implemented in this thesis. The reflexes have been interpreted as specific behaviours and implemented using four parallel SAs as illustrated before. Certain aspects of the CPG are implemented as a Central Pattern
Producer (CPP) in this thesis. The CPP functions like a CPG only in terms of generating and sending out PSS. Thus it does not fit into the definition of a CPG and is not built nor does it work in the same way as a real CPG. In order to avoid any confusion, the term CPP is used in this thesis instead.

In the design, the CPP is used to create a rhythmical PSS for all four legs of the robot. A learning functionality was also integrated into the CPP for it to learn the OWCI. Details of this learning functionality will be discussed later. The phase cycle created by the CPP is in the order of …0-3-2-1-0… for each of the four legs. This order of transition of phases is the same for all legs. However, there is a unit phase shift between two successive legs in motion (see Figure 4.1 to Figure 4.4). At each time instance, the CPP generates the phase signal for each leg which is then passed down to each of the legs. The legs carry out the associated actions, if the required trigger condition is satisfied (i.e. Trigger component returns TRUE). In the next time instance, the CPP changes the phase signals to the next sequential phase for each leg. The above steps are carried out repeatedly during the motion of the robot. The CPP acts as an “open-loop” coordinator of all the four legs as there is no feedback from the legs to the CPP. The robot can carry out its walking without the need of feedback from legs to CPP because the designed behaviours of its legs are able to process “feedback” (sensor information) signals themselves and carry out appropriate actions according to the real-time interaction of the robot with the environment. As for the physical implementation of the CPP in this project, it is implemented using an independent task (thread), running in parallel with all the behaviours. Its status (fired or not fired) and generated phase signal (phase 0, 1, 2, 3) are stored in some public
variables which can be accessed by the behaviour and its component objects.

The initialisation stage in the robot’s walking behaviour involves setting conditions whereby each of the legs is put to the preset phase of 0, 2, 3 and 1 for the LF leg, RF leg, LB leg and RB leg respectively. Given the Stand behaviour has no “trigger” conditions, it will automatically activate provided no other behaviours are active.

The sequence of behaviours is not deterministic but a typical scenario for walking is described below. When the CPP is first started, the Forward behaviour of the left front leg and the Backward behaviours of the remaining three legs are triggered. The Forward behaviour of the LF leg suppresses the Stand behaviour to become activated and moves the leg forward. The leg pushes downward and slightly outward (see Section 3.5.2 for details of leg motions) onto the ground (to generate an upward force against gravity), lifts off, fully swings forward and is placed down on the ground. At the same time, the Backward behaviours of the other three legs push backward on the ground to move the body forward. They cooperate to generate the necessary force to enable the robot to move forward. A smooth transition of leg phases occurs during movement. Visually, it is seen that the robot is walking forward. During these activities, other behaviours such as Balance or LegDown may be activated individually by each leg if the body of the robot tilts beyond a tolerant degree. If the Balance behaviour is triggered, it will suppress any lower level behaviour (e.g. Forward, Backward or Stand behaviour) to become the activated behaviour. Its actions involve adjusting the body of the robot to prevent the robot from tipping over. When the robot regains its balance, the Forward or Backward behaviours are again activated. In the worst case scenario when the robot is about to
fall down, the Protect behaviour is triggered to reset the posture of the robot to a
certain predefined “safe” position (see figure 3.1). This alternation of behaviours
may occur repeatedly until all the walking actions have been completed. After the
first "phase of walking", the phase of each of the legs is cycled by one phase (i.e.
phase 3 for LF, phase 1 for RF, phase 2 for LB and phase 0 for RB), so that it is
ready for the right back leg to be moved forward. Overall, the interactions within the
system will result in the generation of a combination of “preset” and emergent
walking behaviour that enables the robot to move forward. The process discussed
above is for one of the four legs and will occur simultaneously for all the four legs,
with an appropriate phase delay between each leg as previously discussed.

4.4. Setting of walking cycle intervals and integration of
reinforcement learning

4.4.1. About the walking cycle interval

One key issue in the design of this walking strategy is the time duration required for
(1) the CPP to generate the rhythmical PSS for each of the four legs and (2) for the
legs to execute the associated actions. This period of time is defined as the “walking
cycle interval” in this thesis, as mentioned previously. Settings of the value
associated with this interval will affect the speed and quality of the walking
behaviours in the robot. Theoretically, the faster the motion, the smaller the value
associated with this interval. However, there is a physical real-time limit. The robot
can not successfully walk if this value is too small, as the robot does not have
enough time to complete the individual walking behaviours. On the other hand, the quality of the walking is poor (i.e. triggering more assisting behaviours) if the walking cycle interval is too large. This is due to the fact that the robot makes use of dynamic balance to keep its balance while in motion. It needs constant movement to compensate for its temporary instability. Too large a value for the walking cycle interval reduces the momentum of the robot.

There is, therefore, an optimal walking cycle interval (OWCI) in the sense of being a compromise between walking quality and walking speed. As with biological systems, there is a “natural walking period” associated with the motion of an animal and is dependent on the “biomechanical” aspects of the weight, height, limbs, body and how they are attached to each other (Alexander, 1992). This issue is most easily experienced by anyone trying to walk either faster or slower than one's own natural speed. It is more uncomfortable (requiring more energy to maintain balance and move/hold the limbs), and it is partly in this sense that the “quality” of walking is reduced. This is the interpretation used in this thesis whereby the better the walking (i.e. quality), the more the ideal walking behaviours (IWB) are activated relative to the assisting behaviours (AB).

There are two ways to determine the optimal value of the walking cycle interval: empirically or automatically. Both methods have been employed in this thesis in the experiments carried out for implementing the proposed walking strategy. In early experiments, the empirical option was employed. The OWCI was set to an empirical value (i.e. 500 ms) based on experimentation and acquired experience through observations of the robot’s walking. The robot exhibits stable walking behaviours
with this empirical value of the walking cycle interval. Later experiments employed an automatic method to allow the robot to learn the OWCI. This was achieved by integrating a Reinforcement Learning algorithm into the CPP for the robot to learn this value via real-time interaction with the environment, thus allowing the robot to adapt the OWCI (when necessary) to the changing environment.

4.4.2. Design of the learning algorithm

4.4.2.1. Introduction

The goal of the learning in the system was to allow the robot to learn the OWCI while interacting with the environment. The previous discussion implied that the OWCI corresponded to the highest percentage of activation of the ideal walking behaviours. This is the interpretation used in this thesis whereby the more ideal the walking (i.e. quality), the more the ideal walking behaviours are activated relative to the assisting behaviours. Therefore, the percentage of the activation of ideal walking behaviours was used as the evaluation policy for the robot's learning algorithm. The following sections discuss the determination of the OWCI for the walking of the robot on various terrain conditions.

An RL algorithm was employed here to allow the robot to learn the OWCI through trial-and-error interaction with the environment. The actions of the implemented RL algorithm are defined so as to execute a range of discrete values associated with a discrete range of the walking cycle intervals (ms). The action space (i.e. the range of these values) must be chosen to cover and include the optimal value if something is
known about it. If there was little or no knowledge as to the optimal value, it would be best to start first with a “coarse grain” (e.g. 0, 100 ms, 200 ms, …) approach and then determine the optimal value at this granularity. Then successively “finer grains” should be chosen centred around the determined optimal value achieved in the previous iterations. The “fineness” granularity of the tuning is a compromise between the desired accuracy and physical limitations of the robot. Due to previous experimentations and the developed experiences in operating the robot, it was estimated that the optimal value was somewhere between 450ms and 600ms. Thus in this instance there was no need to first start with a coarse grain. Due to limitations in the speed and memory requirements of the onboard computer, a granularity of 15ms was chosen. The one step Sarsa learning (Sutton and Barto, 1998) was employed as the learning algorithm and implemented into the robot.

4.4.2.2. Learning Algorithm Design

As mentioned previously, the learning task was to learn the OWCI, which is interpreted as actions to be taken in the learning. The learning process takes place inside the CPP task. A learning episode starts immediately after each PSS was generated by the CPP and passed to the four respective SAs. After the learning episode has been started, it will select an action based on its selection policy and execute the action (i.e. creation of a delay for the CPP to generate the next PSS). Once the action has been finished, it will receive a reward or punishment by evaluating all the behaviours of the four legs. Based on this reward or punishment, it will update its Q values (Sutton and Barto, 1998) and start another iteration of the learning process. Updating of the Q values will improve its selection policy. This
sequence of iterations continues until the learning process terminates. At this time, the selection policy was expected to converge to the optimal policy, with the associated action being the OWCI. The specific details of the learning algorithm design and implementation is described in the next chapter.

4.5. Chapter Summary

This chapter proposed a four-phase walking strategy, inspired from four legged animals, for a four legged walking robot. The walking strategy is interpreted and implemented as four parallel SAs and a CPP. Specifically, each SA is composed of six specific behaviours: Stand, Forward, Backward, Balance, Leg Down and Protect. Interactions of these behaviours during real-time walking in an environment generates a combination of preset and emergent walking behaviours that enables the robot to walk. This four-phase walking strategy is the foundation for all the experiments carried out in this thesis.

This chapter also provided a detailed discussion on the functionality of the CPP and its importance to the robot's walking. It explains how the CPP coordinates the robot's walking by generating a rhythmic PSS in a walking cycle. It also discusses an important issue in the design of this walking strategy: the frequency (the time interval) in which the CPP generates the phase pattern, and methods to determine an optimal value for the interval, either by employing empirical values or automatically by using a machine learning mechanism.
Chapter 5: Implementation

5.1. Introduction

The four-phase walking strategy is implemented using four parallel SAs. As mentioned previously, the SA concept was introduced by Brooks (1986). This biologically inspired approach is an architecture that is based on lower level species of animals (e.g. insects). It is generally believed that such animals do not have clear “maps” of their living environment and can not “think” or “plan” in advance. They only have simple instinctual behaviours that can be triggered by some sensory activator. For example, the escape behaviour will be triggered when an approaching enemy is sensed. In spite of its limited number of simple instinctual behaviours, such animals can survive very well in nature. Though a single behaviour by itself may be simple, it is the interactions of many of these that can give rise to complex emergent behaviours. For some researchers, it is these types of emergent behaviours that produce an intelligent system (Brooks, 1989).

A SA application works in a simple, efficient and responsive way. It is often referred to as “Sensing – Reacting”. Such a system, responds to the environment according to a set of pre-defined behaviours which are triggered by certain sensor conditions. The main principles of Subsumption Architecture can be summarised as follows:
• There is no representation or model of the environment. It is a bottom-up (reactive) Artificial Intelligence (AI) architecture.

• It consists of a set of pre-defined hierarchy of behaviours. A behaviour is defined as a set of actions triggered by certain sensor (physical or virtual) conditions for achieving a certain goal which will facilitate the achievement of the final system target goal.

• All behaviours in a SA operate in parallel.

• Lower level behaviours can be suppressed by higher level behaviours according to the predefined behaviour suppression rules.

• New behaviours can be added into the system without changing the original architecture. However, it may change the rules of behaviour suppression.

As an aside, the inhibition function of input signals for behaviours proposed in Brooks’s original Subsumption Architecture (Brooks, 1986) are not considered in this thesis. When Brooks proposed the SA, he implemented the system using hardware suppression (i.e. physical electrical signals) and introduced the inhibition function. However in this thesis, software suppression (which can suppress the whole behaviour rather than electrical signals only) was employed and therefore does not need the inhibition function. Conceptually, the SA requires that all its behaviours operate in parallel. The ultimate solution is to implement each behaviour as a separate processor (e.g. Connel, 1990). However, in many cases, the SA is implemented on a single processor system. Depending on the complexity of these systems, two approaches are used:

• To emulate the parallel computation routines in a single-task environment (e.g. (Francis, 1993))
• To implement the parallel routines as multiple processes or threads in a multitasking system (e.g. (Jones and Flynn, 1993)).

Given that modern computer systems can multi-task, it is easier to implement SA using multi-tasking. This approach was adopted in this thesis, whereby each behaviour was implemented as a separate thread (i.e. a task).

To facilitate reusability, an implementation framework for SA is described in the first part of this chapter. After the implementation framework has been proposed, its application to the robot in order to physically implement the proposed four-phase walking strategy is then detailed in section 5.2.3. This approach is an attempt to provide a coherent and modular framework for implementing the SA in robotics. It provides a specific methodological framework that allows the developer to concentrate on the design of behaviours and suppression rules of the system, rather than being concerned with the details of implementation. Using various characteristics of Object Oriented (OO) design principles, all aspects of a subsumption behaviour are encapsulated within a behaviour object that is instantiated from the behaviour class. Actions and sensor trigger verifications associated with the behaviours are developed as reusable components that will also facilitate the management of behaviours.

Other parts of the implementation involve the CPP and learning. As discussed previously, the CPP is implemented in a task to output a periodic PSS for all four legs. The learning task is to let the robot learn the OWCI for the CPP to output the phase signal. RL is employed in this thesis as the learning algorithm.
5.2. Implementation of the four phase walking strategy

5.2.1. Implementation framework

5.2.1.1. Philosophy of the framework design

The framework design presented in this thesis is based loosely upon OO principles and has incorporated concepts such as encapsulation and instantiation. At the core of the design are three basic classes, the Trigger component (TC) class, the Action component (AC) class and the Executor component (EC) class. These provide generic templates for different object instances. These in turn are incorporated into a class that supports the creation of a Behaviour object.

The AC class defines the template for implementing various types of actions (e.g. Stand, Forward). It has a property called Actions (operations or instructions to be performed by the component) and two methods: Get Actions and Set Actions. The TC class defines the template for the construction of functions associated with the verification of the sensor conditions (e.g. Forward conditions, Balance conditions). It only has one method, Get Trigger Status, which executes a series of instructions to determine whether certain sensor(s) conditions are satisfied and returns the result (true or false). These would represent the conditions which may lead to a behaviour being triggered, as explained later. This method associated with various objects of the trigger component class needs to be overwritten to address the specific functionality of the objects. The EC class has two properties: Activated Behaviour (for indicating the behaviour that is activated and whose actions are currently being
executed) and Actions (the actions to be carried out) Associated with the EC class are 5 methods: Get Activated Behaviour, Set Activated Behaviour, Set Actions, Execute Actions and Abort Actions. These methods are self-explained by their names.

A pool of primitive elements associated with the TC (e.g. Forward trigger component) and AC (e.g. Balance action component) can be developed. Various combinations of these TCs and ACs are used to develop different types of behaviours, thus supporting the reusability concept.

The main class of this implementation framework, the Behaviour class, is composed of an AC, a TC and an EC (shared by all behaviour objects within a single SA; see explanation later). An additional property, known as the Suppression Mask (SM), is defined to indicate what other behaviours can suppress a specific behaviour (Francis, 1993) or what other behaviours this specific behaviour can suppress. Methods of the Behaviour class include those inherited from AC, TC and EC, and four additional methods, namely: Get Suppression Mask, Set Suppression Mask, Start behaviour and Stop behaviour, which are also self-explained by their names. Various behaviour objects (essentially behaviours) form a group of behaviour entities that are designed to operate together for a particular application. For different applications, there would be different groups of behaviours. For example, considering the architecture associated with a robotic car whose target goal is to navigate to a certain location, the behaviour group might consist of: align, forward, turn, avoid, backward, approach and escape.
In this way, different behaviour objects can be developed by just replacing different combinations of the AC, TC and EC and setting values of properties (e.g. SM) of behaviour objects. This approach facilitates the construction, management, modification and usage of complex behaviour objects as they can be composed using individual components. For instance, the *Turn* behaviour in the example described in the above paragraph, could be composed of a TC to send out a turning signal when the robot car is about to hit an obstacle, an AC to give motion instructions to steer away from the obstacle and an EC to physically carry out the steering actions.

The top-level class of the framework is the *Subsumption* class, which is composed of some behaviour objects and a single method, *Start Subsumption Architecture*. This method initially sets the data of *Activated Behaviour* of the EC of all behaviours to *NULL* which means that there is no active behaviour and thus no actions are currently executing. The method then starts all the behaviours by calling the *Start Behaviour* method of each behaviour object. For any subsumption system there is at least one *Subsumption* object, which is an instance of the *Subsumption* class. A single *Subsumption* object is essentially an implementation of a SA. If a system has more than one *Subsumption* object, it is referred to as a parallel Subsumption Architecture. For example, for a walking robot, there may be one SA for each leg, with each executing its own actions (as is the case for the robotic system discussed in this thesis).

If the sensor conditions associated with a behaviour are satisfied, the behaviour is referred to as being *triggered*. In addition, if it has the highest priority among all triggered behaviours, it is referred to as being *activated*. All the behaviour objects
within a *Subsumption* object run in parallel. When a behaviour becomes activated, it does not directly execute the actions of the *AC*. Instead the behaviour copies the actions of its *AC* into the *EC* for execution. If at any time another behaviour is activated, it will similarly put its own actions into the *EC* for execution, thus overwriting and automatically terminating any current actions. As an aside, if a particular behaviour is re-activated before its actions are completed, this activation is ignored. This is not unreasonable for real robots. For example, while the actions associated with a *Balance* reflex are occurring, the sensor associated with triggering this behaviour could repeatedly trigger the behaviour. It takes a finite amount of time for any action to generate the desired result, so that the sensor no longer triggers that specific behaviour.

The proposed framework allows new *Action* components and *Trigger* components to be created easily as these will inherit the basics from their associated ancestor. New behaviours can also be developed quickly as they will inherit the basics from the *Behaviour* class and only new combinations of the existing or newly developed *AC*, *TC* and *EC*, as well as other properties, need to be defined such that the goal of the specific behaviour is addressed. The framework is flexible in terms of reuse of existing *Action*, *Trigger* and *Executor* component objects. Overall, the design framework consists mainly of class definitions and object instantiations. Class is the abstract template for associated concrete objects while objects are specific instances of the corresponding class. Figure 5.1 summarises some of the concepts discussed in the above design philosophy.
5.2.2.3. Step By Step Implementation

This section provides a step-by-step implementation framework that has been developed during the investigation for the implementation for the proposed four-phase walking strategy. Specifically, it includes the definition of Component,
Behaviour and Subsumption classes and instantiations of corresponding objects.

**Step 1: Define the Component Classes and create specific components**

Component classes include the AC, TC and EC. All are defined as templates for creating specific component objects (refer to Figure 5.1). For the AC, it includes: *Actions* (data), *get actions* (method) and *set actions* (method). For the TC, only one method, *get trigger status*, is defined for getting the component’s associated trigger status. It is defined as an *interface* which means that it must be detailed for specific sensor trigger requirements when creating its instantiations. As for the EC, it includes: *Actions* (data), *Activated Behaviour* (data), *set actions* (method), *execute actions* (method), *abort actions* (method), *get activated behaviour* (method) and *set activated behaviour* (method).

After these component classes have been defined, specific components are created through instantiations from their respective component class. These components include ACs, TCs and ECs. Associated data values (e.g. *actions*), if required, should be supplied as parameters during the instantiation process.

**Step 2: Define the Behaviour Class and create specific behaviour objects**

The definition of the *Behaviour* class includes components (*AC, TC and EC*), *Suppression Mask* (data) and various methods used to access data. The data of the *Suppression Mask* is a variable used to implement the suppression rules. Francis (1993) defines the *suppression mask* as a 16-bit integer that indicates all behaviours which can suppress this behaviour. Methods of the *Behaviour* class include: *get suppression mask*, *set suppression mask*, *start behaviour* and *stop behaviour*. The
The `start behaviour` method is the main mechanism in implementing the suppression function of behaviours. It consists of four parts, as shown in figure 5.2, which are contained within an infinite loop:

1. Check the data `Activated Behaviour` of its EC to determine whether the behaviour has already been activated. If the value of this data refers to itself, it represents the fact that the behaviour has already been activated; otherwise, the behaviour is not being activated.

2. If the behaviour is not the current activated behaviour, determine whether this behaviour can suppress the current activated behaviour via its data: `Suppression Mask` and `Activated Behaviour`.

3. If the answer to step 2 is yes, execute the behaviour’s `Trigger` component by calling the component’s `get trigger status` method to see whether its sensor information is satisfied.

4. If the answer to step 3 is yes, suppress the current activated behaviour by using its EC’s method, `set activated behaviour`, to reset the value of the `Activated Behaviour` to its ID and calls its components' `Get Actions` (AC) and `Set Actions` (EC) methods to execute its actions via its `Executor` component.

This processing is in control of when and how to activate a behaviour, how to suppress other behaviours and how to execute a behaviour's actions after it has been activated.
After the behaviour class has been defined, specific behaviour objects are created.
through instantiations from the behaviour class. Data values of *Suppression Mask*, *Action* component, *Trigger* component and *Executor* component must be supplied during the instantiation process.

**Step 3: Define Subsumption Class and create specific subsumption objects**

This step consists of a collection of all its behaviours objects (that is, all the behaviours defined for a Subsumption Architecture) and a method to start the Subsumption Architecture. As mentioned previously, the EC was shared by all the behaviour objects within the same *Subsumption* object. If an *Action* component of a behaviour has been successfully executed by the *EC*, the Executor component will clear the value of the data of the *Activated Behaviour*. This means that no behaviour is currently activated within this Subsumption architecture, so that now any behaviour, if triggered, could be activated. If there is no activated behaviour within a subsumption object, there are no actions to be performed by the EC.

After the subsumption class has been defined, specific subsumption objects are created through instantiations from the *Subsumption* class. In the instantiation process, collections of all its behaviour objects need to be passed as parameters. If there is more than one Subsumption Architecture in the system, multiple subsumption objects instantiated from the *Subsumption* class need to be created.

**5.2.3. Applying the Implementation Framework to the Robot to Implement the Proposed Four-phase Walking Strategy**

Using the methodology just described in section 5.2.2, a step by step implementation
of the four-phase walking strategy for the experimental robot is given below. The implementation environment and language specifically employed in this thesis was SwiftX 2.5 (Swiftx Ref., 2000), which makes use of the *Forth* programming language. This environment and programming language are particularly suited to embedded systems, and includes significant support for multitasking.

The implementation of the CPP and learning is not included in the implementation framework and is described in detail in section 5.3. The CPP is implemented via an independent task (thread), running in parallel with all the behaviours. Its status (fired or not fired) and output signals (i.e. PSS) are stored in public variables which can be accessed by various trigger components. They function like "virtual sensor information" that can result in the activation of trigger components. This in turn affects the activation of associated behaviour objects.

**Step 1: Define component class: AC, TC and EC and create specific components.**

For definitions of AC, TC and EC classes, refer to step 1 of the implementation framework. The ACs created in the implementation of the proposed four-phase walking strategy are listed in table 5.1. Details with regards to aspects of the motions of specific behaviours have been discussed in chapter 3.
Stand A1 Lock a leg.
Forward A2 Move a leg from position $a$ to position $d$.
Backward A3 Depending on current walking phase, move a leg from position $a$ to position $b$ (phase 3), or from position $b$ to position $c$ (phase 2), or from position $c$ to position $d$ (phase 1).
Balance A4 Extend a leg if the body corner associated with the leg is lower than horizontal level (or relevant part of balance sensor is triggered by momentum) or contract a leg if the body corner associated with the leg is higher than horizontal level (or relevant part of balance sensor is triggered by momentum).
Leg Down A5 Put a leg down on the ground.
Protect A6 Reset robot to predefined “safe” position.

<table>
<thead>
<tr>
<th>AC</th>
<th>ID</th>
<th>Action Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stand</td>
<td>A1</td>
<td>Lock a leg.</td>
</tr>
<tr>
<td>Forward</td>
<td>A2</td>
<td>Move a leg from position $a$ to position $d$.</td>
</tr>
<tr>
<td>Backward</td>
<td>A3</td>
<td>Depending on current walking phase, move a leg from position $a$ to position $b$ (phase 3), or from position $b$ to position $c$ (phase 2), or from position $c$ to position $d$ (phase 1).</td>
</tr>
<tr>
<td>Balance</td>
<td>A4</td>
<td>Extend a leg if the body corner associated with the leg is lower than horizontal level (or relevant part of balance sensor is triggered by momentum) or contract a leg if the body corner associated with the leg is higher than horizontal level (or relevant part of balance sensor is triggered by momentum).</td>
</tr>
<tr>
<td>Leg Down</td>
<td>A5</td>
<td>Put a leg down on the ground.</td>
</tr>
<tr>
<td>Protect</td>
<td>A6</td>
<td>Reset robot to predefined “safe” position.</td>
</tr>
</tbody>
</table>

Table 5.1: List of ACs

TCs created in this implementation are listed in table 5.2. Their specific trigger conditions are illustrated in the column of "Trigger conditions".

<table>
<thead>
<tr>
<th>TC</th>
<th>ID</th>
<th>Trigger Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stand</td>
<td>T1</td>
<td>Always True</td>
</tr>
<tr>
<td>Forward</td>
<td>T2</td>
<td>A leg is at phase 0 and CPP has fired</td>
</tr>
<tr>
<td>Backward</td>
<td>T3</td>
<td>A leg is at phase 1, or 2, or 3 and CPP has fired</td>
</tr>
<tr>
<td>Balance</td>
<td>T4</td>
<td>Body loses balance (i.e. Balance sensor was activated) and a leg is on the ground</td>
</tr>
<tr>
<td>Leg Down</td>
<td>T5</td>
<td>Body loses balance (i.e. Balance sensor was activated) and a leg is not on the ground</td>
</tr>
<tr>
<td>Protect</td>
<td>T6</td>
<td>Robot loses control of itself (i.e. The robot could not regain its balance no matter what other behaviour was activated)</td>
</tr>
</tbody>
</table>

Table 5.2: List of TCs

The ECs created in this implementation are listed in table 5.3. Their specific functionalities are illustrated in the column of "Component Function".
<table>
<thead>
<tr>
<th>EC</th>
<th>ID</th>
<th>Component Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executor 1</td>
<td>E1</td>
<td>Execute actions for LF Leg</td>
</tr>
<tr>
<td>Executor 2</td>
<td>E2</td>
<td>Execute actions for RF Leg</td>
</tr>
<tr>
<td>Executor 3</td>
<td>E3</td>
<td>Execute actions for LB Leg</td>
</tr>
<tr>
<td>Executor 3</td>
<td>E4</td>
<td>Execute actions for RB Leg</td>
</tr>
</tbody>
</table>

*Table 5.3: List of ECs*

For each behaviour there is one Executor component object. However, the Executor component associated with all behaviour objects within a single Subsumption object, refer to one physical entity. In our example there are four Subsumption objects, one for each of the four legs. Thus there are four Executor component “entities” (i.e. E1, E2, E3 and E4). These execute the actions supplied by the corresponding activated behaviour. As a result, actions for all four legs can occur concurrently.

**Step 2: Define behaviour class and create specific behaviour objects**

For the definition of the behaviour class, please refer to step 2 of the implementation framework. Six types of behaviour objects: Stand, Forward, Backward, Balance, Leg Down and Protect are created for each of the four legs. In total there are 24 behaviour objects of six types created in this implementation, as illustrated in table 5.4. In the table, n stands for which leg a behaviour belongs to. In this thesis, “n=1” represents LF leg, “n=2” represents RF leg, “n=3” represents LB leg, and “n=4” represents RB leg.
<table>
<thead>
<tr>
<th>Behaviour object</th>
<th>ID</th>
<th>AC</th>
<th>TC</th>
<th>EC</th>
<th>Behaviour Sub-goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stand (n)</td>
<td>1</td>
<td>A1</td>
<td>T1</td>
<td>E (n)</td>
<td>To lock a leg at the current position</td>
</tr>
<tr>
<td>Backward (n)</td>
<td>2</td>
<td>A2</td>
<td>T2</td>
<td>E (n)</td>
<td>To carry out work of other phases (1,2,3) of a leg to push body forward</td>
</tr>
<tr>
<td>Forward (n)</td>
<td>4</td>
<td>A3</td>
<td>T3</td>
<td>E (n)</td>
<td>To carry out the phase 0 work of a leg to swing a leg forward</td>
</tr>
<tr>
<td>Balance (n)</td>
<td>8</td>
<td>A4</td>
<td>T4</td>
<td>E (n)</td>
<td>To balance the body to avoid falling over</td>
</tr>
<tr>
<td>Leg Down (n)</td>
<td>16</td>
<td>A5</td>
<td>T5</td>
<td>E (n)</td>
<td>To put a leg down to make contact with the ground</td>
</tr>
<tr>
<td>Protect (n)</td>
<td>32</td>
<td>A6</td>
<td>T6</td>
<td>E (n)</td>
<td>To set the legs and body of the robot to “safe” positions</td>
</tr>
</tbody>
</table>

Table 5.4: List of behaviour objects (n = 1, 2, 3, 4)

Here, a Behaviour ID is defined as a power of 2 (i.e. a single bit). As such, multiple behaviours can be referred to based upon which bits are set in a bit pattern. This is used in the data of Behaviour Suppression Mask (BSM) of the behaviour objects, as shown in table 5.5, to deal with behaviour suppression relationship. This thesis employs a 16 bit variable which allowed up to 16 behaviours to be defined. This was more than adequate for the implementation discussed here. However, if there are too many behaviours to be represented by a single bit of an appropriately sized variable, it is a simple matter to modify the implementation. For example, the behaviours can be represented by a set of single numbers and the suppression information would then be implemented as a list of numbers associated with behaviours to be suppressed.
### Table 5.5: Behaviour BSM list

<table>
<thead>
<tr>
<th>Behaviour</th>
<th>Suppress behaviours</th>
<th>BSM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stand</td>
<td>(none)</td>
<td>0</td>
</tr>
<tr>
<td>Backward</td>
<td>Stand</td>
<td>1</td>
</tr>
<tr>
<td>Forward</td>
<td>Stand, Backward</td>
<td>1+2=3</td>
</tr>
<tr>
<td>Balance</td>
<td>Stand, Forward, Backward</td>
<td>1+2+4=7</td>
</tr>
<tr>
<td>LegDown</td>
<td>Stand, Forward, Backward, Balance</td>
<td>1+2+4+8=15</td>
</tr>
<tr>
<td>Protect</td>
<td>Stand, Forward, Backward, Balance, LegDown</td>
<td>1+2+4+8+16=31</td>
</tr>
</tbody>
</table>

### Step 3: Define Subsumption class and create specific Subsumption objects

For the definition of the subsumption class, please refer to step 3 of the implementation framework. Subsumption objects created in this implementation are listed in table 5.6. As discussed previously, there will be one SA for each leg. Therefore in total, there are four Subsumption objects in this system.

<table>
<thead>
<tr>
<th>SA objects</th>
<th>ID</th>
<th>Object Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA Object 1</td>
<td>S1</td>
<td>Subsumption object created for SA 1 – LF Leg</td>
</tr>
<tr>
<td>SA Object 2</td>
<td>S2</td>
<td>Subsumption object created for SA 2 – RF Leg</td>
</tr>
<tr>
<td>SA Object 3</td>
<td>S3</td>
<td>Subsumption object created for SA 3 – LB Leg</td>
</tr>
<tr>
<td>SA Object 4</td>
<td>S4</td>
<td>Subsumption object created for SA 4 – RB Leg</td>
</tr>
</tbody>
</table>

### Table 5.6: List of Subsumption objects

Aspects of S1 are listed in table 5.7. Other Subsumption objects are similar but composed of different instances of AC, TC and EC respectively. To start a SA, it is necessary to execute the `start Subsumption Architecture` method of its Subsumption object. As all required operations have been defined and encapsulated inside various classes, all specific objects instantiated from these classes will automatically acquire the necessary operating knowledge to perform correctly.
As an example of this implementation, a typical scenario involving the left front leg of the four-phase walking strategy is described in the following section, assuming that the leg is on the ground but not extended (see left front leg of Figure 4.1, at position a) at the start. When the CPP has fired, as a result of this initial condition, the behaviours Stand and Forward are both triggered. However, based on the suppression rules, the Forward behaviour will suppress the Stand behaviour. Consequently, the Forward behaviour becomes the activated behaviour and the actions associated with this behaviour are executed. The left front leg pushes downwards towards the ground first, to create movement for compensation of the sway action following up. At the same time, cylinder δ (Figure 3.3) associated with the leg is extended to move the centre of gravity of the robot to the opposite side of the left front leg. Next the left front leg is then raised and swung forward and subsequently placed down on the ground. At the same time, the Backward behaviour of the other three legs will be activated, each with different leg phases. During this process, either the Balance or LegDown or Protect behaviours may be activated depending upon certain sensor information. For example, the LegDown behaviour will be triggered if the robot loses its balance when swinging the leg forward. If any
one of these behaviours is triggered, it will suppress the Forward behaviour and takes over as the activated behaviour according to the predefined priorities of the behaviours. Its actions will then be carried out in an attempt to prevent the robot from falling down. When the robot’s balance is restored, the Forward behaviour is activated again. These series of behaviours may occur repeatedly until the stepping action is completed. When this occurs (all actions of the Forward behaviour have been completed), the Forward behaviour is no longer the activated behaviour and the Stand behaviour becomes activated, providing that the CPP has not fired again at this point of time. The Stand behaviour locks all the limb positions of the leg (however, the leg may still be moving depending on the momentum of the robot). While actions of the Stand behaviour are being executed, any other behaviours, if triggered, will become activated, as the Stand behaviour has the lowest priority. For example, during the process of actions of the Stand behaviour being executed, the CPP may fire again resulting in the Backward behaviour of the same leg (the CPP will generate the phase signal for the Backward behaviour to be activated after the Forward behaviour has been carried out) to be activated. Thus repeating a similar process. The above discussion scenario is for the left front leg only. A similar process is actually happening simultaneously in all the other three legs. The interactions of all the behaviours of the four legs generates a combination of preset and emergent walking behaviour that enables the robot to walk.

5.3. Implementation of the CPP and learning

In the implementation of the CPP, the main issue here was to determine the value of
the OWCI for the CPP task. As mentioned previously, both empirical (determined by trial & error through experience) and learning (via RL) methods were employed. In the empirical method, an empirical value of 500 ms was used as the walking cycle interval, which resulted in a satisfactory overall performance of the robot’s walking. In the learning method, reinforcement learning was employed to allow the CPP to learn the value of OWCI for various terrain conditions.

The learning task involving the CPP was to learn the value of the OWCI. A learning episode starts immediately after each PSS was generated by the CPP and passed to the four respective SAs. After the learning episode has been started, it will select an action based on its selection policy and execute the action (i.e. creation of a delay for the CPP to generate the next PSS). Once the action has finished, it will get a reward or punishment by evaluating all the behaviours of the four legs. Based on this reward or punishment, it will update its Q values (Sutton and Barto, 1998) and start another round of the learning process. Updating of the Q values will improve its selection policy. This sequence goes on until the learning process was terminated. By that time, the selection policy is expected to converge to the optimal policy, whose action is the OWCI.

In order to implement this RL algorithm, the state, action and reinforcement function has to be properly defined. The robot has two states: the ideal walking (IW) state and non-ideal walking (NIW) state. The IW state is defined as that state for which the robot is in its ideal walking condition (i.e. only the Forward and/or Backward behaviours are activated). The NIW state is defined as that state for which the robot is not in its ideal walking condition (i.e. one or more assisting behaviour(s) were
activated).

The individual actions for the learning algorithm are defined as the individual values of the walking cycle interval required as a part of the walking cycle of the robot. Therefore, the value of the OWCI is the action of the IW state corresponding to the optimal policy. As previously mentioned a range of 450ms to 600ms was chosen for the walking cycle interval. Given the action number in the learning space is \( n \), \( \text{Action}(i) \) is defined as \( 450 + \frac{i \times (600 - 450)}{n} \) (ms) = \( 450 + \frac{i \times 150}{n} \) (ms). The action number was set to 10 in our experiments (i.e. 15ms interval), due to the physical limitations of the on-board computer. There is no inherent limitation in the implemented algorithm itself, and thus the size of the action space could be increased with a higher speed computer with more memory. The total learning space (Q number) is calculated using equation 1.

\[
Q_{\text{number}} = \sum_{i=1}^{n} a_i \quad (n : \text{number of states}; \; a_i : \text{number of actions at state } i) \quad (1)
\]

In this instance, the number of states is 2 and the number of actions is 10 for each state. Therefore, the learning space for the robot is: \( 2 \times 10 = 20 \).

The reinforcement function is defined in terms of the following rules:

1. If an action is taken in a state and results in the robot reaching or remaining in the IW state, that action is given a positive reward (any integer number, e.g. 1)

2. If an action is taken in a state and results in the robot reaching or remaining in the NIW state, that action is given a negative reward (e.g. -1).
5.4. Chapter Summary

This chapter presents the implementation of the four-phase walking strategy proposed in Chapter 4. To facilitate reusability, an implementation framework for SA is given in the first part of this chapter. The implementation framework was designed using various characteristics of OO design principles. Behaviours were implemented as specific objects instantiated from the behaviour class. The second part of this chapter applied the implementation framework to the physical implementation of the robot incorporating the proposed four-phase walking strategy. This involved the creation of various AC, TC and EC components, behaviour objects and subsumption objects via instantiations of their associated classes.

This chapter also presented the design and implementation of the CPP and a RL algorithm for the robot to learn the value of the OWCI. The one step Sarsa learning algorithm was employed in a real-time implementation for the robot’s walking.
Chapter 6: Experiment Results and Discussion

6.1. Introduction

Real-time walking experiments with the robot were carried out to test and investigate the proposed walking strategy. As outlined below, four different types of experimental “terrain” conditions were employed to examine the robustness of the architecture.

- Flat ground -- a horizontal laboratory floor covered with rough carpet, as shown in figure 6.1-a;

- Incline and decline -- a tilted rectangular hardwood surface covered with rough carpet, as shown in figure 6.1-b and 6.1-c respectively. In this example, the incline had a 3.5 degree tilt and the decline had a 5.0 degree tilt, each relative to the horizontal;

- Uneven ground -- a horizontal laboratory floor covered with rough carpet, with books and irregular articles randomly placed under the carpet, as shown in figure 6.1-d. The maximum height of these objects is 31 mm in this experimental terrain.
Experimental results were recorded, analysed and presented using three different methods: Behaviour Plots, Behaviour Autocorrelation and Behaviour Cross-correlation. A Behaviour Plot is used to demonstrate the behaviour and phase transitions of the robot’s walking. Autocorrelation is used to demonstrate the existence of periodic phase patterns that occur in the robot’s walking. Cross-correlation is used to demonstrate the coordination among all legs of the robot during locomotion and the phase differences between the legs. This is important to the robot as coordination must exist between the robot’s legs in order for it to walk.
To reduce the amount of onboard memory required for recording experimental data, behaviour values were encoded in the manner shown in Table 6.1.

<table>
<thead>
<tr>
<th>Behaviours (and phases)</th>
<th>Encoded Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backward @ phase 1</td>
<td>1</td>
</tr>
<tr>
<td>Backward @ phase 2</td>
<td>2</td>
</tr>
<tr>
<td>Backward @ phase 3</td>
<td>3</td>
</tr>
<tr>
<td>Forward @ phase 0</td>
<td>4</td>
</tr>
<tr>
<td>Stand</td>
<td>5</td>
</tr>
<tr>
<td>Balance</td>
<td>6</td>
</tr>
<tr>
<td>LegDown</td>
<td>7</td>
</tr>
<tr>
<td>Protect</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 6.1: Encoding Scheme

As discussed previously, the CPP was activated to output the PSS for all four legs at a certain interval (i.e., the Walking Cycle Interval - WCI). Both empirical values of the WCI and learning methods were used to determine the value of the Optimal WCI (OWCI). In the empirical method, the value of 500 ms was employed. This value was chosen based upon experimental experience in which the robot exhibited a “stable” walking behaviour on “all terrains”. In this thesis, when the term “all terrains” is used, it only means all the experimental terrains that have been employed in this thesis whereby the conditions (e.g., the maximum steep degree of an incline) are within the definition and limitation of the experimental environments of this thesis. This empirical value was actually determined from experiments involving the robot’s walking on a flat ground. This same value was then used in the robot to walk on other terrains. The obvious question to ask is that, because the value of the OWCI is obtained while observing the robot’s walking on flat ground, is this value still correct to be applied to other terrains employed in this thesis? This question is
answered by the learning method whereby the results show that this value based on the robot’s walking on flat ground is also a suitable value for other terrains (see learning results presented in section 6.3). The learning algorithm is one step Sarsa learning (Sutton and Barto, 1998). Real-time learning experimental data was recorded and plotted as a learning curve for investigation.

6.2. Experimental results and discussion for the case of an empirical value for the OWCI

6.2.1. Flat ground

For the experiments conducted on flat ground, the robot walked from one end of a laboratory room to the other end (approximately 6 metres). Based on the designed walking strategy, ideally there should be a four-phase walking behaviour, with the robot’s movements continually following the phase order of …0-3-2-1-0… (see figure 4.1), while the robot walks. A “Behaviour Plot” (e.g. Figure 6.2-a) is employed here to investigate this aspect of the robot’s walking, whereby the activated behaviours (encoded value) of the robot are plotted as a function of time. The robot walks at the speed of approximately 6 to 7 metres/minute, depending on the real-time interaction of the robot with the environment as the walking associated with the robot is non-deterministic.
Figure 6.2-a: Behaviour plot of the robot to walk through a flat ground.

Figure 6.2-a shows the behaviour plot of one typical set of recorded data for the four legs (LF, RF, LB, RB) of the robot, with one plot for each leg. Interpretations of the encoded values of this figure clearly show that for the majority of the time, each of the legs are following the "ideal walking cycle". Visually this is easily seen as the repeating “4 step staircase-like” pattern. More precisely, it consists of the following: … Phase 0 Forward (Behaviour value 4 in Figure 6.2-a) - Phase 3 Backward (Behaviour value 3 in Figure 6.2-a) - Phase 2 Backward (Behaviour value 2 in Figure 6.2-a) - Phase 1 Backward (Behaviour value 1 in Figure 6.2-a), followed again by Phase 0 Forward (Behaviour value 4 in Figure 6.2-a) - … . When comparing the ideal walking cycle with Figure 4.1, these plots clearly show that a rhythmic walking phase cycle does indeed exist for each leg. There are a few walking cycles that do not follow the phase order described above and they are known as non-ideal
walking cycles. In these instances, the Stand and/or Balance behaviours were activated. That is, some of the legs at certain times were not undertaking the ideal walking cycle. This is due to some undesirable real-time situations such as the body of the robot being tilted beyond an acceptable range. The results also show that the robot has the capability to recover from those undesirable situations, as indicated in the plots where the walking cycle is restored to the ideal walking cycle immediately after these non-ideal walking cycles. Analysis of these results shows that the behaviours Forward and Backward are employed by the robot most of the time, with a few exceptions when the Stand and Balance behaviour is triggered.

The length of the walking cycle interval is the same for both ideal and non-ideal walking cycles as this is a fixed parameter within the CPP. The plots also show that there is a phase difference of one unit between the LF and RB, RB and RF, RF and LB legs. These results demonstrate that the proposed walking strategy is working.

Autocorrelation is employed to investigate the existence of a rhythmic walking cycle of the robot. Although already quite apparent in the previous plots, it is usual to use autocorrelation to demonstrate periodicity, particularly with signals which are not so obviously periodic. A periodic peak value in autocorrelation plot indicates a periodic pattern existed in the source signal. The designed walking strategy implies that a four-phase walking pattern is apparent in the robot’s movements. The autocorrelation (Box and Jenkins, 1976) function is defined as: Given measurements $Y_1, Y_2, ..., Y_N$ at time $X_1, X_2, ..., X_N$, the lag $k$ autocorrelation function is defined as:
Autocorrelation results for the four legs are shown in Figure 6.2-b. As can be seen from the plot associated with each leg, a periodic peak occurs every 4 steps. For instance, in the plot of the LF leg, a peak value has been shown in step 0, 4, 8, 12, 16, 20, ..., with the periodic interval of 4 units of steps. Each step in the plots (i.e. time increment), represents a different phase for a leg of the robot. This demonstrates that the robot is carrying out a cyclic four-phase walking pattern.

In order to investigate the coordination and phase delay between legs, cross-correlation of the data between different legs is employed. A periodic peak value in a
cross-correlation plot indicates periodic coordination existed in the two source signals and the number of steps for the first peak value indicates the lag between the two source signals. It is to be expected from the designed walking strategy that coordination exists in all legs of the robot during locomotion. Specifically, RB leg moves one phase behind LF leg, RF leg moves two phases behind LF leg, LB leg moves three phases behind LF leg, RF leg moves 1 phase behind RB leg, LB leg moves 2 phases behind RB leg and LB leg moves 1 phase behind RF leg. The cross-correlation (Box and Jenkins, 1976) function is defined as: Given two signal measurements $X_1, X_2, ..., X_N$ and $Y_1, Y_2, ..., Y_N$ at time $T_1, T_2, ..., T_N$, the lag $k$ cross-correlation function between $X$ and $Y$ is defined as:

$$
\tau_{XY}(k) = \frac{\sum_{i=1}^{N-k} (X_i - \bar{X}) (Y_{i+k} - \bar{Y})}{\sqrt{[\sum_{i=1}^{N} (X_i - \bar{X})^2] [\sum_{i=1}^{N} (Y_i - \bar{Y})^2]}^{1/2}}
$$
Cross-correlation results are shown in figure 6.2-c. In the figure, leg movement delays reflected by cross-correlation are listed as follows:

- RB leg is one time step behind the LF leg (see LF/RB plot, where the first peak occurs at step 1);
• RF leg is two time steps behind the LF leg (see LF/RF plot, where the first peak occurs at step 2);

• LB leg is three time steps behind the LF leg (see LF/LB plot, where the first peak occurs at step 3);

• RF leg is one time step behind the RB leg (see RB/RF plot, where the first peak occurs at step 1);

• LB leg is two time steps behind the RB leg leg (see RB/LB plot, where the first peak occurs at step 2);

• LB leg is one time step behind the RF leg leg (see RF/LB plot, where the first peak occurs at step 1);

Each step in the plot (i.e. time increment) represents a phase delay in the plot. This verifies the phase delays of different legs implemented by the designed walking strategy.

In observing the robot’s locomotion, some slippage did occur as the robot walked. The smoother the ground and the faster the robot walked, the more often slippage occurred. Slippage did not greatly impair the robot’s walking, as the robot tried to use dynamic balance to compensate for walking disturbances (e.g. slippage and tipping). In a very few extreme circumstances, slippage caused the robot to lose its balance, where the Protect behaviour kicked in to prevent the robot from tipping over. As the robot walked its speed was not uniform. This is easily understood as the robot’s walking is non-deterministic. The walking speed was affected by the real-time interactions of the robot with the environment, in which the portion of the activation of dynamic and static balance and percentage of activation of assisting
behaviours will have the most important effect. With regards to the direction of locomotion, the robot did show constancy of moving forward in the long term. However at various times, there were slight disturbances in the walking, which were also caused by the non-deterministic characteristics of the system. For example, when a leg was carrying out a Forward behaviour, it might or might not be able to fully complete this behaviour as the Balance behaviour might kick in at any time.

6.2.2. Experiments on Other terrains: Incline, Decline and Uneven Ground

Experiments to test the robot walking in other terrains, including incline, decline and uneven ground, were also carried out. Experimental data show similar results to that of the robot walking on a flat ground, although with some, perhaps expected, differences in the quality of performances. Figure 6.3-a to 6.3-c show typical behaviour plots of the robot's walking on incline, decline and uneven ground respectively. Experimental terrain conditions were as follows: the incline had a 3.5 degree tilt relative to the horizontal; the decline had a 5.0 degree tilt relative to the horizontal; and obstacles had a maximum height of 31 mm. The walking cycle interval in all cases was set to the empirical value of 500 ms. These behaviour plots clearly show rhythmic data signal cycles (..→4→3→2→1→…), which represent the ideal walking cycle of Forward (phase 0) → Backward (phase 3) → Backward (phase 2) → Backward (phase 1), does exist in the robot’s walking on different terrains.

In figure 6.3-a where the robot was walking down a decline, the behaviour plots
show that there are more Balance behaviours (and some Leg Down behaviours) being activated, compared to the walking results of the robot on flat ground. It is not difficult to appreciate this difference in the robot's walking quality, considering the fact that the robot can more easily lose its balance when walking on a decline as compared with that on flat ground. This issue is most easily experienced by anyone trying to walk down a hill. It is more difficult to keep balance under these circumstances than walking on flat ground. With regards to the walking speed, the robot walks slightly faster (approximately 10 metres/minute) than when it walks on flat ground. Although there are more balance behaviours being triggered, observation of the robot, shows a larger step stride (this is most probably due to the effect of gravity when walking down a decline than walking on flat ground, resulting in a faster walking speed.

Figure 6.3-a: Behaviour plot of the robot walking down a decline
Figure 6.3-b shows the behaviour plots for the robot walking up an incline, whereby it is interesting to note that there were more Stand behaviours being activated, compared to the results of the robot walking on flat ground and decline. This is not unreasonable when taking account the fact that the robot needs more force against gravity while walking up an incline. In some circumstances, the robot does not have enough strength to adequately perform the walking, resulting in the Stand behaviour being triggered. This also is easily experienced by anyone when climbing up a hill. Frequent stops ("stand") for a break are required, as more energy is required in climbing than walking on flat ground or down a hill. In experimental observation, the robot walks much slower (approximately 3 metres/minute) than when it walks on a flat ground, due to the fact of more Stand behaviours being triggered and the smaller step stride.

*Figure 6.3-b: Behaviour plot of the robot walking up an incline*
Figure 6.3-c: Behaviour plot of the robot walking through an uneven ground

Figure 6.3-c show the behaviour plots of the robot's walking over uneven ground. It can be seen from the figure that the robot's walking quality is much reduced when compared with walking on the other three terrains. As can be seen in the figure, there are many non-ideal walking cycles (phase patterns that are different to what is shown by the marked ideal walking cycle in the plot). It can be seen from the figure that more *Leg Down* behaviours were activated when compared with the results of the robot walking on the other terrains. This is because the uneven ground makes it even easier for the robot to lose balance when walking on this type of terrain, resulting in the activation of more assisting behaviours, such as *Balance* and *Leg Down* behaviours. Most people have experienced this, whereby when someone is walking
on uneven ground such as that which has lots of holes and obstacles, it is less
comfortable and requires more effort to walk compared with walking on flat
“uncluttered” ground. From experimental observation, the robot walks at a speed
(approximately 4 to 5 metres/minute) somewhere between the speed of walking on
flat ground and the speed of walking up an incline.

With regards to the issues of slippage, uniformity of speed and the constancy of
walking direction, they had similar results as was observed for the robot walking on
flat ground.

One possible measurement for determining the walking performance of the robot in
different terrains is to use the percentage of times that the robot is employing ideal
walking cycles relative to the total time of the experiment. That is 100% walking
quality is achieved if no Stand, Balance, Leg Down or Protect behaviours are
activated for all four legs. (i.e. ideal walking). The percentage of activation of ideal
walking behaviours is calculated using the formula below:

\[
P(\%) = \frac{\sum_{all\ legs} (Ideal\ walking\ behaviours)}{\sum_{all\ legs} (all\ behaviours)}
\]

\[
= \frac{\sum_{all\ legs} (Forward\ and\ Backward\ Behaviours)}{\sum_{all\ legs} (all\ behaviours)}
\]

A lower value means a poorer quality of walking by the robot. Table 6.2 gives the
“walking quality” percentages of the robot for 100 walking cycles, whereby
snapshots of this data were shown in Figures 6.3 (a, b, c). The table illustrates that
the robot’s walking quality decreased in the order of flat ground, decline, incline and
uneven ground.

<table>
<thead>
<tr>
<th>Terrain</th>
<th>Percentage of ideal walking cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flat ground</td>
<td>92%</td>
</tr>
<tr>
<td>Decline</td>
<td>83%</td>
</tr>
<tr>
<td>Incline</td>
<td>79%</td>
</tr>
<tr>
<td>Uneven ground</td>
<td>71%</td>
</tr>
</tbody>
</table>

Table 6.2: Percentage of activation of ideal walking cycle in different terrains

Additional experiments were carried out to determine that the robot was able to walk up a maximum incline of 5.1 degrees. The robot needs more powerful pneumatic cylinders to generate a greater force against gravity to be able to walk up steeper inclines. The robot was able to walk down a decline of up to 8.3 degrees, being limited by the current implementation of the balance sensor. As for uneven ground, the robot could walk through terrain with obstacles of a maximum height of 65 mm, with 12 obstacles randomly placed on a 3.5 meter wide and 5.4 metre long floor. The current mechanical design of the robot’s legs prevents the robot from raising its legs any higher in order to overcome obstacles with a height greater than the 65mm given above.

From the above investigation and discussion of the experimental results, it can be seen that the robot employs the proposed four-phase walking strategy and exhibits a rhythmical four-phase walking cycle. The walking cycle revealed by experimental results is: Phase 0 (Forward) → Phase 3 (Backward) → Phase2 (Backward) → Phase 1 (Backward)…. This result verifies that the four-phase walking strategy
design has been successfully implemented. Experimental results also show that the robot can carry out its walking behaviours in various kinds of terrain conditions, including flat ground, incline, decline and uneven ground.

6.3. Experimental results and discussion of the robot’s learning

6.3.1. Learning results for the robot walking on flat ground

Real-time experiments were carried out to test and investigate the learning algorithm to learn the value of the OWCI for different terrains. The one step Sarsa learning (Sutton and Barto, 1998) was employed as the algorithm for the robot. In the experiments that have been carried out, the value of the learning rate $\alpha$ is set to a small value so that the robot can carry out sufficient learning experiences to avoid the problem of local maximum. The discount factor $\gamma$ (see equation 1 and discussion of section 2.2.3 of chapter 2) is also given a small value because the robot needs to give more credit for the immediate feedback when selecting its next action, otherwise the robot will easily fall down as the experimental platform is a relatively unstable system. The robot needs to quickly react to the immediate valuation of the result of its actions to make correction to its policy in the real-time experiments. The learning rate $\alpha$ was set to 0.05 and the discount factor was set to 0.1. The learning time step was set to 2000 so that the robot had a reasonable number of learning steps for iterations. The exploration factor $\varepsilon$ (see equation 1 and discussion of section 2.2.3 of chapter 2) was given a relatively large value (0.3) so that the robot had more
opportunity to explore the whole learning space to avoid any bias and/or local maximum. The positive reward was set to 1 and the negative reward (punishment) was set to -1. All Q values were set to 0 at the beginning of the learning process. Due to the limitation of the onboard memory of the robot and the fact that the IW state is the state of interest, only Q values of this state were recorded on board during the learning process and saved to a file for evaluation at the end of the learning process. Learning curves of Q values (IW state only) vs learning step were employed to investigate the learning results of the robot. All Q values are plotted in one diagram for comparison. The optimal policy is defined as $P(s) = \text{Max} (Q(s,a))$. A typical learning curve corresponding to the robot walking on flat ground is shown in figure 6.4-a. The learning curve of the optimal policy is plotted using a black solid line (in the top of the figure). Other learning curves of sub-optimal policies are plotted using a grey-dotted line (in the middle of the figure). As mentioned previously, the robot only recorded the learning data of the IW state. The Q value of each (state, action) pair $(s, a)$ ($s=\text{IW}, a=1 \text{ to } 10$) is recorded as an individual data array for the whole learning process. Therefore there are ten data arrays to record learning data, which are plotted as ten curves in the learning curve figures. The optimal curve is the one with the highest values and is plotted using a black solid line and the other curves are plotted using grey-dotted lines.
As shown in the figure, at the beginning of the learning process (the first few dozens of learning steps), learning has not had enough experiences for its policy to be “close” to the optimal policy, reflecting in the figure that all Q values are oscillating around a value of zero. As can be seen by observation of the robot’s walking, this is evident from the jerky motion of the robot in the beginning. The robot continues to explore the environment and exploit what has been learnt in the previous learning episodes. As the learning process progresses, it gradually makes more correct decisions based on its improving policy. The learning curve shows that the robot could actually find its optimal policy in about 200 learning steps (approximately 2 minutes). However, to avoid any bias and local maximum, the learning of the robot continued to proceed for 2000 steps, which took about 20 minutes.

The learning curve (black solid line in figure) shows the oscillation of the optimal
policy even after the robot has achieved its learning goal. This could be explained by the fact that a relatively large value (0.3) was given to the exploration factor $\varepsilon$. Additionally the employed Sarsa learning algorithm was an on-policy algorithm, resulting in the robot constantly exploring the environment even after it has found the optimal policy. The oscillation of the optimal policy also affected by the way the learning experiments were carried out. Due to the limitation of the experimental terrain conditions, these terrains actually do not have enough space (walking distance) for the robot to walk for the whole process of the robot’s learning in all the learning experiments that have been carried out (including terrains other than the flat ground). To solve this problem in the real-time experiments, the robot was periodically pulled back to the starting place when it had walked to the end of the terrain. During the period when the robot was being pulled back, the learning process continued regardless of this distraction (“noise”). When the robot was pulled back, it resulted in a significant amount of jerky motion in response to the pulling action, which contributed to the oscillation of the optimal policy. As discussed previously, the robot walked at a speed of about 6 to 7 metres per minute on flat ground (6 metres long in this experiment). Therefore it took about a minute for the robot to walk from one end to the other end of this experimental terrain, resulting in the robot being pulled back in an interval of about a minute. The learning result shows that the average walking cycle interval is around 500ms. Therefore the robot was pulled back at intervals of about 120 learning steps. The influence of such pulling action did reflect in figure 6.4-a where the learning curve showed oscillations in an interval of round 120 learning steps. The robot (and the learning algorithm) does show the robustness to recover from these abnormal conditions.
The learning curves reveal that action 3 (a time interval for the CPP to output a PSS, see definition in chapter 5) is the optimal action for the IW state. Learning curves of other action \((a=1, 2, 4, 5, 6, 7, 8, 9, 10)\) are shown as grey-dotted lines (in the middle of the figure). As only the optimal curve is of interest to the robot, those sub-optimal curves are not specifically marked for each action. Based on the definition of the action, the optimal policy which the robot has learnt is interpreted as: the value of OWCI for the robot is \(450 + 3 \times 15 \pm \text{half time step} = 495 \pm 7.5 \text{ ms}\). It could be seen from the learning result that the empirical value 500 ms that had been chosen through experimental experience and used in previous experiments fell within the range of the optimal value achieved by learning.

6.3.2. Learning results for the robot walking on other terrains

A set of typical learning curves of the robot’s walking on other terrains are shown in figures 6.4-b, 6.4-c and 6.4-d, for the terrains of decline, incline and uneven ground respectively. Terrain conditions of these learning experiments are the same as those described in section 6.1.
Figure 6.4-b: Learning curve of the robot walking down a decline

Figure 6.4-c: Learning curve of the robot to walk up an incline
Learning curves of the robot’s walking on these other terrains exhibit similar results to that of the robot’s walking on flat ground, revealing that action 3 (see explanation before) is the optimal action for the IW state. At the same time, the robot’s learning in different terrains do show some difference in learning quality, which decreased in the order of flat ground, decline, incline and uneven ground. This is based upon the decrease of the value of the Q value for the optimal policy and the increase of the level of oscillation of the optimal policy for the terrains in the order of flat ground, decline, incline and uneven ground. Figure 6.4-b where the robot walks down a decline shows the closest result to that of the robot walking on flat ground, with slightly less optimal Q value and larger range of oscillation. This is due to the fact that when the robot walks down a decline, its balance sensor was more easily triggered which in turn caused the activation of the behaviours of *Balance*, *Leg Down* or even *Protect*, resulting in a reduced learning quality. Figure 6.4-c where the
robot walks up an incline shows an interesting result, whereby learning around the first 200 steps did not proceed very well. This can be explained by the fact that the robot was not that successful in carrying out its behaviours against the different influences of gravity which exists on the incline. Figure 6.4-d where the robot walks through uneven ground shows the worst learning result in comparison to that of the robot walking on flat ground. This is due to the fact that the uneven ground constantly triggers non-ideal waking behaviours (e.g. Stand, Balance, Leg Down or Protect), which deteriorates the robot’s learning.

These findings are consistent with experiments that have been carried out before using the empirical value for the OWCI. The learning results also show that the robot has the same OWCI for all terrains employed in this thesis. This is an interesting finding and it would appear to be somewhat contradictory to what one may have expected. It is reasonable to expect that there is a different OWCI for different terrains. However the experimental results reveal that for all the experimental terrains where the robot has carried out walking, it has the same OWCI. This may be explained by the fact that the same behaviour actions (i.e. ACs) were implemented in the robot for walking on different terrains. Although a behaviour itself is generic for the robot to walk on different terrains, it may require different actions (i.e. ACs) or action parameters (i.e. the time period for how long a leg is swung for the Forward AC) for different terrains. Another possible explanation for the same OWCI for different terrains is that there is a limitation to each terrain other than the flat ground where the robot could walk on (see section 6.2). The robot may have the same OWCI for different terrains when those terrains are within the
limits as described before, however it may require different OWCI when walking on terrains that are beyond these limits. For example, when the robot walks down a decline which is more than 8.3 degree steep (the current decline limit that the robot can walk down), it may require a smaller value of the OWCI as the robot may need a smaller interval period to take actions to prevent it from tipping over. Similarly, when the robot walks up an incline which is more than 5.1 degree steep (the current incline limit that the robot can walk up), the robot may need a larger value of the OWCI so that it can have enough time to carry out actions against gravity. These issues are left as future work in any follow up investigation.

Effects of using different values for the parameters (i.e. \(\alpha, \gamma, \varepsilon\)) associated with the learning algorithm were also tested and analysed. When the learning rate factor \(\alpha\) is large, the robot learns faster but shows more jerkiness during the learning process compared to a small \(\alpha\) value. From the learning algorithm (equation 2), it can be seen that the larger the learning rate factor \(\alpha\), the more influence the reward (punishment) has in updating the value, resulting in a faster learning speed. However, by doing this, noise is more easily introduced into the update process. With regards to the discount factor \(\gamma\), it is found that the robot falls down more easily with larger values of \(\gamma\). Large values of \(\gamma\) means that the robot gives more credit for the long term influence of the current actions, which may result in a “slow reaction” of the robot in responding to any rapid changes in the environment. As the experimental platform is relatively unstable, the robot in this case, is more vulnerable to falling down. Experiments also showed that the robot has a low “walking quality” with very small values of \(\gamma\). Very small values of \(\gamma\) means that the robot takes little into account of
the long-term influence when it chooses current actions, which may result in the robot being “short-sighted” and therefore reduces usage of the dynamic balance in its walking. In terms of the exploration ratio factor $\varepsilon$ (see equation 1 and discussion of section 2.2.3 of chapter 2), the robot learns faster with larger values. However, its performance during the learning process is not as good as that of a smaller value. Larger values of $\varepsilon$ means more probability is given to exploration. This will provide more opportunities for the robot to try different possible instances, resulting in a faster learning rate. However, by doing this, there will be more oscillations in the learning process as a result of a high exploration rate.

### 6.5. Chapter Summary

This chapter presents the experimental results for the implementation of the proposed four-phase walking strategy using four parallel SAs and a CPP. Both empirical and learning methods were employed to decide the value of the OWCI. Four different types of terrains, including flat ground, incline, decline and uneven ground, were employed to test the flexibility and robustness of the robot’s walking strategy. Experimental results show that the robot employs the proposed walking strategy and it can successfully carry out its walking behaviours under various experimental terrain conditions. Learning experimental results show that the robot can learn the OWCI for different terrains.
Chapter 7: Conclusions and Future Work

This thesis presents the design and implementation of a four legged walking robot that incorporates a biologically inspired four-phase walking strategy implemented using behaviours, which enables the robot to successfully carry out a combination of preset and emergent walking behaviours in different terrains. Four parallel SAs and a CPP are used in the robot to physically implement the walking strategy. An implementation framework for implementing SA was also given in the thesis to facilitate the reusability of some of the concepts proposed in the thesis. The thesis also presents the design and implementation of a learning method using Reinforcement Learning for the robot to learn the OWCI via real-time interaction with the environment. Experimental results show that the robot can successfully carry out its walking behaviours in various terrain conditions: including flat ground, incline, decline and uneven ground. Experimental results also show that the robot can successfully complete its learning task on the different terrains.

Enhancements of the current work could involve the robot carrying out other locomotion gaits (e.g. trotting and pacing). This requires more powerful cylinder movements to provide the robot with enough force to carry out this type of high speed locomotion. To achieve this goal, a set of higher quality sensors (e.g. a better balance sensor; vision sensor) as well as providing redundant sensors would need to be incorporated into the robot.

Enhancements to the existing work could also involve experiments with the robot in
terrains with greater inclines, declines and more uneven ground. This requires a better mechanical design and construction of the robot. At least, making it slightly more physically robust and able to lift its feet slightly higher. More appropriate sensors would also be beneficial, particularly those which could provide some analog information in order for the robot to obtain more accurate information about the different types of terrains. More behaviours will need to be built for the robot to handle “hard” terrains. For example, a *Jump* behaviour may be needed for the robot to cope with terrains having a gap or high obstacles.

As mentioned in section 6.3.2, an area of future work could also involve the learning experiments to be carried out for the robot to walk on a wider range of terrains that are beyond the limitations described in this thesis. These experiments could be used to investigate as to whether the robot has different values of the OWCI for terrains other than those described in this thesis. Of course these experiments could only be performed providing that any future changes in the mechanical design and construction of the robot is able to cope with these types of terrains. Results of these experiments may then provide a better understanding of the role that the OWCI plays in the robot’s walking on various terrains.

Other future work could also involve the automation of successive refinements of the WI used in the learning process and incorporation of an eligibility trace $c(\lambda)$ into the learning algorithm. These will have the advantages of not requiring any initial experimentation and of increasing the learning speed within each learning process. The basic idea is to setup an automatic process for the robot to gradually learn refinements of the interval used in the learning process from a coarse to fine
granularity. For example, in the beginning of the learning, the robot can set the interval to 200 ms. Once the robot has finished learning for this interval, it could then set the interval to 100 ms, etc. The process would continue until the robot has achieved a desired level of refinement of the learning interval (e.g. 10 ms). Determination of this desired level or when to stop the refinement would also have to be considered.

The learning of the robot could also be expanded in other areas. For example, it could be used for the robot to learn new behaviours, or to fine tune motions of existing behaviours. It could also be used to build dynamic subsumption rules of behaviours based on different real-time environmental situations. In such a case, more complicated emergent behaviours may be observed.
Reference:


Survey", available at:  
http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume4/kaelbling96a-html/rl-  
survey.html

Science.”, Appleton & Lange, Norwalk, CT.

Walking and Running of the Quadruped Using Neural Oscillator.” Autonomous  
Robots, 7(3), 247-258.

on Irregular Terrain - autonomous adaptation using neural system model.”. IEEE Int.  

Walking of a Quadruped Robot on Irregular Terrain Using a Neural System Model.”;  

Quadruped Robot Using a Neural System Model.” Advanced Robotics, 5(8), 859-  
876.

Motion Adaptation for a Quadruped Robot.”, Int. Symp. on Adaptive Motion of  
Animals and Machines (AMAM2003), Kyoto.

Adaptive Dynamic Walking of the Quadruped on Irregular Terrain.”, ISRR'99, 271-  
278.

Dynamic Walking of the Quadruped on Irregular Terrain.”; Robotics Research,  


animation, and biology.” *Autonomous Agents and Multi-Agent Systems*, 2(1)(March), 23-44.


New Micro Inc. (2000). “New Micro Inc.” , TX, USA.


Puterman, Martin L. and Shin, Moon Chirl (1978). "Modified policy iteration


Thorndike, Edward L. (1911). "Animal Intelligence".


