**Universidade do Minho**
Escola de Engenharia

Pedro André Moreira da Silva

# Adaptive quadruped locomotion: learning to detect and avoid an obstacle

Outubro de 2011

**Universidade do Minho**
Escola de Engenharia

Pedro André Moreira da Silva

# Adaptive quadruped locomotion: learning to detect and avoid an obstacle

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho efectuado sob a orientação de
**Doutor Cesar Analide**

e coorientação de
**Doutora Cristina Peixoto Santos**

Outubro de 2011

# Adaptive quadruped locomotion: learning to detect and avoid an obstacle

Pedro André Moreira da Silva

October 31, 2011

# Agradecimentos

Durante o desenvolvimento desta dissertação passei por diversos períodos de aprendizagem intensa e esforço redobrado no sentido de a concluir com sucesso. Tais períodos, bem como o caminho seguido para a conclusão deste trabalho, foram acompanhados por diversas pessoas que contribuíram e me ajudaram a atingir este objectivo.

Gostaria primariamente de agradecer ao meu orientador e co-orientador, os Professores Doutores Cesar Analide e Cristina Peixoto Santos. Também a todo o grupo do *Adaptive Systems Behaviour Group* (ASBG) da Universidade do Minho, no qual fui inserido e com o qual trabalhei e desenvolvi este trabalho, em especial ao Vitor Matos e Miguel Campos, alunos de doutoramento que me acompanharam mais de perto e cujo trabalho está relacionado com o meu.

A nível pessoal gostaria também de agradecer à minha namorada pelo seu apoio e suporte durante todo o processo de desenvolvimento, em especial à dedicação durante a fase de escrita deste documento. Também à minha família, à minha mãe e minha irmã, pela compreensão e preocupação durante todo o processo e ainda aos meus colegas e amigos Emanuel Braga e Pedro Soares, bem como a todos os outros, pelo suporte e presença.

Apenas com a presença e incentivo de todos me foi possível desenvolver este trabalho.

# Resumo

Autonomia e capacidade de adaptação são características chave na criação de sistemas robóticos capazes de levar a cabo diversas tarefas em ambientes não especificamente preparados nem configurados para tal. Estas características são comuns nos animais, sistemas biológicos que muitas vezes servem de modelo e inspiração para desenhar e construir sistemas robóticos. A autonomia e adaptabilidade destes sistemas advém parcialmente da sua capacidade de aprender. Quando ainda jovens, os animais aprendem a controlar o seu corpo e a movimentar-se, muitos mamíferos aprendem a caçar e a sobreviver com os seus progenitores.

Ultimamente tem havido um crescente interesse em como aplicar estas características no desenho e criação de sistemas robóticos. Nesta dissertação é proposto um mecanismo que permita que um robô quadrúpede seja capaz de detectar e evitar um obstáculo no seu caminho. A detecção é baseada num *Forward Internal Model* (FIM) que aprende a prever os valores dos sensores de percepção do robô, os quais procuram detectar obstáculos no seu caminho. Por forma a evitar os obstáculos, os sinais de detecção dos obstáculos são usados na criação de um mapa que permitirá ao robô alterar a sua locomoção mediante o que é necessário. Este mapa é construído à medida que o robô falha e tropeça no obstáculo. Nesse momento, o mapa é definido para que tal situação seja evitada no futuro. Ambos os processos de aprendizagem são levados a cabo em tempo de execução e mantêm esse processo activo por forma a possibilitar a readaptação a eventuais novas situações.

Este mecanismo foi inspirado nos trabalhos [14, 17], mas aplicados aqui a um quadrúpede com uma configuração de sensores diferente e específica. O mecanismo será interligado ao gerador da locomoção, baseado em *Control Pattern Generator* (CPG) proposto em [22]. Por forma a atingir o objectivo final, o controlador irá enviar comandos para o CPG a fim da locomoção ser alterada como necessário.

Os resultados obtidos mostram o sucesso em ambos os processos de aprendizagem. O robô é capaz de detectar o obstáculo e alterar a sua locomção de acordo com a informação adquirida nos momentos de colisão com o mesmo, conseguindo efectivamente passar por cima do obstáculo sem nenhum tipo de colisão.

**Palavras Chave: Controlador Adaptativo; Autonomia nos Robôs; Locomoção Quadrúpede; Aprendizagem de máquina; *Forward Internal Model*; Inspiração Biológica.**

# Abstract

Autonomy and adaptability are key features in the design and construction of a robotic system capable of carrying out tasks in an unstructured and not predefined environment. Such features are generally observed in animals, biological systems that usually serve as an inspiration models to the design of robotic systems. The autonomy and adaptability of these biological systems partially arises from their ability to learn. Animals learn to move and control their own body when young, they learn to survive, to hunt and avoid undesirable situations, from their progenitors.

There has been an increasing interest in defining a way to endow these abilities into the design and creation of robotic systems. This dissertation proposes a mechanism that seeks to create a learning module to a quadruped robot controller that enables it to both, detect and avoid an obstacle in its path. The detection is based on a Forward Internal Model (FIM) trained online to create expectations about the robot's perceptive information. This information is acquired by a set of range sensors that scan the ground in front of the robot in order to detect the obstacle. In order to avoid stepping on the obstacle, the obstacle detections are used to create a map of responses that will change the locomotion according to what is necessary. The map is built and tuned every time the robot fails to step over the obstacle and defines how the robot should act to avoid these situations in the future. Both learning tasks are carried out online and kept active after the robot has learned, enabling the robot to adapt to possible new situations.

The proposed architecture was inspired on [14, 17], but applied here to a quadruped robot with different sensors and specific sensor configuration. Also, the mechanism is coupled with the robot's locomotion generator based in Central Pattern Generators (CPG)s presented in [22]. In order to achieve its goal, the controller send commands to the CPG so that the necessary changes in the locomotion are applied.

Results showed the success in both learning tasks. The robot was able to detect the obstacle, and change its locomotion with the acquired information at collision time.

**Keywords: Adaptive robot controller; Autonomy in robotics; Quadruped locomotion; Learning; Forward Internal Model; Biological inspiration.**

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The work described in dissertation concerns the adaptive control of a quadruped robot. It proposes a mechanism that can be coupled to the locomotion's controller, and endows that robot with the ability to detect an obstacle and to learn to step over it. In order to address these goals the proposed solution uses biologically inspired concepts that grant the necessary features, such as adaptability to changing situations.

Biologic concepts are widely used to address very different problems that also concern adaptive control of robotic platforms. The following sections will give an insight to these problems and the type of solutions that can be used to address them. Then, the objectives and methodologies used to design and build the mechanism are presented.

## 1.1 Motivation

There has been an increasing interest in creating autonomous mechanical systems, robots, that can move and perform tasks in an unstructured and not predefined environment. These characteristics are very natural in animals, biological systems, which often serve as inspiration models to the creation of those machines.

Usually, when the creation of a robotic system is inspired on a biological system, it means more than mimicking physical properties or limb structure. It also includes the control structures and methods that make the biological system autonomous. The creation of a biologically inspired robotic systems might be seen by two perspectives. They can be used to move and act in the environment, autonomously carrying out hard and uninteresting tasks or dangerous ones, that can be injurious to human health. Or else, they can be seen as evaluation platforms for the control mechanisms and the physical properties that were thought to be a part of biological systems.

Autonomy in a robotic system is expected to make a robot capable to adapt to unexpected situations without the need for recalibration or without requiring instruments in the surrounding environment in order to safely move in it. Usually, these features require more than a mathematical model of the controller or an environmental geometric representation. Such approaches, are not suitable to deal with unexpected situations. Also, this type of interaction is not likely to be programable due to its extensibility or complexity in some cases. So, ideally, at least up to some degree, this type of controller should be able to learn and self tune according to its and the environment's properties.

There are several types of problems that can be regarded as dynamical or adaptive control. Some works are focused on autonomous robotic arm manipulator that seek to build a visuomotor map. This type of structure defines a relation between visual information and the robot's configuration or motor commands. It can be used to autonomously control a robotic arm to reach a specific point in its workspace, or follow a moving object with its end effector ([2, 25, 18, 27]). Other works are focused in adaptively control walking machines such as bipeds, quadrupeds or hexapods. Some locomotion controllers seek to adaptively an autonomously perform tasks such as overcoming obstacles, moving in rough or uneven terrain, or walking up and down slopes ([11, 9, 14, 20]). Also the control of wheeled or tracked robots seek to achieve adaptive and autonomous control in order to create autonomous behaviors, such as navigation or obstacle avoidance ([29, 19]). Although these problems are very different, they all share a common goal: building a controller that grants the robot autonomy and adaptability when interacting with the environment.

In most cases the adaptive controller is required to capture the existing relation between the sensing of the environment and the robot's state. This usually means that the robot needs to define the relation between the sensory information and the motor commands that somehow will change the environment, or at least the robot's perception of the environment. Or else, this relation might be used to define the robot's actions according to its sensory information, its state and its goals. The required type of relation might be very hard, or even impossible in some cases, to achieve by using analytical models or by manual definition. The high nonlinearity, usually observed in this type of relations, and the high dimensionality of both side variable spaces, make such type of approach unfeasible. Also, there is not a complete knowledge about the system, or the environment, which makes it even harder to manually or analytically define this relation.

Some well known biologically inspired machine learning methods are capable of acquiring highly non linear relation or to gradually adapt the control to necessary conditions. Therefore, this type of approach can be applied to the creation of autonomous and adaptive controllers. The Artificial Neural Networks are models inspired in the animal nervous system; Reinforcement Learning is a unsupervised learning technique that relies in learning through reward and penalty; and Genetic Algorithms (GA) are optimization algorithms based on the Darwin's evolution theory.

Ideally this required relation, or part of it, should be acquired autonomously, according to the required goal. This should happen online through a learning process that adapts the controller to the environment and to the mechanical system it is controlling.

This document presents a first attempt to create an adaptive controller that will allow a quadruped robot to detect an obstacle in its path and to learn how to avoid stumbling on it. The detection of the obstacle relies on self generated expectations for a set of preceptive sensors that scan flat ground in front of the robot. In order to avoid the obstacle, the robot builds a map that dictates the necessary alterations to the locomotion according to the obstacle detections. Both tasks are learnt online and keep that state throughout the "life of the robot".

## 1.2 Objectives

The mechanism proposed here presents a way of endowing a quadruped robot with learning capabilities, that will enable it to detect and avoid an obstacle in its way. The robot's locomotion is achieved by a bio-inspired architecture that uses CPGs based on dynamical systems. The mechanism will be applied to that locomotion controller in order to use the acquired knowledge to adequately parameterize the CPG. This will result on the necessary alterations to the locomotion that will grant the robot to achieve its goal.

The main tasks concern the detection of the obstacle and the creation of a mapping between these detections and the required changes to the locomotion. Both the tasks require a learning process that will be carried out online and is maintained even after conversion. This allows the controller to adapt to new conditions that might come up.

This work is inspired in Lewis and Simó work in [14, 17], but defined here to a quadruped robot instead of a biped. Besides, the type of sensory information is different and the quadruped is not fixed to anything. As it is able to move freely, the controller is also required to address the stability and steering problem, which in a quadruped robot should not be so hard.

Follows a list of the objectives that will define the steps to design and build the proposed mechanism:

- Have a testing environment that grants what is necessary for the mechanism to be evaluated, including an adequate obstacle.
- Configure the robot's perceptive and proprioceptive sensors and grant its correct placement and orientation.
- Structure all the information about the robot's state and its environmental perception in a way that meets the requirements of the remaining tasks.

- Achieve the creation of reliable expectations about the robot's environmental perception values, exploiting the relation that exists between the usually observed pattern and the locomotion cycle.
- Filter the difference between the expectations and the acquired values in order to discern the obstacle detections from the noise, taking into account the reliability of such expectations.
- Create a mechanism that enables the robot to acquire information during the failed attempts to step over the obstacle, thus learning how to avoid such situations in the future.
- Define a way to change the robot's locomotion parameters in the CPG, according to the knowledge acquired during the previous encounters.
- Evaluate the performance achieved by the mechanism through several tests. These tests shall evaluate all the components behavior according to the expected results. Also, verify the mechanism capability to actually fulfill its goal.

## 1.3 Methodology

The proposed objectives were tackled into several tasks that seek to fulfill the goals and guarantee its correctness. Each one is then related to its execution's details in the remaining chapters.

The quadruped locomotion architecture based in dynamical systems were created in [22]. This mechanism must be coupled to this locomotion architecture in order to produce the alterations through an adaptive learning process. The definition of such mechanism was then divided in the following tasks:

**World and robot :** Define the environment in which the mechanism will be tested and configure the robot to the mechanism's implementation. The environment creation must take into consideration the obstacle properties and the robot's need to reach the obstacle in a specific way. The robot's configuration concerns the sensor placement and orientation, in order to obtain the desired effect, the scanning of the ground in front of the robot. This task is described in section 4, where the evaluation platforms and their environments are presented.

**Creating previsions :** Processing the raw data and define its structure. Also, define a prevision layer capable of generating expectations for the perceptive sensor inputs. The creation of expectations must consider prevision models that are fit for the purpose and that grant the learning capabilities and the adaptability requirements. The learning should be an ongoing process that allows the robot to continuously adapt to new conditions. Also, it is required that these hypothesis are tolerant to noise and to the actual detection of obstacles. Chapter 3 presents the implementation details of this tasks.

**Filtering the noise :** In order to detect the obstacles, it is necessary to eliminate the noise and discern the obstacle detections from it. This task addresses the definition of such mechanism, which takes into account the reliability of the created expectations in order to achieve such goal. This task's details are also presented in chapter 3.

**Learning to avoid the obstacle :** Whenever the robot has an encounter with the obstacle, it tries to step over it. If it fails, it must learn from that situation in order to avoid it in the future. This requires an iterative learning process, which builds a structure that dictates what are the necessary changes to the locomotion to avoid the same failed attempts. The details about this task's implementation are also presented in the chapter 3.

**Changing the locomotion :** Define a way to use the acquired knowledge about the changes to the locomotion to actually change the robot's locomotion, whenever an obstacle is detected. Chapter 3 details the definition of this task.

**Evaluate the mechanism :** Evaluate all the elements that compose the system, comparing their results to the expected ones. Then evaluate the mechanism's performance about the main goals: the obstacle detection and the ability to step over the obstacle avoiding any collisions. Chapter 5 shows these tests and results, describing the achieved performance according to what was expected.

**Similar works and problems :** The search for similar problems and solutions gives a deeper knowledge about the adaptive control and about other possible approaches. So, this task is carried out in parallel to the remaining ones in order to compare the proposed methods to different ones. Also to be aware of the several types of problems in adaptive control and their solutions. Chapter 2 presents several works that are related to the one presented here.

## 1.4 Document Structure

This dissertation is organized as follows. Chapter 1 presents the introduction, where the theme is introduced and the problem, the objectives and methodology are presented. The following chapter 2 presents works that relate to this one by goal, type of solution, or principles in the solution's definition. In chapter 3 the proposed mechanism is presented, where all the components are defined alongside with their specific goals and purposes. The application of the mechanism to the robots and the environment definition are presented in chapter 4. It includes the robot's configuration details and the environment creation. Chapter 5 presents the tests and obtained results that evaluate each of the components that compose the mechanism, as well as the mechanism itself. Finally the conclusions are presented in chapter 6, where the results are discussed alongside with the objectives and possible future enhancements.

# Chapter 2

# Related Work

The main purpose of the presented document, is to design a learning module that can be couples to a quadruped robot's locomotion controler and endows it with the ability to detect and avoid an obstacle in its path. Also, the robot should perform online learning to achieve both goals. The learning should remain active even after conversion, which enables the mechanism to adapt to new conditions. The architecture seeks autonomy in the learning and execution of the required tasks, which means that the robot adapts it self to the environment in order to achieve its goals.

Although different goals might have different requirements, the principles that drive and inspire this type of control are widely investigated in several kinds of problems (e.g. robot legged locomotion and robotic arm control). These principles concern the autonomy in the robotic system control. Those considered to be more relevant here, are presented next. The robot should possess a certain level of self-awareness that enables it to expect the self caused changes in the sensory inputs, which are consequences of self generated movements. There should exist entrainment between the controller and the mechanical system, which allows the adequate and optimized control of that mechanical system. Also, the motor actions should be chosen according to the system's sensory inputs, configuration state, and goals

This chapter will present several approaches to adaptive control in different types of problems. The goal is to give an insight of several types of solution and how they are applied to different kinds of problems. Also, how these solutions relate to this work, by pursuing the same goal or using similar methods in their solutions.

## 2.1 Legged Robot Locomotion

A challenging and complex state-of-the-art problem is the realization of real-time robotic walking control systems. Legged robots involve the coordination of a high number of degrees-of-freedom. Further, it is an under actuated problem; needs to take advantage of the robot and environment dynamics; to address balance and steering; gait switching; visually guided feet placements; and needs to adapt to perturbations and uncertainties. However, the ability to traverse a wide variety of terrains while walking is basically a requirement for performing useful tasks in our human centric world.

### Adaptable quadruped locomotion

The work of Fukoaka and colleagues presented in [5] proposes a controller based on biological concepts that enables a robot to autonomously adapt its locomotion to rough terrain. The solution uses a neural controller composed of a CPG and reflexes, that grants the required stability and energy consumption optimization during locomotion. Furthermore, a passive ankle was designed to enhance the robot's locomotion and adaptability. Later, this work has been extended to enable the robot to walk on natural ground [12]. This included a set of new reflexes to help the robot to deal with the wider range of unexpected situations.

A reflex is used here as a response to a specific situation. It seeks to grant a specific parameter that is required to assure the robot's stability and well being. Besides, the CPG is entrained with sensory feedback to adapt itself to the locomotion and the reflex execution. These are the key features that create the system's ability to produce adaptive locomotion in rough terrains and natural ground.

These proposed solutions present an example of a robust controller that responds to different types of possible disturbances, and adapts the locomotion to a challenging terrain. Besides, the physical properties of the robot were also defined to enhance the robot's locomotion. On the work proposed in this dissertation, the goals are very specific; the robot is required to learn how to detect the obstacle and how to avoid it. Therefore, the same type of approach do not apply. Nevertheless, these studies propose a robust solution to an adaptive locomotion controller.

### Head motion stabilization

Head stabilization during locomotion is a problem that relates to robot legged locomotion, although it is not explicitly about its control. In order to achieve a stable image acquisition during quadruped locomotion, [26] proposes a way to stabilize the robot's head resorting to CPG and Genetic Algorithms (GA). The goal is to obtain an optimized set of parameters - offset, frequency and amplitude - to parameterize a CPG used to control head movements. These movements seek to compensate the robot's oscillations during locomotion. Thus, more stable conditions are created to image acquisition, which can enhance several visual perception dependent tasks.

Although such an approach relies in offline optimization, which differs from the solution presenter here, the learning seeks the finding of optimized parameterization for the robot's head motion controller. Thus, the adaptation of the controller to the specific mechanical system it is optimal.

### Optimizing quadruped locomotion

In [13] the authors present the results of applying four methods into optimizing quadruped locomotion. The evaluated methods were: GA; Amoeba algorithm, Hill Climbing and Policy Gradient algorithm. The testing platform was the AIBO and they sought to achieve optimal controller parameterization, using the robot's velocity as tha main optimization target. The goal is achieved since the obtained results were better than the known hand tuned parameters.

This is another example of offline parameter finding. The controller can be autonomously defined to better control its mechanical system. This work falls from the presented dissertation scope but defines an usual way to approach optimal entrainment between the mechanical system and the controller.

### Learning to classify possible foot positions

A work presented in [4] proposes an approach that enables a quadruped robot - the LittleDog - to select collision free foot placement positions on rough terrain. After selecting a set of random possible choices, the robot uses a classifier to discern those that are collision free from those that are not. The classifier is trained offline by a supervised learning model - the authors evaluated both Support Vector Machine (SVM) and Ada-Boost to build the classifier. After learning, this classifier can be applied to real time reasoning about the possible collision free foot placements.

This approach concerns a quadruped robot controller that chooses foot placement points in the environment and generates locomotion according to a Center of Gravity's (CoG) desired trajectory. The presented work defines a way to improve such a controller to avoid foot placements that could result in a collision with the environment by using a specifically trained classifier. The training is executed offline but after it is complete, the robot is aware of its own limitation and is able to choose the safer foot placements.

This approach ([4]) uses machine learning to enhance a controller to achieve adaptable locomotion in rough terrain. Despite the offline learning, it is able to perform adaptable and autonomous locomotion in harsh conditions.

### Locomotion over rough terrain

Another interesting approach is presented in [10], where the authors propose a controller that enables the LittleDog robot to traverse a patch of terrain with a high level roughness. The performed locomotion is fast and nearly autonomous. That controller does not present any specific method for control, it breaks down the problem into smaller ones and addresses each at a time. However, despite the fact that the most of the processing is done online, knowledge about the terrain's spots and an approximation of the best path, needs to be fed to the robot before each crossing. This considerably reduces the perception problem and provides the robot enough knowledge and processing time to address the remaining challenges.

This approach achieves remarkable results in controlling a quadruped robot through harsh conditions. Although it is able to deal with high levels of roughness, it requires a base of a priori knowledge about the terrain that is going to cross, as well as a pre computed optimal approximated path. Thus, it does not perform fully autonomous locomotion over rough environments and it is unable to traverse this type of terrain without a priori knowledge about it. Nevertheless, the achieved results indicate that a robot can perform fast walking over terrains with a level of roughness that were thought to be nearly impossible to be traversed by a walking machine.

### Quadruped locomotion with adaptable posture control

The studies presented in[9, 1] define controllers to the walking machine BISAM that seek to produce adaptive quadruped locomotion.

In [9], the authors present a controller based in coupled neural oscillator, that operates at three different levels: joint level, intra-leg coordination and inter-leg coordination. Each level has a neural structure that enables the system to perform stable locomotion. The locomotion is generated by the interaction of two different levels, intra-leg and joint control, which then controls the neural oscillators placed at the joint level. The system uses Reinforcement Learning (RL) to learn the correct sensorimotor interaction that triggers the reflexes, in order to perform adaptive control. This includes reflexes that grant the robot's posture and the adaptation to uneven terrain. The learning tasks were evaluated in simulation of a single leg.

Another controller proposed in [1] uses Center of Gravity (CoG) trajectories and inverse kinematics in order to implement the locomotion. A posture control similar to the previous study ([9]) was implemented, resorting to fuzzy rules and RL. The robot learns to trigger reflexes in order to grant its posture and stable control. This enables it to perform adaptive locomotion and to surpass unknown obstacles.

Both proposals seek adaptive posture control in order to achieve stable locomotion. The latter [1] reported results in the control of a robot, which means that the machine was able to self tune in order to achieve its goal. These approaches define a way to adaptively control a quadruped robot and to enable it to surpass obstacles. However, this is achieve only by using a set of predefined responses to a certain range of specific situations. In our case, the goal is to sense the obstacle and learn how to adapt the locomotion in order to avoid the it. The robot is not specifically told about what an obstacle is, or how to avoid it. It simply learns to avoid undesired situations, the collisions with the obstacle.

### General CPG controller for quadruped locomotion

Another work that concerns adaptive quadruped locomotion presented in [30]. The authors propose a CPG controller capable stably generate all primary quadruped gaits. Also they present a biometric controller for a generic quadruped robot, that entrains the presented CPG with several reflex modules. These reflex modules seek to grant the robot's desired response to certain situations in order to maintain the robot's stability and posture.

The proposed work in [30], defines a generic controller that seeks adaptable and stable locomotion through a CPG and entrained reflexes. Although the concept is relevant in the field, since it performs autonomous locomotion, it does not address the theme pursued in this work. The learning processes applied to detect environmental features and to learn how to avoid undesired situations.

### Coupling Locomotion Generations with Sensory Feedback

The integration of continuous sensory information in the locomotion generator, namely a CPG, is an important concept to adaptive locomotion control. Studies such as [8], propose a way to entrain the sensory information directly in the CPG controller through their phase. It observes the phase in the cyclic sensory input and defines the output of a master-oscillator-base CPG that drives the different joints. This enables the synchronization of the generated movements in all joints with the perceived cyclical sensory information, in order to achieve an adaptive locomotion. It allows the generation of locomotion in several types of legged robots.

Although the achieved level of adaptivity in [8] differs from the one sought here. That proposed solution presents an important concept that enables legged locomotion closely coupled with the sensory feedback. This feature grants the generated locomotion to adapt to the robot and to the environment, thus the controller is tuned to the mechanical system it is controlling and to the environment.

In [24] the authors present a generic way to build a CPG controller based in coupled oscillators. This controller is designed in a way that can generally be applied to legged locomotion. Furthermore, they present a systematic way of adding sensory feedback to that CPG, such that the achieved controller is strongly coupled with the mechanical system it is controlling.

This type of controller seeks better performance by coupling the CPG with the sensory feedback. Such feature grants the self tuning of a controller to the environment and the robot, since it includes performance information - the sensory feedback - in the locomotion generation. It defines an interesting way of adequately build controllers to specific mechanical system, taking into account the mechanical system it is controlling and the environment where it is inserted.

### Learning biped locomotion with Reinforcement Learning

In [23] the authors resorted to Reinforcement Learning (RL) in order to acquire biped locomotion. They apply an actor-critic RL technique to the neural oscillator based CPG, in order to learn the locomotion parameters. The learning process seeks to define the weights for the sensory feedback signals in order to acquired the desired locomotion. The authors call the designed controller an CPG-actor-critic model.

The proposed controller defines a way to autonomously acquire locomotion by simply observing its performance through sensory feedback. The fact that the locomotion is acquired autonomously, grants the correctness of the mechanical system control. This is another example of adaptive locomotion generation, which differs from our work that focuses in learning tasks concerning the detection and avoidance of an obstacle. Nevertheless, the proposed approach in [23] defines an ideal solution to adaptive locomotion control. However, the reported results show that the locomotion acquisition by the CPG-actor-critic models requires too much time, which makes it inadequate to real robots.

### Defining a neuronal controller

To control a quadruped walking machine, the authors in [21] build an oriented modular neural controller. The proposed work defines a controller composed of three neural modules: one responsible for processing the sensory input; another that creates the cyclic locomotion patterns, and the third to control and regulate the robot's velocity. The final controller is able to explore an indoor environment, avoid obstacles and get out of dead-lock situations, such as getting suck in a corner.

The proposed controller is designed in order to produce a reactive behavior in that specific robot and environment. It grants the robot the required abilities to achieve its goal. However, it is not able to answer to situations other than the predefined ones. Thus, the robot is limited to act according to its design, it cannot adapt to new conditions or situations. This fails the learning and self tuning to the execution of the tasks. Principles that define our work.

### Forward internal model and its biological inspiration

The biological concept of efference copy states that a copy of the generated motor commands - efference signals - in animals (and robots) can be used to expect the sensory consequences of such acts [31]. These expectations can help distinguish the sensory input caused by the robots movements, an reafference signal, from the ones caused by external factors, exafference signals. Later, another study [7] revealed that the efference copies cannot be directly compared to the afference signals (the complete sensory input information) due to their different dimensionalities. A Forward Internal Model (FIM) was proposed in [32], as a neural method to "translate" the copies from the motor signals's configuration space to the space that describes the sensory inputs.

The discerning of externally caused signals from those generated by the robot, enables the system to perceive environmental features that might be relevant to the robot's control. This concept is used in our work to detect the obstacle. The robot uses an internal model to create expectations about range sensors readings, that expects flat ground. If an obstacle is in the robot's path, this perception will be altered, creating an exafference signal.

Although these studies [31, 7, 32] do not propose an adaptive locomotion controller, they define rather relevant concepts to build one. Some works concerning these concepts are presented next.

### Using the concept of Forward Internal Model

In [28], the authors used the biologically inspired concepts of FIM and efference copy to detect changes in the ground's slope. The detected changes are then used to stabilized a biped robot locomotion - which movements are restricted to the saggital plane - when the ground's slope changes. The presented solution uses an FIM that predicts the expected values of the accelerometer sensor when the robot moves in a given type of terrain (flat). If the terrain's slope changes, the robot senses the difference in the accelerometer and detects the new slope. Then, according to the detected changes, an active upper body component changes its configuration, leaning the robot forward or backwards in order to compensate the changing slope. This mechanical component uses a weight that allows the robot to shift its center of mass forward or backward.

The detection of slope changes is based in an FIM previously trained to a given type of ground's slope. Also, the responses in the upper body component are predefined and designed offline. Although the type of approach in [28] uses the same concepts as ours, their solution does not address the online learning of the internal model or the actuator's responses. Furthermore, our solution keeps the learning active through time, enabling the continuous adaptation to new conditions.

### The works that inspired the ons presented here

The work presented in this dissertation is partially based on [17], where the authors present a mechanism that enables a biped robot to change its stride length in order to avoid a detected obstacle. The necessary changes to the stride length, that will grant the robot to achieve its goal, are learned every time the robot steps on the obstacle. Therefore, the robot learns through experience. The learning is an iterative ongoing process that is only complete when the robot stops stepping on the obstacle.

In a later work that also inspired ours ([14]), the authors specify the method to detect the obstacle. Until then they had simply considered an element, which they called *Range Encoder*, that detected the obstacle at different distances. The obstacle detection is based on the concepts internal model and efference copy that is applied to create expectations about the robot's visual disparity information. These sensors scan the ground in front of the robot and its value changes according to the locomotion pattern. When the object is in front of the robot, it will disrupt the usual activation pattern and the robot will know about its presence. Furthermore, these obstacle detections are used to update a weight map at time of collision with the obstacle, so that such situation is avoided in the future. The two tasks are trained online and are kept active throughout the "life of the robot". This enable continuous adaptation to possible new conditions.

Later, a similar work [15] presents an approach for detecting obstacles in the robot's path using optical flow in stead of disparity. That work did not include the obstacle avoidance task, it simply focused on the detection using optic flow. However, it relies on the same principles to perform the detection.

This document presents an architecture that works similarly to these ones, however, the used robot is a quadruped and the sensor configurations are different. The quadruped locomotion offers better stability, nevertheless, in our work the robot it is not thetered, as were the biped in the mentioned works. This means that it will be necessary to address the robot's stability and balance problem.

### Learning quadruped locomotion

In [16] the authors define a control model that enables a quadruped robot to quickly learn a stable locomotion. To achieve this, they use Unit CPG (uCPG), a set of simple reflexes and adaptive models. The uCPGs are defined by the authors as functional units that together, control the robot's movements. The stable locomotion is maintained both by the reflexes and the adaptive models. Each reflex is triggered by a specific sensor signal and helps stabilize the robot by granting a required parameter. The adaptive models use forward internal models to create expectations for sensory input according to the motor commands. Also, it applies changes to the uCPGs when detects undesired differences in the acquired sensor values.

The presented work in [16] proposes a model for acquiring basic locomotion gait in a quadruped. It is inspired in the responsive and instinctive gait acquisition in quadrupeds, such as a foal. Reflexes and adaptive models generate the necessary responses to certain requirements that grant the robot stable locomotion. This dissertation seeks learning at a higher level of abstraction. Nevertheless, the controller proposed in [16] is interesting since the robot learns autonomously.

## 2.2 Other Control Problems

The self awareness of a system seems to be a key feature to build an autonomous controller. Such feature is usually hard, or even impossible, to be user defined or programmed, specially when there are a high number of degrees of freedom. So, the ideal solution is to have this map defined by the system, according to its experience.

### Visuo-motor Coordination

One common problem of an internal model based solution is a visuo-motor coordination of a robotic arm. The required model should recognize the visual consequences of the robotic arm movements. Such problem has been widely studied in the field of robotics and biological inspired control.

An example of such a system is [25], where the authors present a learning mechanism through a self-discovery phase. Due to the long time required to learn through a motor-babbling phase, where the configuration space is too big, the authors developed a mechanism to reduce the learning time. They created a state confidence mechanism that helps the robot to perform active learning and search the most relevant parts of the configuration space. The results show that the learning time was reduced, when compared to simple reinforcement learning, where there is no rule to active learning.

Another example of this kind of work is presented in [2]. The authors defined a learning mechanism based on Self Organizing Maps (SOMs) and the biological concept of internal model. The goal is to enable a robotic arm to reach a three-dimensional point in the robot's work space, and to relate those movements to the robotic's arm internal state and visual information. The robot creates a relation between the visual sensory information and state spaces, and the work space in which it can operate.

A different approach is presented in [3], where besides the building of a visuo-motor map and the dynamical systems used to generate the robotic arm's movements, the authors seek to achieve imitative behaviors. The defined approach supports the explicit imitation of external actors, e.g. humans, and emergent imitative behaviors that mimicked full, multi staged movements. Both the imitation behaviors are achieved as a side effect of the learnt visuo-motor map. These behaviors can be used to control the robotic arm simply moving an arm in the robot's visual space, i.e. in front of the robot's camera.

All these works achieved interesting results in the construction of a self aware robotic manipulator. In addition to the self awareness about generated movements, the solutions were also able to generate the motor commands to reach a specific point in the visual space. Although these works a specific problem, the designed solution relies in principles relevant for most of the autonomous control problems. This dissertation presents a work based on these principles, however applied to a different problem.

### Autonomous Locomotion

Affordances ([6]) and sensorimotor maps are widely known concepts in robotics and rather relevant when designing of a robot's controller. A sensorimotor map defines a self aware structure that the robot uses to expect sensory alterations according to motor commands, and the other way around; to define motor commands according to a desired robot state in the environment, or a perceived environmental features. The latter can also be understood as an affordance, which defines a possible action according to a perceived situation, an affordable action.

The authors of [19] propose an adaptive controller that grants these features. Plus, how the robot - a wheeled robot - can project this model into the feature in order to exploit affordances and achieve goals. This was applied to robot navigation problem, in which the robot must know what is where and learn how its internal state and its environmental perception, are changed by the self generated motor actions. After learning the sensorimotor map and to recognize affordances, the robot is able to define ways to achieve its goals.

When successfully built, such a controller is able to autonomously act in the environment and to pursuit goals. However when dealing with unpredictable and dynamic environments, such control becomes complex and is very hard to define. Our proposed model is applied to a legged robot and the control problem is restricted to online learning that seeks to detect and avoid obstacles in the robot's path.

A different kind of controller is presented in [29], where the author defines a goal seeking behavior that enable the robot to select actions that will get it one step closer to its desired sensory state. The controller uses two Self Organizing Maps (SOMs) - associative memories -, one for creating a continuous representations of the situation space, and another to generate the actions that will shift the state from the current one, getting the robot one step closer to its final goal.

As the previous work, this one does not concern a legged locomotion, or limb coordination problem. Instead it is applied to obstacle avoidance in a wheeled robot. The controller adapts to the robot and to the world, and learns how to maintain a good performance while pursuing its goal. Such a controller does not concern the robot's specific physical properties, it focuses in the sensors readings. This means that it adapts the robot to the world, independently of the robot it self, it simply seeks to fulfill its goal. Although not very applicable to robot legged locomotion due to its requirements in the close coupling of mechanical system and movement control, the goal seeking behavior learning poses an interesting approach to learning adaptive and autonomous control.

# Chapter 3

# Mechanism Description

The proposed mechanism defines a quadruped robot controller that is composed of two main parts. The first one detects an obstacle in the robot's path and the other uses these detections to iteratively learn to avoid those obstacles. The detection part uses an internal model to detect differences in the environment perception that suggest the existence of an obstacle in front of the robot. The second part creates a map from the obstacle detections to the necessary changes in the locomotion, that will enable the robot to avoid stepping on the detected obstacle. This map is built iteratively every time the robot steps on the obstacle. Both parts perform online learning and maintain that state even after the initial convergence. This allows the robot to adapt to changes in the perception conditions.

Figure 3.1 shows the mechanism's architecture, where all the components and subsystems are present. The details about the thought solutions and each component are presented in the following sections.

## 3.1 Detecting the obstacle

Robot legged locomotion is performed through a series of cyclic movements in the limbs. The period of this cycle is defined by the duration of one stride. It is expected that the robot's proprioceptive and environmental perception information are related to this cycle, since the generated movements affect several aspects of the robot. This relation is the base for the obstacle detection task, which is carried out in this first part of the mechanism.

In the proposed work, the robot uses infra-red range sensors to scan the ground in front of it. As the sensor readings vary with the robot's movements during the locomotion cycle, a Forward Internal Model (FIM) is used to create expectations about the those values when facing flat ground. The expectations are created with relation to the robot's state in the locomotion cycle, which is defined by proprioceptive information such as leg joint positions or robot's body pitch angle. These values are closely related to the robot locomotion cycle

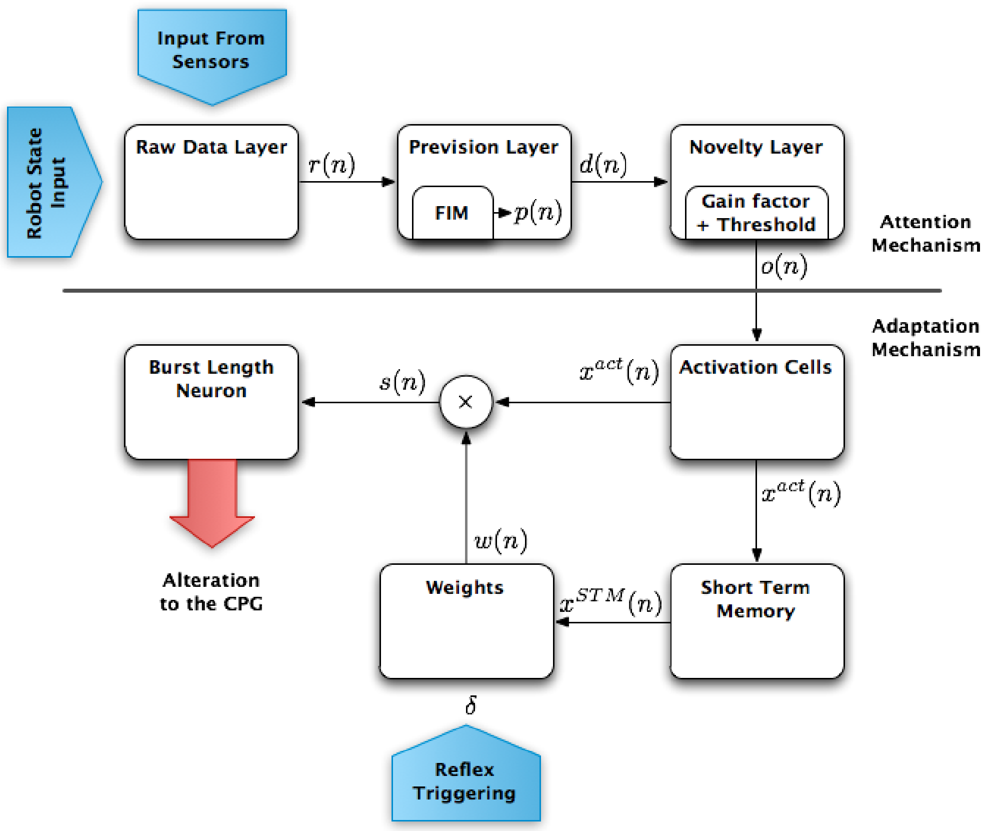

**Figure 3.1:** Overall system architecture. FIM stands for Forward Internal Model.

since some of them actually help generate the locomotion (e.g. as the joint information). When facing an obstacle, the range sensors acquire different values from those expected by the internal model. This difference denounces the obstacle presence in the robot's path.

Three layers are used to implement this internal model and to produce the obstacle detections. The raw layer, that processes the information from the sensors; the prevision layer, which implements the creation of expectation for the range sensors; and the novelty layer, that filters the noise. Their implementation is detailed next.

## Raw Layer

This layer processes the raw data from the robot's state and the range sensors into a structure that fits the requirements for the prevision layer and the rest of the mechanism. The raw data structuring needs to take into account the locomotion cycle. First, the information is divided into separate strides, and then each one is split into $n_p$ equal chunks, each one defines a specific phase of the stride. This division is applied to both the robot's locomotion state and the range sensor information.

Dividing the stride into discrete blocks means that there will be several values for each block. In order to solve this problem, the data acquired during the period of time correspondent to one stride phase will be averaged. This is actually useful because it helps filtering the noise. Since the periods of time for each stride phase are quite small - a fractional part of one stride - there is hardly any risk of relevant information loss.

During each stride phase $j$ (where $j \in [1, n_p]$) the state of the robot $x_j$ is defined by $n_f$ signals of proprioceptive information - each robot has its own set as described in the implementation chapter 4. Each one $x_k$ ($k \in [1, n_f]$) will defined by the average of the acquired values during that stride phase's period. The same process is applied to the values acquired by each of the range sensors, during the stride phase $j$: $y_{ij}$, where $i$ defines the range sensor ($i \in [1, n_s]$).

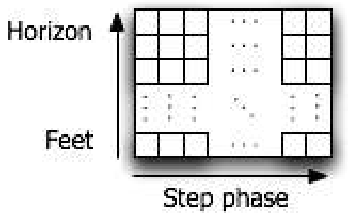

**Figure 3.2:** Structure that defines the relation between the distance of the obstacle and moment of the stride that the obstacle is detected.

The processing of raw data also includes the value normalization, when that applies. The range sensor values are normalized according to the highest value acquired by a given sensor. The proprioceptive values normalization depends on each implementation and the type of proprioceptive sensors. For instance, a binary touch sensor does not require normalization.

For a complete stride we get a matrix structure (figure 3.2) of $n_s \times n_p$ elements. Each matrix cell holds the range sensor value for one stride phase and one of the range sensors. A cell also can be interpreted as a scanned distance, which depends on the range sensor, during a specific moment of the stride. This structure is maintained throughout the mechanism as a reference to space and moment of the stride. It helps the robot to learn and perform the required tasks.

The output of this layer is produced every stride phase, therefore, for each stride phase $j$ the layer's output is given by:

$$r_j = (x_j, y_j), \tag{3.1}$$

where $x_j$ is the state of the robot during stride phase $j$; $y_j$ is a vector of all the range sensor readings during that stride phase; and $r_j$ is the output for the raw layer.

## Prevision Layer

This layer has two main purposes: to create expectations for each of the cells defined previously, and to output the difference between the raw values and the expectations for each cell. The difference defines the cell's error. This layer holds the Forward Internal Model (FIM) that creates expectations to the range sensor's values, taking into account the robot's state during the locomotion cycle. When the difference between the expectations and the raw data is considerable, it is possible that an obstacle has been detected in the robot's path. The difference is computed following:

$$d_{ij} = y_{ij} - h_{ij}(x_j), \tag{3.2}$$

where $y_{ij}$ is the acquired value by the range sensor $i$, during the step phase $j$; $h_{ij}(x_j)$ is the expectation for that same cell, and $d_{ij}$ is the difference between them.

Two approaches were considered and evaluated to create the expectations: the Least Mean Square (LMS) learning rule and the Newton's method. The first is a linear regression model and the second is a maximization/minimization algorithm. The expectations have the general form of:

$$h_{ij}(x_j) = \omega_{ij0} + \sum_{k=1}^{n_f} \omega_{ijk} x_{jk}, \tag{3.3}$$

where $\omega_{ijk}$ is the applied weight to one of the elements of the robot's locomotion state $x_{ijk}$ and $\omega_0$ is the bias term, which input is always 1. The state of the robot's elements will be the inputs, or features, for the internal model. They will be used as locomotion phase reference in order to produce the expectations about the range sensor values. In machine learning nomenclature, the range sensors are often called targets. To measure the internal model's performance, both models will use the mean square cost function:

$$J_{ij}(\omega_{ij}) = \frac{1}{2} \sum_{n=1}^{m} (h_{ij}(x_j^{(n)}) - y_{ij}^{(n)})^2, \tag{3.4}$$

where $J_{ij}(\omega_{ij})$ is the cost for computed expectations of cell $(i, j)$ (number of sensor and stride phase, respectively) according to that cell's weight vector $\omega_{ij}$. $m$ is the number of training samples, in our case it means the number of strides. So, the cost is computed by the square error from all $m$ the training samples. Both approaches seek to find the best weight vector $\omega_{ij}$, such that the cost $J_{ij}(x_j)$ is minimum. This is done for each cell. The difference between the proposed approaches is the method used to search the weight vector.

As mentioned before in the beginning of this section, the expectations are created based on the relation that exists between the range sensor values and the robot's state during the locomotion cycle. Both change cyclically with the locomotion. Therefore, each stride phase will have similar activations in different strides, both in the robot's state and the range sensor values. So, there is a linear relation between them for every stride phase.

### Least Mean Square

The LMS learning rule will search the weight vector $\omega_{ij}$ that minimizes the cost function $J_{ij}(x)$ for each cell $(i, j)$. To search this vector, the LMS updates each weight according to the steepest descent rule:

$$\omega_{ijk} := \omega_{ijk} - \mu \frac{\partial}{\partial \omega_{ijk}} J(\omega_{ij}), \tag{3.5}$$

where $\omega_{ijk}$ is the weight $k$ that belongs to the cell $(i, j)$, and $\mu$ is the learning rate. This algorithm converges to the optimal weight as long as $\mu$ is not too big. If so, the method might fail to converge. We need to get the partial derivatives of $J(\omega_{ij})$ according to a single weight $\omega_{ijk}$. In order to simplify the calculation, let us consider only one sample $n$, instead of all $m$ training examples. Combining the steepest descent rule 3.5 with the cost function 3.4 we get :

$$\begin{aligned}
\frac{\partial}{\partial \omega_{ijk}} J(\omega_{ij}) &= \frac{\partial}{\partial \omega_{ijk}} \frac{1}{2}(h_{ij}(x_j) - y_{ij})^2 && (3.6)\\
&= 2\frac{1}{2}(h_{ij}(x_j) - y_{ij}) \frac{\partial}{\partial \omega_{ijk}} (h_{ij}(x_j) - y_{ij}) && (3.7)\\
&= (h_{ij}(x_j) - y_{ij}) \frac{\partial}{\partial \omega_{ijk}} \Big( \sum_{u=1}^{n_f} \omega_{iju} x_{ju} - y_{ij} \Big) && (3.8)\\
&= (h_{ij}(x_j) - y_{ij}) x_{jk}. && (3.9)
\end{aligned}$$

Where $u$ is a secondary variable that iterates the features (the elements composing the state of the robot). $k$ defines one of the features, the one used to calculate the weight update rule, and $x_{jk}$ is the $k^{th}$ feature of the robot's state. The variable $u$ was used to differentiate the two ways the features are present in the equations. However, when $u = k$, they mean the same. Thus, the partial derivative in the penultimate expression results only in $x_{jk}$. So, we get the following weight update rule:

$$\omega_{ijk} := \omega_{ijk} + \mu(y_{ij}^{(n)} - h_{ij}(x_j^{(n)}))x_{jk}^{(n)}. \tag{3.10}$$

There are two versions to implement this update rule into an algorithm that finds the required weight vector. One is the batch, which uses the error from all the training samples to train each weight in a single learning iteration. The other is the stochastic, that uses the error computed in each training iteration to update the weights. In our case the learning samples are the strides and we do not have access to all of this information apriori. The present problem requires the system to learn online and to adapt to new, unexpected conditions. Therefore, it is not feasible to use the batch learning, instead we use the stochastic learning algorithm.

Choosing the learning rate is critical to guarantee the system's success. An ideal learning rate is defined by the following inequality.

$$0 < \mu < \frac{1}{\lambda_{max}}, \tag{3.11}$$

where $\lambda_{max}$ is the maximum eigenvalue of the correlational matrix $\mathbf{R_x}$:

$$\mathbf{R_x} = [\mathbf{x}_j\mathbf{x}_j^T] \tag{3.12}$$

An adaptive method to define the learning rate will be defined further ahead. Next, will be presented the details about the Newton's method and its application in the system.

### Newton's Method

The Newton's method is a minimization/maximization algorithm that searches a function's roots, by creating tangent approximations to its curve. It is used here to search the cost function's minimum, therefore it is applied to the cost function's first derivative. Its zeros match the cost function's minimums or maximums. Since this cost function is quadratic, there will be only minimum.

We get the weight update rule:

$$\omega_{ij} := \omega_{ij} - \frac{J'_{ij}(\omega_{ij})}{J''_{ij}(\omega_{ij})}. \tag{3.13}$$

Which when the cost function is applied we get :

$$\omega_{ij} := \omega_{ij} - H_{ij}^{-1}\nabla_{\omega_{ij}}J_{ij}(\omega_{ij}), \tag{3.14}$$

where $H_{ij}^{-1}$ is the Hessians matrix's inverse computed for the cell $(i, j)$ and $\nabla_{\omega_{ij}}J_{ij}(\omega_{ij})$ is the vector with the partial derivatives of that cell's weight vector $\omega_{ij}$. The entries for the Hessian matrix are given by:

$$H_{ij}^{(ab)} = \frac{\partial^2 J_{ij}(\omega_{ij})}{\partial\omega_{ija}\partial\omega_{ijb}}. \tag{3.15}$$

Here $H_{ij}^{(ab)}$ is the entry of indexes $a$ and $b$ to the Hessian matrix of the cell $(i, j)$. The denominator $\partial\omega_{ija}\partial\omega_{ijb}$ defines the second degree derivative according to the $a^{th}$ and the $b^{th}$ entries for the weight vector $\omega_{ij}$.

Since the cost function is a quadratic, its derivative is a straight line. This means that the newton's method will converge almost immediately, absorbing all the features in the raw data. This includes the obstacle detections, which we want to discern from the remaining signals and not absorb it as a part of the locomotion pattern - this problem will be adequately addressed in the next section. In order to smooth the learning process, a learning rate variable $\mu$ will be included in this method. So, we finally get our Newton's method weight update rule:

$$\omega := \omega - \mu H^{-1} \nabla_{\omega} J(\omega). \tag{3.16}$$

Some approaches choose to set $\mu$ dynamically in order to optimize the learning process, making it faster, while granting conversion. Here, a similar approach will be presented that seeks to answer those same requirements, as well as some specific to this problem. All the details about such mechanism will be presented in the next section.

### Adaptive learning rate

Usually, the goal of an adaptive learning rate is to achieve fast convergence while preventing the system to diverge. If set too high, the learning rate might prevent the system to converge. But, in this case, there are some extra requirements, namely:

- Prevent the absorption of the obstacles as part of the locomotion pattern;
- Enable the system to readapt if the locomotion pattern changes.

The prevision layer is required to ignore instantaneous alterations to the raw layer's usual pattern, since these alterations usually mean the presence of an obstacle. So, the differences are supposed to be detected and not absorbed as part of the pattern. On the other hand, the system is also required to adapt to new conditions that might come up, such as a change in the locomotion gait. In this case, the prevision layer should detect a consistent alteration to the pattern which is usually perceived by a persistent error. The prevision layer will adapt the learning rate every iteration according to these requirements. There are as many predictors as there are cells, so, each one requires a specific learning rate. The adaptive learning rate $\mu_{ij}$ is computed for each cell $(i, j)$ according to:

$$\mu_{ij} = \eta_{ij} - \beta_{ij}\eta_{ij}, \tag{3.17}$$

where $\eta_{ij}$ is the maximum learning rate value for cell $(i, j)$ and $\beta_{ij}$ is a variable meant to reflect the confidence on the expectations created for that cell. The value of $\eta$ to the Newton's method was chosen through a trial and error process that seeks the fastest result.

The LMS cell's'$\eta$ is chosen taking into account the inequality 3.11. The confidence $\beta_{ij}$ is then responsible to define $\mu_{ij}$ at every iteration with relation to its maximum value. The more confident the system is about a given cell's expectations, the less necessary is to change its weight vector. So, the lower $\mu_{ij}$'s value needs to be. $\beta_{ij}$ evaluates a given cell's confidence according to :

$$\Delta\beta_{ij} = -\beta_{ij} + \alpha_1(1 - e_{ij})^2\beta_{ij} + \alpha_2\varepsilon, \tag{3.18}$$

Where $\alpha_1$ and $\alpha_2$ will serve as weights such that $\alpha_1 + \alpha_2 = 1$; $\varepsilon$ expresses the mechanism's experience as a function of the number of learning iterations $n$ - the number of strides -; and $e_{ij}$ measures the cell $(i, j)$ performance. This performance is measured according to :

$$e_{ij} = \frac{1}{N}\sum_{u=n-N}^{N} d_{ij}^{(u)}, \tag{3.19}$$

where $u$ will iterate the last $N$ errors produced by the cell $(ij)$; $e_{ij}$ is computed as the average of the last $N$ errors produced by a given cell. This value tells the system if that cell's error has been systematic, or else if it usually produces good expectations. If $e_{ij}$ has an high value, most of the last $N$ learning iterations received a much different raw value from what was expected, which means that possibly the locomotion pattern has changed. But, if an obstacle detection has occurred during those $N$ learning iterations, a single different value could not change $e_{ij}$. Therefore, there is no risk in absorbing that change. The variable $e_{ij}$ helps the system to answer the mentioned requirements.

The system's experience is expressed in function of the number of learning iterations $n$:

$$\varepsilon(n) = e^{-0.1e^{-(\frac{n}{5}-10)}}. \tag{3.20}$$

The experience function is based on the Gompertz function, which behavior matches our needs. The system's experience will grow steadily only a while after the beginning of the learning process, ending in an asymptotic behavior that approaches 1, but never really reaches it. Since $\beta_{ij}$ is initially 0, the system will have no confidence in itself until it acquires some experience, which depends only in the number of learning iterations.

## Novelty

The novelty layer receives the difference $d_{ij}$ for every cell and will filter them in order to discern the obstacle detections from the lesser error values. It works similarly to a high pass filter. The filtering process uses a gain factor variable for each cell $g_{ij}$ which serves as a reputation register. This value serves as an amplification signal if a cell usually produces good expectations (low error values), or as a reduction signal otherwise. The obstacle detections are high values of $d_{ij}$, but the frequency at which these occur is not enough to change $g_{ij}$. So, it is granted that if a cell behaves good, it will have a good reputation. This cell output is given by :

$$o_{ij}(n) = (g_{ij}(n)d_{ij}(n))_{th}, \tag{3.21}$$

where $g_{ij}$ is the gain factor for the sensor $i$ and step phase $j$ and $th$ is the threshold that limits the errors. If the computed value is greater than the $th$, it will be seen as an obstacle detection. The gain factor variable needs to be updated every iteration in order to maintain a correct register of that cell's performance. Generally, this is done according to :

$$g_{ij} := gij + Fg_{ij}\alpha^{gf}, \tag{3.22}$$

where $\alpha^{gf}$ is a rate that limits how much $gij$ can be changed in a single iteration and $F$ is a function that decides how the gain factor should be updated. $F$ can be defined in many ways as long as its output reflects how the system performed during that iteration. It will increase or decrease a given cell's reputation in the interval $]0, gf_{max}]$. Two approaches were considered to define $F$: a gaussian curve and the confidence mechanism previously presented.

**Gaussian** The Gaussian function's bell shape (figure 3.3) offers the required behavior. It is defined as a function of the error, it favors a prediction close to 0 while penalizes others with higher values. The function is :

$$F(d_{ij}) = -\frac{1}{2} + e^{-\frac{(\mu - d_{ij})^2}{2\sigma^2}} \tag{3.23}$$

Where $\mu = 0$ and $\sigma = 0.1$. Also, a factor of $-\frac{1}{2}$ was added so that the function's output is within the interval of $[-0.5, 0.5]$. This limits $g_{ij}$ update to a percentage of its own value, which prevents the variable to become null.

**Confidence** As the confidence system purpose fits the requirements of this problem, it was also considered as a possible candidate. It is applied just as the same way as in the previous layer.

In the novelty layer's output, each cell dictates if there was an obstacle detection. As mentioned before during the raw layer description, the structure formed by all cells is a relation to the distance and the phase of the stride. Therefore, If one cell holds an obstacle detection value, the robot will know at what distance and during each moment of the stride that obstacle was detected. This information is relevant to the second part of the mechanism, as it will be shown.

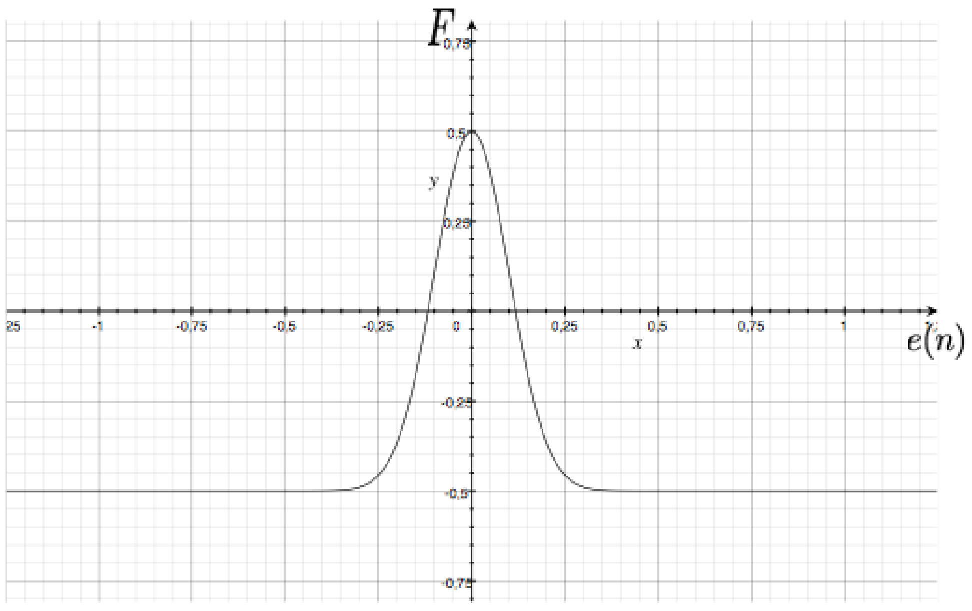

**Figure 3.3:** Gaussian function's plot used in novelty layer.

## 3.2 Learning Mechanism

The obstacle detections occur at a given distance from the obstacle - detected from one of the sensors - and during a specific moment of the stride. This spatial and temporal relation is granted by the cell structure defined in the raw layer. When approaching the obstacle, several cells will be activated, forming an Activation Pattern (AP). An AP defines a way to approach the obstacle, since the robot can approach it in more than one way, there are several possible APs.

The goal of this part of the mechanism is to create a map from the activations to changes in the locomotion. These changes will enable the robot to avoid stepping on the obstacle. The changes to the stride length will grant the robot to place its paw optimally, before stepping over the obstacle. The map is built every time the robot fails to clear the obstacle without touching it. The necessary conditions to the learning take place are the following: the robot must know how it collided with the obstacle and remember the sensed activation pattern during the approach. After enough iterations, the robot will be able to respond to an activation pattern in order to change its locomotion to achieve its goal. The following sections describe the components that carry out this task.

## Activation Cells

The activation cells receive the obstacle detections from the novelty layer. Their values will be used to compute the necessary alterations to the stride length. Each activation cell $x_{ij}^{act}$ has its value updated according to:

$$\tau^{act}\frac{\partial x_{ij}^{act}}{\partial t} = -x_{ij}^{act} + o_{ij}, \tag{3.24}$$

where the $\tau^{act}$ defines the update rate for these cells.

## Short Term Memory Cells

The Short Term Memory (STM) cells play a major role in the learning process, they are activated alongside with the activation cells. However, the STM cells's values are supposed to last until the robot steps over the obstacle, defining a memory of short duration about the activation patterns. If the robot collides with the obstacle, it recalls the activation pattern. But more on that in the next section. The STM cells $x_{ij}^{STM}$ are updated following:

$$\tau^{STM}\frac{\partial x_{ij}^{STM}}{\partial t} = -x_{ij}^{STM} + x_{ij}^{act}. \tag{3.25}$$

The convergence rate $\tau^{STM}$ has a double meaning, its value grants the correct activation of the STM cells whenever there are new activations, and gradually decrease the activation value otherwise. It is defined as follows:

$$\tau^{STM} = 1 + \frac{2}{1+e^{d}}, \tag{3.26}$$

where $d = 100(x_{ij}^{act} - x_{ij}^{STM})$. When $x_{ij}^{act} > x_{ij}^{STM}$, $\tau^{STM}$ will converge to 0, and to $x_{ij}^{act}$ otherwise. This will grant the required double effect previously mentioned.

## Weights and Learning

The weights define the map from the activations to the changes in the stride length. This component is both the target to the learning iterations, and the memory which dictates how the robot should react to the obstacle detections. First, the map needs to be built. When the robot steps on the obstacle it triggers a learning signal $\delta$ which defines the type of collision. Such signal is defined according to:

$$\delta = \begin{cases} 1 & \text{if paw placement,} \\ -1 & \text{if paw extension.} \end{cases} \tag{3.27}$$

The triggered $\delta$ depends on the phase of the locomotion and the touch sensor that detected the obstacle. Such details concern the mechanism application to the robot. However, it is required that each signal reflects the necessary alterations to the locomotion in order to avoid such situation in the future. When the robot should reduce the stride length, the learning signal should be $\delta = -1$ - the robot is in the paw extension phase of the step - and when the stride length should be increased, the learning signal should be and $\delta = 1$ - triggered during the paw placement phase of the step.

The learning iteration depends on the triggered learning signal and the activation pattern sensed during the obstacle approaching. The activation pattern is still present in the STM cells, due to those cell's ability to remember the activations. So, when the learning signal is triggered, the weights are updated. The learning rule updates a single weight $w_{ij}$ according to:

$$\Delta wij(n+1) = \delta x_{ij}^{STM}(|w_{ij}(n)| + c)\alpha^{STM}, \tag{3.28}$$

where $\alpha^{STM}$ limits how much a weight is changed in a single learning iteration and $c$ is a small constant that starts the learning process. The factor $\delta x_{ij}^{STM}(|w_{ij}(n)| + c)$ is meant to reflect how much the given weight contributed to that learning iteration. The higher this factor is, greater is the need to change $w_{ij}$, in order to avoid this situation in the future.

After each learning iteration, the weight map is normalized such that $||w_j|| = 1$, where $w_j$ is the weight vector for each stride phase. Each vector contains all the cells for each sensor $i$ and one of the stride phases $j$. The normalization avoids the over saturation of one single cell and maintains the relevance of each one according to the others.

## Stride Length Modulation

This component computes the synapse values $s_i$ for each stride phase. The computation takes the activation cells and the weight map into account. It combines the obstacle detections and the acquired knowledge during the learning process, in order to produce the necessary alteration to the locomotion.

During each stride phase $j$, $s_{ij}$ is computed according to:

$$s_j = \sum_{i=1}^{n_s} \omega_{ij}|x_{ij}^{STM}| \tag{3.29}$$

The system's output $bl$ is computed according to:

$$\tau^{bl}\frac{\partial bl}{\partial t}bl_j = -bl + s_j, \tag{3.30}$$

where $\tau^{bl}$ defines the rate of change of $bl$. This value is then passed on to the system to be sent to the CPG. Since the output is in the interval $[0, 1]$, this value might need to be amplified in order to adequately change the locomotion. But that concerns the mechanism application to a specific robot.

# Chapter 4

# Mechanism Application

In this chapter, both the environments used to test the proposed mechanism and the details about the application of the mechanism to the mentioned platforms, the sony AIBO and the Bioloid quadruped version, will be presented. The next sections describe relevant information about the following items: the sensor configuration in the robot, both the proprioceptive and the preceptive ones; the specification of how the sensor values are prepared for further system use; the definition of the step over reflex activation and execution; alteration to the locomotion; and everything else that seems relevant to include as part of the system configuration and mechanism application.

A few details about the evaluation procedure are given now due to their relevance in the process of the mechanism application to the robots. The evaluation is composed of several iterations called trials, during which, the robot approaches the obstacle and attempts to step over it. As a first approach, the controller will focus on making the fore right leg step over the obstacle. So, the obstacle will be removed after this leg has tried to step over it. In the end of each trial the robot is placed back at an initial position, at a certain distance from the obstacle, as if there were several obstacles in a continuous environment. The correct steering is guaranteed at the beginning of every trial so that the robot reaches the obstacle in the expected manner.

## 4.1 AIBO implementation

The first tests were applied to the AIBO robot, it's locomotion is modeled by a Control Pattern Generator (CPG) based in dynamical systems presented in [22]. The learning mechanism is coupled with this controller in order to change specific parameters, namely the joint oscillation amplitudes, which will enable the robot to step over the obstacle. This robot has 3 degrees of freedom in each leg, two in the hip; the swing and the flap, and one in the knee. The flap is not required in the present work since the robot will only walk forward.

### The Environment

The environment was defined as a Webots world, it includes the robot, the obstacle and a gutter, whose purpose is explained next. As it was previously mentioned, a trial comprises an encounter with the obstacle in a specific way, which includes stepping over the obstacle with the fore right leg. Therefore, it is necessary to assure that the robot reaches the obstacle in this required manner. As it was found that this robot has a considerably biased steering towards the right when moving, a gutter was included in the environment to guarantee that the robot moves only forward, thus, reaching the obstacle with its right paw.

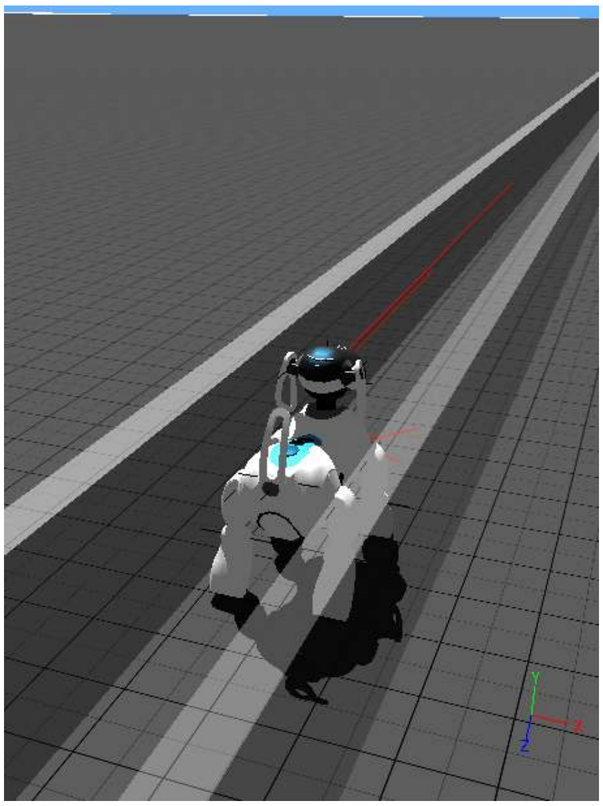

**Figure 4.1:** Environment defined for the AIBO robot. The gutter assures the robot to reach the obstacle in the required manner.

It is expected that the gutter interferes with the robot's locomotion dynamics and even with it's environment perception (the range sensor scanners), which is crucial to the learning procedures. However, as the system is trained online to deal with the range scanner inputs and to learn their activation pattern, it is expected that this interference is taken in consideration in the learning process. After all, the interference with the system happens cyclically with the locomotion, to both the robots state and it's perception inputs. Therefore, this disturbance is accounted for in the learning process instead of disrupting it.

### Sensor Configuration

There are two types of sensory information. Proprioceptive information, which defines the state of the robot during the locomotion cycle, and the robot's environment perception. The last one results from the process of scanning the ground in front of the robot, which seeks to detect unexpected objects. Also, the robot must know when and how did it touched the obstacle in order to avoid doing so in the future. Therefore, the environment perception includes touch sensors applied to the robot's paws.

In AIBO, the state of the robot during it's locomotion cycle is defined by the following items:

- The values for the hip swing joints of all four legs;
- The Touch sensors in the base of each of the robot's paws;
- And the robot's body pitch and roll values

All this information is intrinsically related to the robot 's locomotion cycle, all the values oscillate according to this periodic state. Hence, they match the requirements named in the prevision layer.

The normalization process defined in the raw layer input data is applied to the pitch and roll values only. These angles are normalized to the range of $[0, 90]$ degrees, their neutral point being defined around 45 degrees. The remaining values to the input are fed to the prevision layer without further processing.

As for the range sensors scanner, it is composed of five range sensors $n_s = 5$ in the AIBO robot. Each of which is placed in the robot's right side of the chest (figure 4.2) (close to the right fore leg). Together, the sensors scan the ground in front of the robot with an aperture of 40 degrees, each one is placed with an angle of 10 degrees from its neighbors (figure 4.2 c). The first sensor, the one pointing closer to the robot's paws, is placed at 30 degrees from the robot's coronal plane (figure 4.2 b). So, as the sensors are placed at 14.5 centimeters above the ground, the scanned length is approximately 23.2 centimeters (figure 4.2 a).

In order to return the range sensors activations, Webots uses a lookup table to compute the sensors readings according to the actual existing distance an induced noise. Table 4.1 presents the values used in the process. The first column dictates the real existing distance, the second determines the returning value and the last column shows the applied noise. The values not specified in the table are computed through linear interpolation.

The robot also receives input from the paw's interactions with the environment through the touch sensors. There are three for each paw, one placed in the base (figure 4.5), which value is included as part of the robot's state during the locomotion cycle; it also detects the executed stance phases. Other two are placed at the fore and back sides of each paw (figures

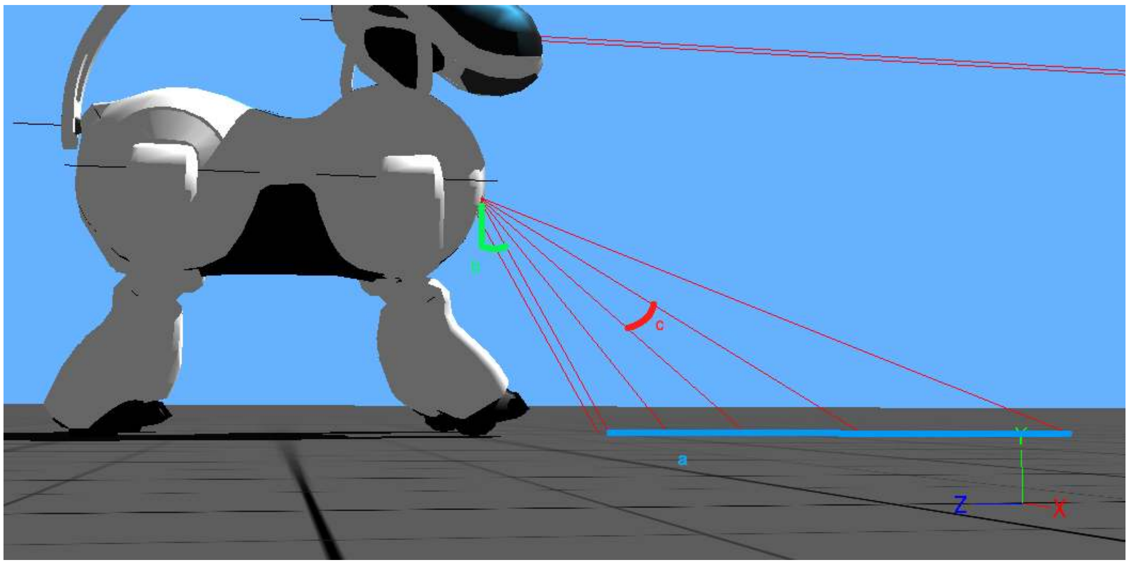

**Figure 4.2:** AIBO's range sensor configuration. These sensors scan the ground in front of the robot. The distance in (a) is the scanned distance,(b) is the angle of the first sensor ray according to the robot's coronal plane, and (c) is the angle between two range sensor's rays.

**Table 4.1:** Values acquired from the range sensors (column 2), according to that sensor's distance to a surface (column 1) and the applied noise(column 3).

| Actual Distance | Sensor return value | Noise |
|---|---|---|
| 0 | 100 | 0 |
| 0.1 | 100 | 0 |
| 0.5 | 500 | 0.1 |
| 0.9 | 900 | 0.2 |

4.3, 4.4 respectively). Both touch sensors are used to detect collisions with the obstacle during the step over reflex, their activations will give rise to learning signals according to the triggered sensor. As described in chapter 3, the paw's fore touch (figure 4.3) sensor will trigger the paw extension learning signal $\delta = -1$ so that the stride length is reduced in future similar situations. If the the back touch sensor (figure 4.4) is activated, the paw placement learning signal $\delta = 1$ is created in order to increase the stride length in similar situations in the future.

### Step Over Reflex Triggering

In order to clear the obstacle without triggering any learning signal, the robot must place it's paw in the optimal spot before executing the step over reflex. These are the ideal conditions that will enable the robot to avoid the obstacle. The reflex is modeled by increasing the amplitude for both the hip-swing and knee joints, during the swing phase of that step. The

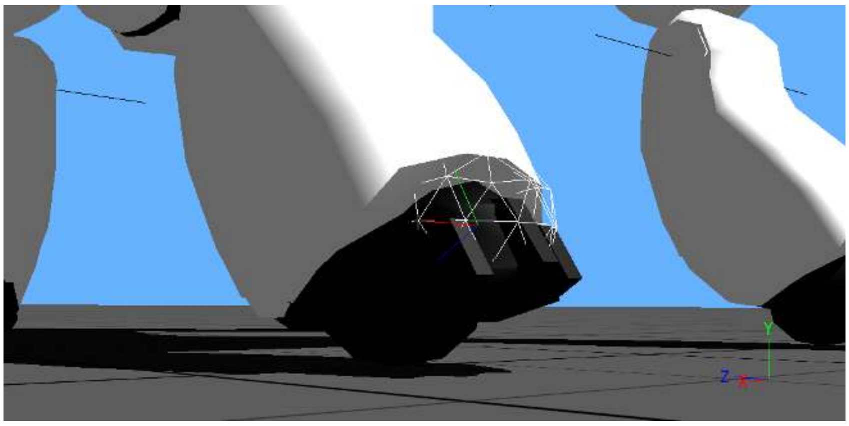

**Figure 4.3:** The visible polygon in the figure defines the fore touch sensor's bounding box, that is applied in the fore right paw in the AIBO robot.

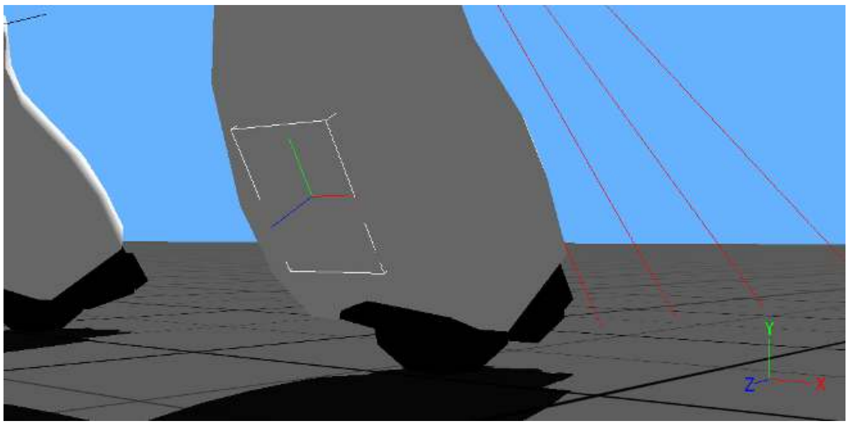

**Figure 4.4:** The back touch sensor is presented in the figure as a polygon in the AIBO's paw.

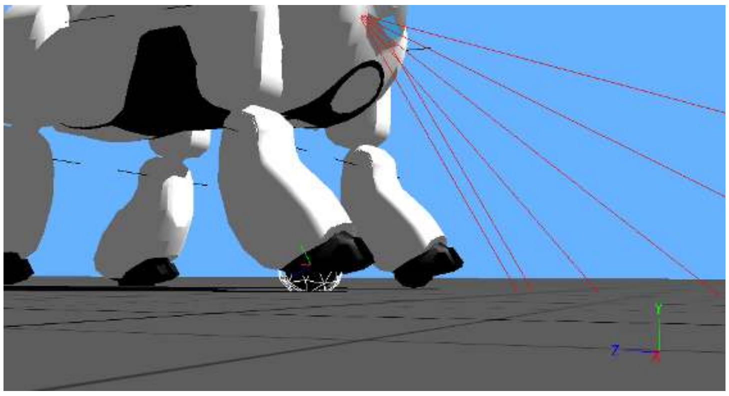

**Figure 4.5:** The polygon under the AIBO's paw defines the bounding box for the base touch sensor.

reflex is only applied to the leg on which the reflex is being executed, in this case the fore right leg.

Regardless of the ideal conditions are met or not, the step over reflex is activated at the correct moment and at a minimum distance from the obstacle. This occurs after a minimum distance between the robot's paw and the obstacle is reached, at that moment the reflex is activated. When the correct stride phase becomes active, the step over reflex increases the amplitudes in the CPG and the robot tries to step over the obstacle. If it fails, a learning signal is triggered as previously described, if not, the system has already learned to avoid the obstacle under these approaching circumstances.

The minimum distance from the robot's paw to the obstacle is given by $kbl$, where $bl$ is the usually observed stride length, and $k$ is a constant meant to reflect how bigger the reflex stride is when compared to the normal stride. The constant is set to $k = 1.3$. $bl$ is measured and fed to the robot at all times. The robot increases it's knee amplitude by $k_{rk} = 50$ and it's hip-swing amplitude by $k_{rs} = 50$. These values were chosen through a trial and error process.

### Locomotion Adaptation

In order to achieve its goal, the mechanism changes the locomotion by increasing or decreasing the stride length at all times. However, this information is in the weight map value space and needs to be transformed in order to be appropriate to change the CPG and then the robot locomotion. This is achieved by amplifying the synapse's output before the CPG parameterization. The parameters that require changing are the hip-swing amplitudes of all legs, which will generate different stride lengths.

### AIBO Gaits

The locomotion gaits are an important point in the success of this learning mechanism, alongside with the robotic platform itself. Some locomotion gaits will be now evaluated according to their stability, in order to choose the best fit to execute and test the proposed mechanism. It is required that the gait offers a stable locomotion and that the activation pattern of it's sensors is maintained through time. I.e., the activation patterns verified during the locomotion cycle are repeated every stride.

Figure 4.6 displays two sets of bars, the full height ones shows the activation of the touch sensors in each of the paws and the others show the desired stance phases for all legs, according to the gait being executed. The first one is expected to reflect the stance phase executed in the specific gait and how well it was executed. By comparing both, one can get an idea of how well is that gait being executed.

Evaluating the AIBO's walk gait (duty factor and phase relation, both set to 0.75), a certain fragmentation is visible in the hind legs touch activation. This is caused by the bouncing effect of the locomotion at the beginning of the stance phase. Also, a misplaced activation of the touch sensors according to the expected stance phase is clear, this is the result of the delay in the actuators when compared to the CPG modulation. Besides these

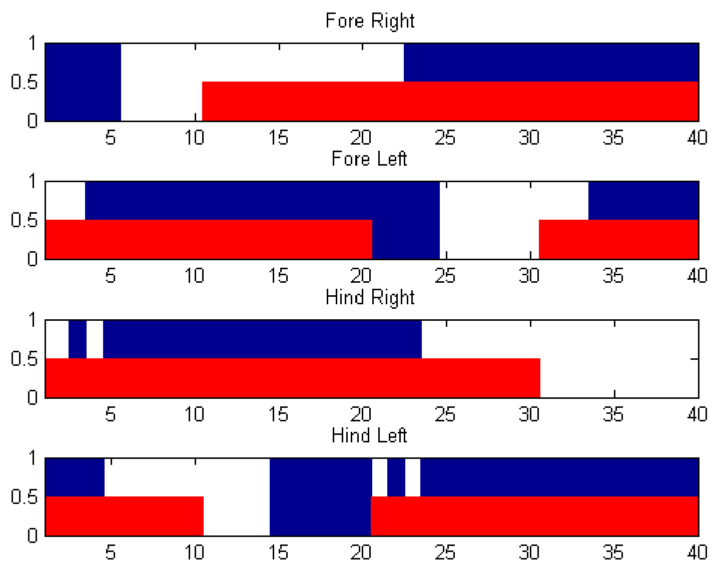

**Figure 4.6:** Activation of the base touch sensors in AIBO (full height plot) during the walk gait, and the desired stance phases for the same gait (half height plot).

features and the duration of the stance periods, there is a reasonable difference between the expected stance phases and the touch activation, specially in the hind legs.

In addition to the incorrect execution of this gait, the observed activations are not maintained through time. Figure 4.7 shows several similar graphics where the base touch sensor are compared to the desired stance phases. Each one was acquired in different strides, in sequence, starting on stride 145 and ending in stride 148. Mainly in the fore and hind left legs (the two graphics in the center of each image) there is a considerably difference in the base touch sensor activation. This denotes a different duration and fragmentation of the executed stance phase, which will result in changing perception patterns acquired by the range sensors. Therefore, this gait does not offer the required stability to the learning mechanism since the locomotion pattern is not maintained.

Another walk gait was tested with the duty factor set to 0.8 and the phase relation set to 0.75. In this case the activations were maintained and the locomotion granted better stability, however, the locomotion speed was very low. Figure 4.8 shows the activation pattern for this gait. Again, we can see the shift caused by the actuators delay and the robot's locomotion dynamics, and the difference in the length of the active periods. Still, this does not pose a problem to the prevision layer, since it is capable of acquiring the range sensor's activation pattern during the locomotion cycle, without any specific requirement exept from being cyclic and maintianed through time.

Although the AIBO's physical and dynamic properties do not suggest the correct execution of a trot gait, it was nevertheless tested in order to evaluate the stability and the

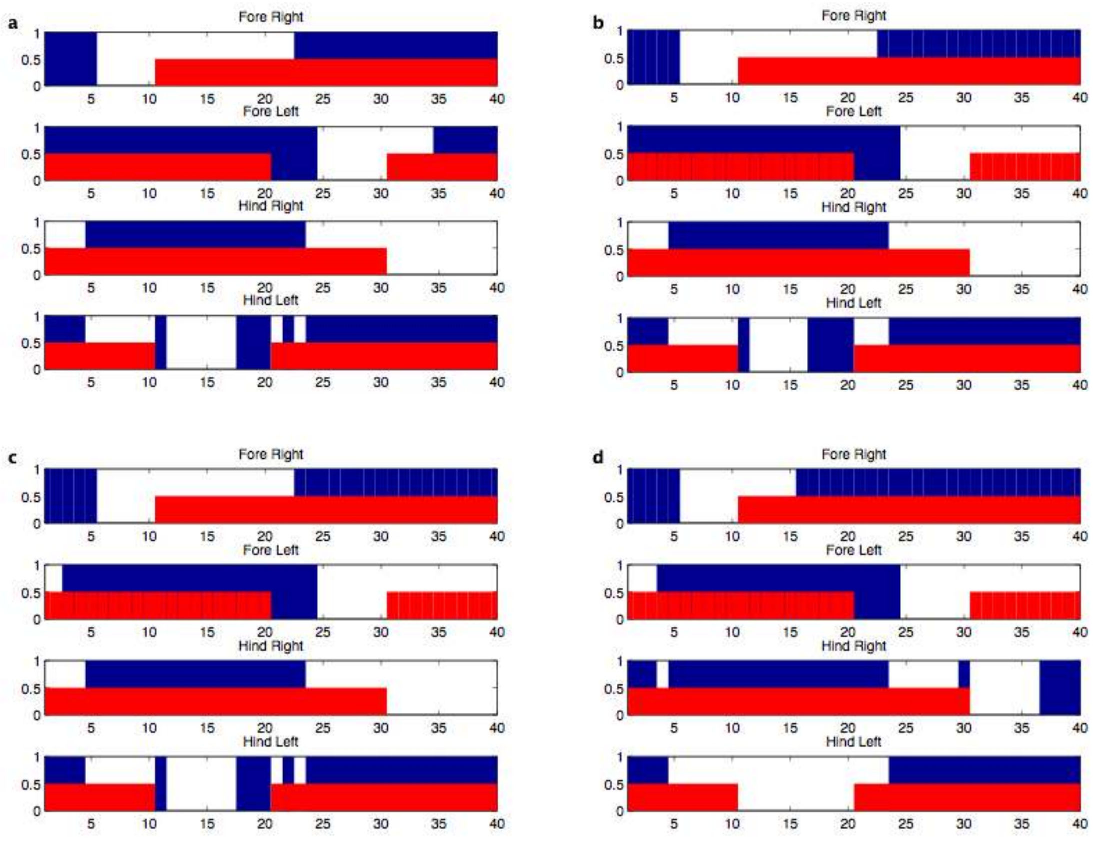

**Figure 4.7:** Evaluation of the walk gait in four different strides, from stride 145 to stride 148. Each one displays the difference between the expected stance phase and the base touch sensors' activations.

applicability to the learning mechanism. Figure 4.9 shows the results. It is noticeable that the fore sensors behave closely to what is expected, only biased by the usual shift. However, the hind legs have much worse activations.

Unlike the normal walk gait, which activations had substantial changes with time, this gait offered locomotion stability and greater speed. Therefore, this was the chosen gait to test the mechanism.

## 4.2 Bioloid Implementation

The second robot used to test the proposed mechanism was the quadruped version of the Bioloid. This second platform was chosen after the AIBO robot was revealed not to be fit for the purpose, due to it's inadequate physical and dynamic properties (big, blunt paws that made hard for it to clear the obstacle, and the difficulty achieving stable locomotion gaits).

The Bioloid has the same number of degrees of freedom as the AIBO, three for each leg 4.11, the swing and flap in the hip, and the knee. As the robot will only be walking forward

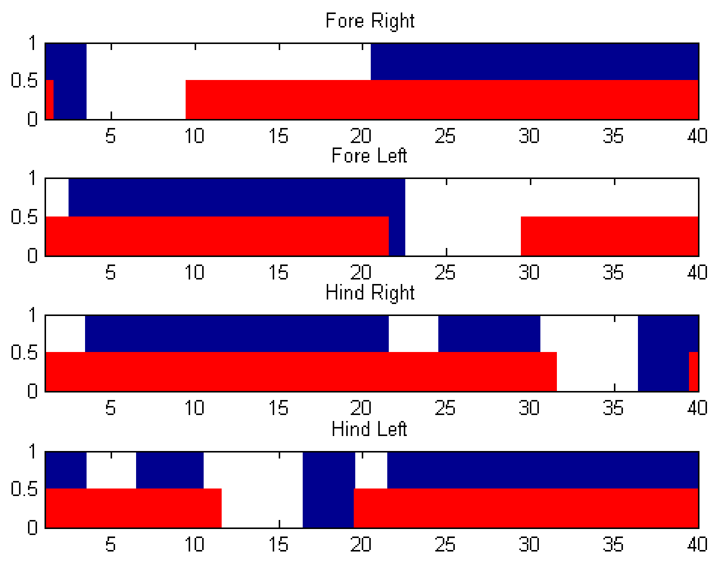

**Figure 4.8:** Comparinson between the touch sensors' activations and the desired stance phases, during the slow walk gait.

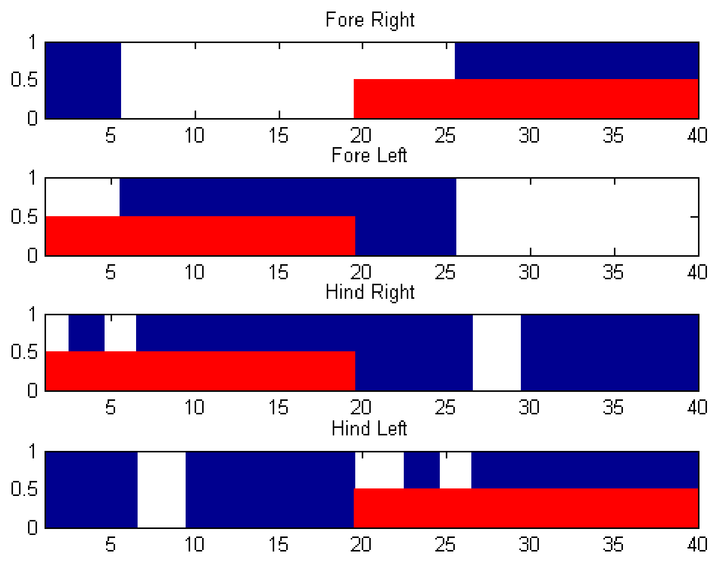

**Figure 4.9:** AIBO robot trot gait's results, desired stance phases and acquired base touch sensors activations.

there will be no need to use the flaps. This robot has thinner and taller legs, which enables it to better step over obstacles and to easily obtain stable gaits. The locomotion generator is the same CPG used in the AIBO, only with different settings that better fit this robot physical and dynamical properties.

### Environment

The environment designed to the Bioloid robot is composed only of the robot and the obstacle. Since the locomotion is stable enough not to steer out of the predefined path (at least not within the distances being considered), there is no need of a gutter to force the robot's movements. So, the robot simply heads for the obstacle (figure 4.10) at each trial.

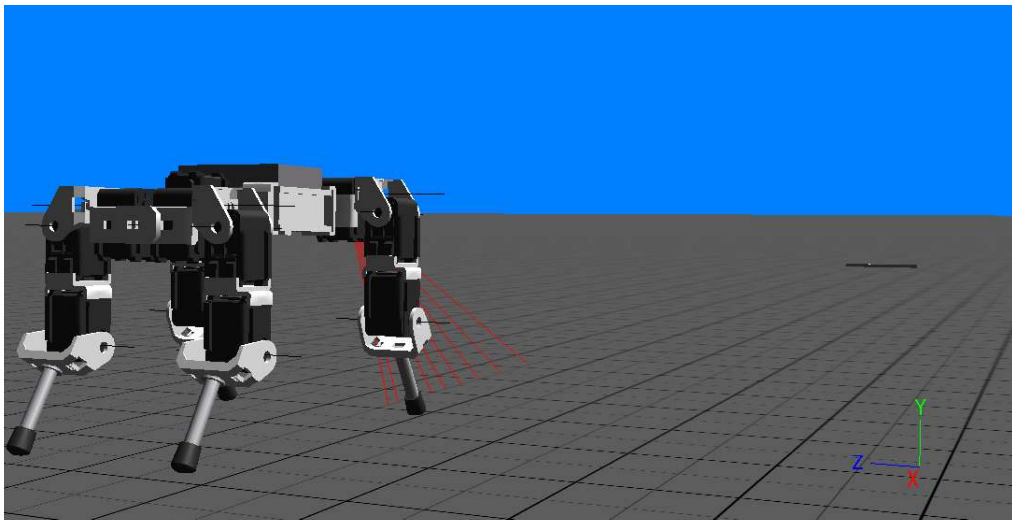

**Figure 4.10:** Environment created to the Bioloid's evaluation.

As the collisions with the obstacle are expected, since it is a part of the learning process, when a new trial begins, the steering is corrected, granting the desired direction towards the obstacle.

### Sensor Configuration

The sensor inputs in this robot include the range sensor, which scan the ground in front of the robot, the proprioceptive sensors that define the robot's state according to the locomotion cycle, and finally the touch sensors that sense the ground and the obstacle in contact with the robot's paws. The sensory inputs are similar to the AIBO's only it's configuration and number is different.

The Bioloid's state during it's locomotion cycle is defined by the following components:

- The values for the hip swing joints of all four legs;
- The Touch sensors in the base of each of the robot's paws;

There are no normalized values in the raw layer to this robot. Both the hip-swing joint values and the base touch sensors are fed to the prevision layer unnormalized, the only limits are the physical ones imposed imposed in the robot.

The range sensors in Bioloid follow the same configuration as in the AIBO. They were placed in the robot's right side of the chest, close to te right fore leg. The Bioloid scans the ground ahead of it with 10 range sensors ($n_s = 10$), which enables it to detect the obstacle from a greater distance and to start adapting earlier. Also, this offers a higher capacity in the weight map, which being bigger, it is able to better absorb the detections and the necessary alterations to the stride length. The first sensor is configured with an angle of 22.5 degrees from the robot's coronal plane and each sensor is placed with an aperture of 5 degrees from each other. The 10 sensors create a total aperture of 67.5 degrees. As the sensors are placed at 16.4 centimeters above the ground, the total scanned distance is about 32.8 centimeters.

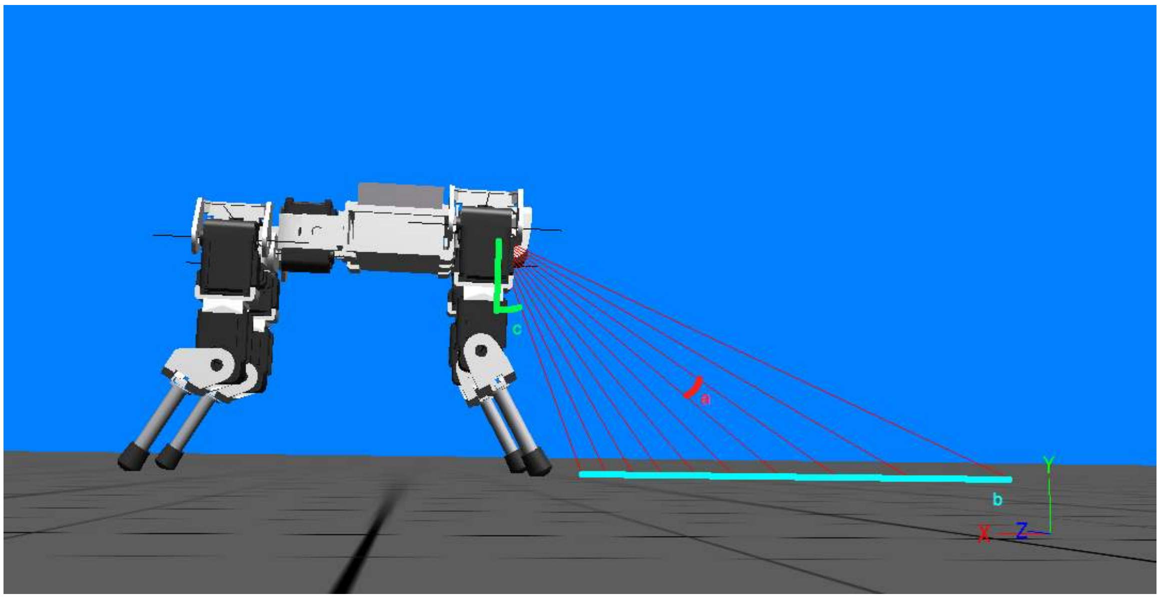

**Figure 4.11:** Bioloid range sensors' configuration to scan the ground. The scanned distance (a), the aperture angle from one sensor to another (b) and the angle between the first sensor and the robot's coronal plane (c).

Table 4.2 presents the values used by Webots to compute the range sensor readings. The first column shows the actual existing distance from the sensor to a surface, the second column shows the values returned by the sensor and the third defines the noise factor induced in the sensor readings.

The touch sensors are responsible for triggering the learning signals at time of collision with the obstacle. They are placed in each paw in the front and in the back. Also, there is a base sensor placed in the every paw that is used to sense the ground during the stance phase, or the obstacle during the step over reflex.

The activation of the back sensor 4.13, when the robot is placing it's paw after stepping over the obstacle, will result the triggering of a learning signal $\delta = 1$. This will cause the mechanism to try to avoid this situation in the future by increasing the stride length in the steps prior to the encounter with the obstacle. In order to trigger the other learning signal

**Table 4.2:** This table defines the values acquired from the range sensors in the Bioloid robot (column 2), according to that sensor's distance to a surface (column 1), and the induced noise(column 3).

| Actual Distance | Sensor return value | Noise |
|---|---|---|
| 0 | 100 | 0 |
| 0.1 | 100 | 0.01 |
| 0.5 | 500 | 0.1 |
| 0.9 | 900 | 0.2 |

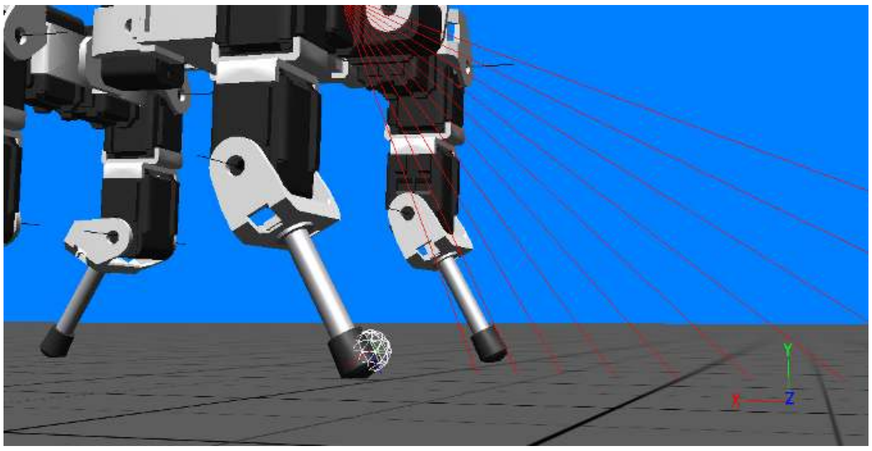

**Figure 4.12:** The polygon defines the touch sensor bounding box, which detects the collisions with the fore side of the paw.

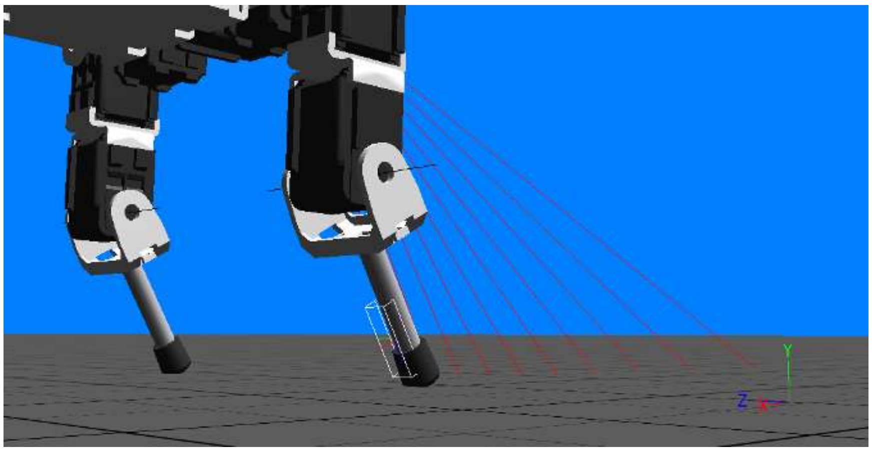

**Figure 4.13:** The back side of the paw collisions are detected with the back touch sensor, which bounding box is visible as a grid polygon.

$\delta = -1$, the fore touch sensor 4.12 is activated, or else, the base sensor (figure 4.14) is activated during the swing phase of the step over reflex. After the leg has lost contact with the ground, the base touch sensor becomes inactive, then, it is monitored in order to detect any activation during the swing phase. If any activation in this sensor occurs during this

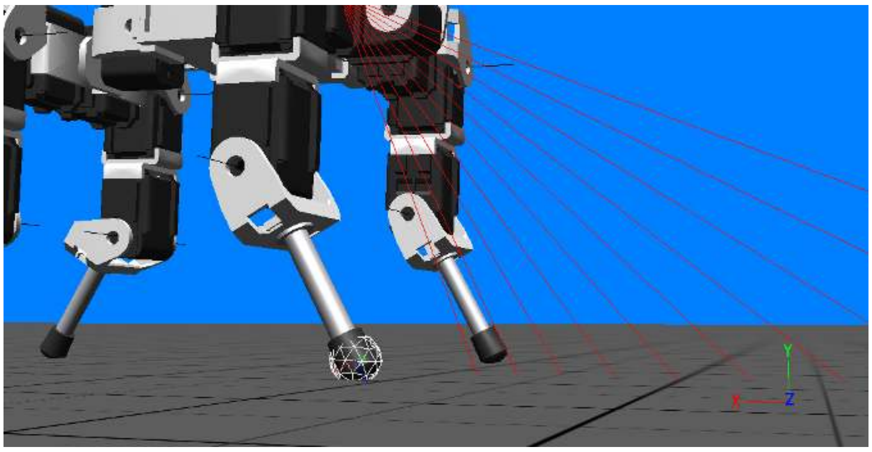

**Figure 4.14:** The base touch sensor is displayed in the figure as a grid polygon, which defines the sensor's bounding box.

period, the sensor will be activated prematurely, before the stance phase when the sensor is expected to be reactivated. This means that the robot touched the obstacle while trying to step over it. In this situation the robot will try to avoid repeating such situation by reducing the stride length in future similar situations.

There can only be one learning signal, a mechanism based on a cool down variable prevents the triggering of this reflex in a row in order to maintain consistency.

### Step Over Reflex Triggering

The step over reflex is triggered based on the distance from the paw's end effector to the obstacle. This information is fed to the robot at all times. If this distance becomes lower than the expected length of the step over reflex, this reflex becomes activated. The comparison is based on the length of a normal stride $bl$, and a correlation factor $k = 1.3$, this value was chosen through a trial and error phase.

In order to model the reflex, the amplitude of both the swing and knee joints is increased during the swing phase of that stride. The knee amplitude is increased by $k_{rk} = 60$ and the hip-swing amplitude is increased by $k_{rs} = 200$. The reflex continuous triggering is avoided through a cool down variable.

### Locomotion Adaptation

The alterations in the stride length are defined according to the output from the system and an amplification factor of $k_{bl} = 200$. This factor will transform the mechanism's output in appropriate values to be passed on to the CPG, as alterations to the amplitude for the hip-swing joints of all legs. That will cause the stride length to be altered so that the robot approaches the obstacle in the required manner.

### Bioloid Gaits

Some gaits were tested in the Bioloid platform in order to choose the best fit for the learning mechanism. The walk gait with the duty factor and the phase relation set to 0.75, a slow walk gait using a duty factor set to 0.8 and a phase relation of 0.75 and the trot gait, with both parameters set to 0.5.

Figure 4.15 shows the activation of the touch sensors with relation to the desired stance phases, during the performed walk gait. It is clear that they do not match, there is a considerable difference between the ideal stance phases and the actual touch sensor activations. However, the activation are continuous and sufficiently stable. Furthermore, there are no alteration to this pattern during the executed test, as it happens in the AIBO's walk gait. This grants a stable locomotion and perception of the environment through the range scanners. Thus, enabling the learning and the creation of reliable hypothesis and changes to the locomotion.

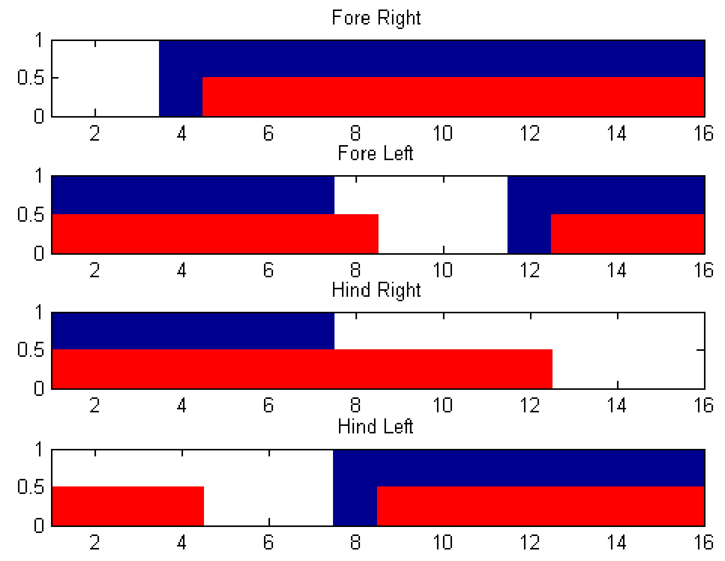

**Figure 4.15:** Bioloid touch sensors' activations and desired stance phases according to the performed walk gait.

The fore legs show an activation pattern that closely resembles the desired stance phase, on the other hand, the hind legs show an activation of a shorter duration than what was required for this gait.

The slow walk gait 4.16 shows similar results to those of the normal walk gait. The fore legs have a stance phase similar to the ideal one and the hind legs have an activation of shorter duration with relation to the expected stance phases.

The stability of this gait is also maintained throughout the execution, the main difference to the normal walk is the slower speed.

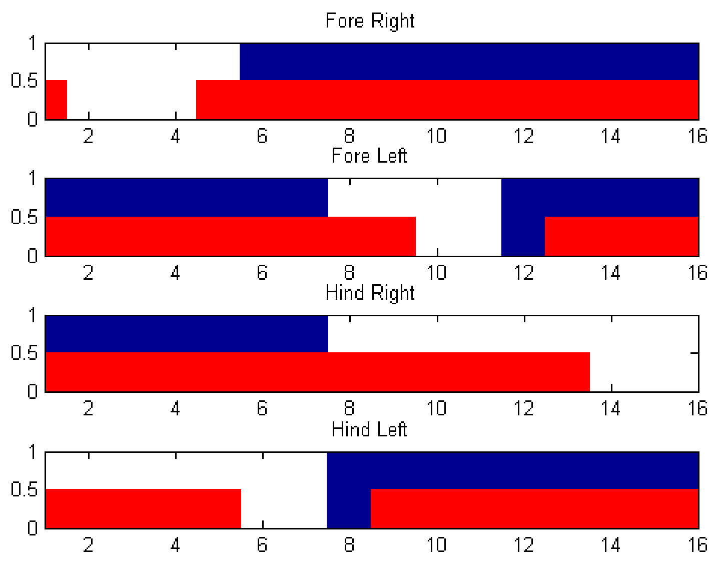

**Figure 4.16:** Leg base touch sensors activation and the desired stance phases during the slow walk gait performed by Bioloid.

The trot gait was also tested 4.17, the fore leg activations show an activation similar to the desired one, but the hind legs seem to be almost constantly supporting the robot. There is not any alteration to this pattern through time, however, there was a substantially biasing of the robot's steering towards the right, even when crossing short distances. This means that when applied, the robot steers out of the desired path.

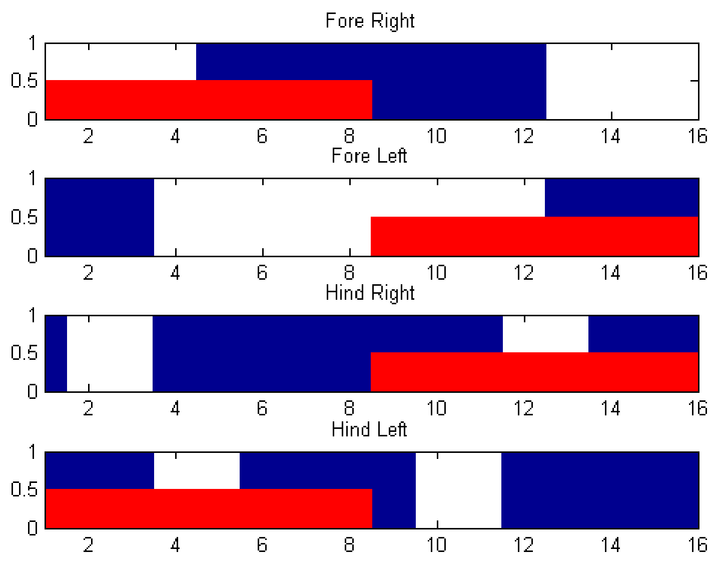

**Figure 4.17:** Bioloid executing trot gait, figure displays the desired stance phases and the acquired activations for the legs' base touch sensors.

The Bioloid's locomotion gait to be used in the tests was the normal walk, it was the most stable and fast gait, therefore the more adequate to the learning mechanism. ïż£

# Chapter 5

# Results

This chapter presents a series of tests designed and executed in order to evaluate the mechanism's capabilities in fulfilling the proposed goals. As the complete mechanism is composed of several elements, each one will be evaluated at a time. Thus, assuring the correct functioning of the complete mechanism. Also, both the considered platforms will give rise to different tests and results.

As mentioned before, the mechanism is divided in two main parts. The first one focus on the obstacle detection and the second in learning the required changes to the locomotion. Each one will be addressed in different sections.

## 5.1 Detection Mechanism Validation

The first part concerns the detection of obstacles and the mechanism's capability to adapt to new conditions in runtime, such as a change in the locomotion gait. All elements will be evaluated according to their specific goal and to their contribution to the results expected in this part of the mechanism.

The evaluating points considered during this section are the following: the stride division and the relation that exists between the robot's state and its environmental perception; the creation of reliable hypothesis and how can they be used to detect the obstacles and to adapt to new conditions; and finally, the filtering of the undesired noise from the obstacle detections.

Although both platforms were evaluated according to the defined items, the results are very similar. So, the mechanism's evaluation will be mainly presented according to the Bioloid robot. The results obtained with AIBO will be shown in a resumed version, highlighting any difference that might exist between the two applications.

### Stride division

This fist part of the mechanism treats the input raw layer from the sensors, both proprioceptive and perception sensors, in order to produce the desired structure presented in chapter 3.

The main concern in this process is the clustering of raw, noisy data into discrete and predefined chunks that divide the stride into $n_p$ equal parts. This process affects all the sensors (except the touch sensors used to detect the obstacle collision) and produces a single value that captures all the relevant information acquired during that period of the stride. Figure 5.1 shows the input and output of this process in the Bioloid robot.

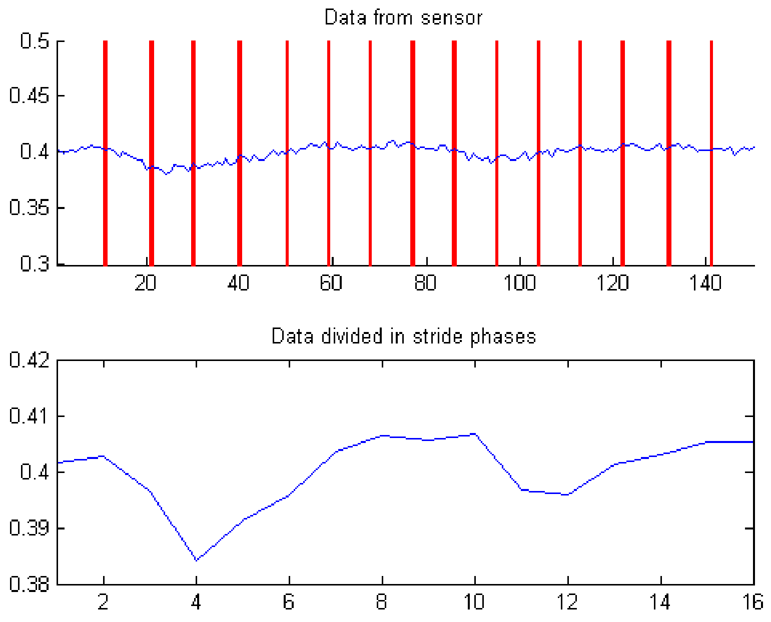

**Figure 5.1:** Stride division process, upper graphic shows the raw data, directly from the range sensor number 7 on Bioloid, and below is shown the data after processing. Stride is divided in 16 $n_p$.

It is clear that the raw data from the sensors is very noisy and presents many oscillations. This could complicate both the learning process in the prevision layer and the novelty layer filtering. The excessive noise could give rise to false detections, or define a bad reputation for a very noisy cell.

So, this process smoothens the raw data, making it more reliable, while creating the required data structure that will be used throughout the mechanism (the division of the range sensor values in the $n_{phases}$. Figure 5.2 shows the same process to the AIBO robot.

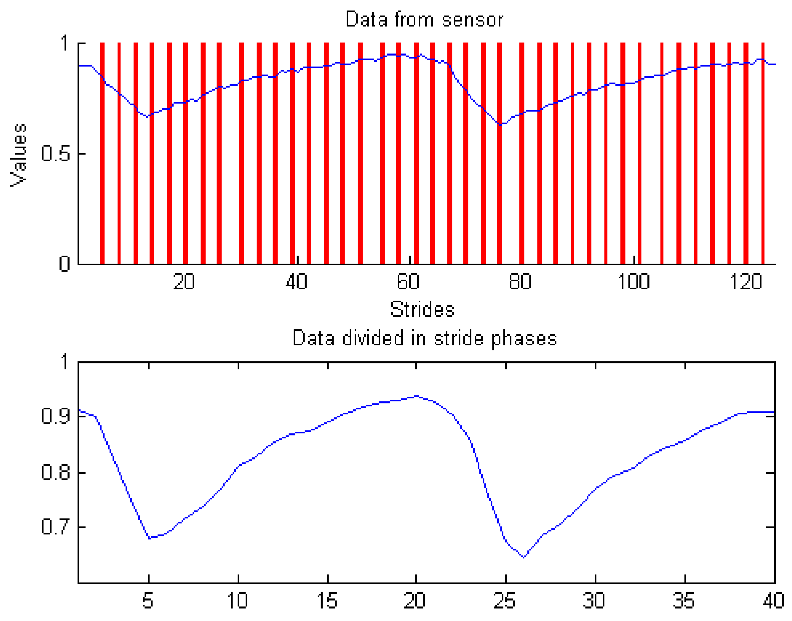

**Figure 5.2:** Aibo's stride division process with 40 divisions, applied to the range sensor pointing further away from the robot.

### Locomotion cyclic relation

It is important to show the cyclic relation between the perception data and the locomotion cycle, since this relation is what makes possible for the system to create expectations about the range sensor values. All the used sensor information, the perception and proprioceptive data, varies with the robot's locomotion cycle.

The expectations for the range sensors' values of each cell are created according the state of the robot in its locomotion cycle, which is defined by the proprioceptive information (described in the implementation chapter 3). Figure 5.3 displays this relation in the data. The upper plot shows the activation of the range sensor 7 during 3 strides and bellow the fore right hip-swing joint values during the same 3 strides. One can see that there is a cyclic behavior in both plots and that both oscillate with the same period.

We can find the same relation between the same types of sensor modalities in the AIBO, figure 5.4. The upper plot shows the activation of the range sensor is the number 5, the one pointing further away from the robot, and the lower plot shows the values for fore right hip-swing during that same period.

### Prevision capabilities

The prevision capabilities seek to enable the mechanism to detect obstacles and to adapt to changes in the locomotion pattern. So, two evaluation scenarios were created in order to

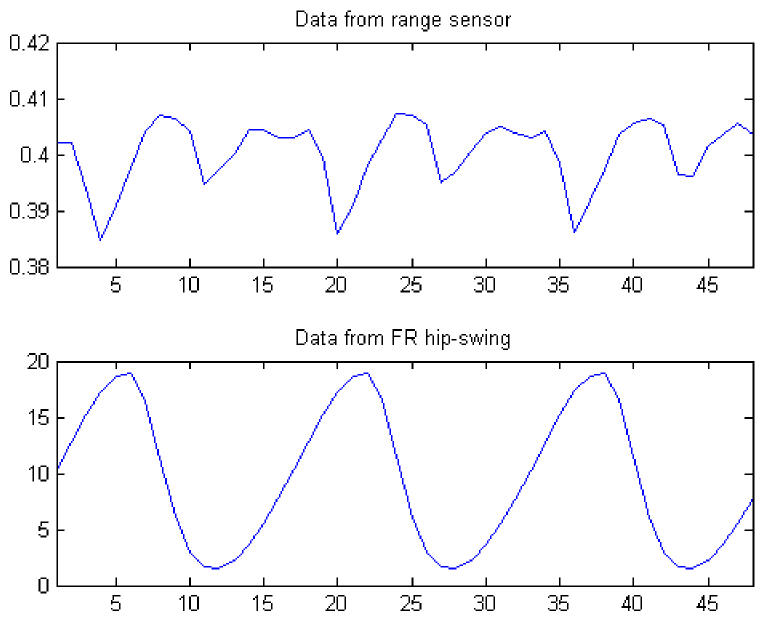

**Figure 5.3:** The relation that the range sensors and the robot's state have with the locomotion cycle. Upper plot shows the range sensor activation from stride 20 to 23 and below the fore right hip-swing joint value during the same strides.

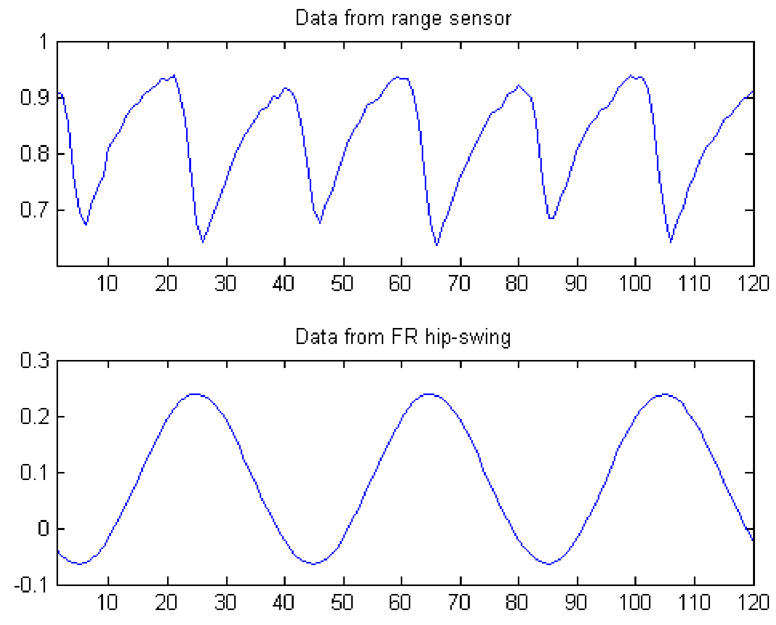

**Figure 5.4:** Activation of the fore right hip-swing joint value (lower plot) and the range sensor 5 in the AIBO robot, during the strides 30 to 33.

evaluate the mechanism's response to each feature. On the first the robot will encounter several obstacles in its way, and will move with the previously defined optimal gait. The other scenario will not include obstacles, instead, the robot will have its locomotion gait changed within predefined periods of time.

First, the prevision capabilities for both the Least Mean Square (LMS) and the Newton's method, will be evaluated without the adaptive learning rate mechanism presented before. This will show the behavior of both methods without further enhancement, highlight the problems that arise by using such type of approach in these situations.

The first mechanism to be evaluated will be the LMS, this learning rule will use a static learning rate $\mu$ defined according to the inequality 3.11. The value is computed at the beginning of every test during the first iterations, according to:

$$\mu = \frac{1}{100\lambda_{max}}, \tag{5.1}$$

where the maximum eigenvalue $\lambda_{max}$ is computed according to the correlational matrix (3.12) of the first 10 acquisitions.

Figure 5.5 shows the performance achieved with this learning rule, using the optimal constant learning rate. The presented signals include the raw values acquired by the sensors, the hypothesis created for those values through time and the difference, or error, between them.

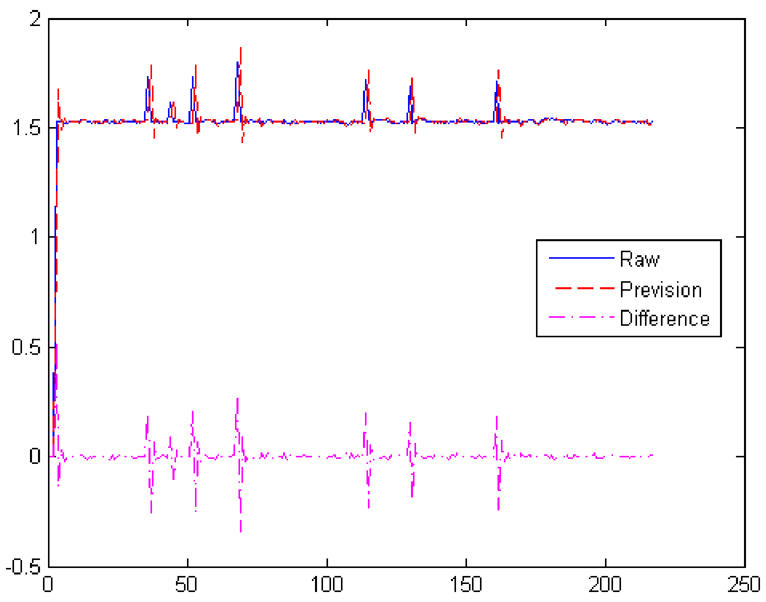

**Figure 5.5:** Performance of LMS with constant learning rate.

The previsions show that the system quickly converges to the raw data in each iteration. However, after each peak in the raw layer input, which is caused by an obstacle detection, the prevision layer simply absorbs those values as if they were a part of the locomotion pattern. This produces two high error signals instead of one, since the mechanism expects higher error values after the obstacle detections.

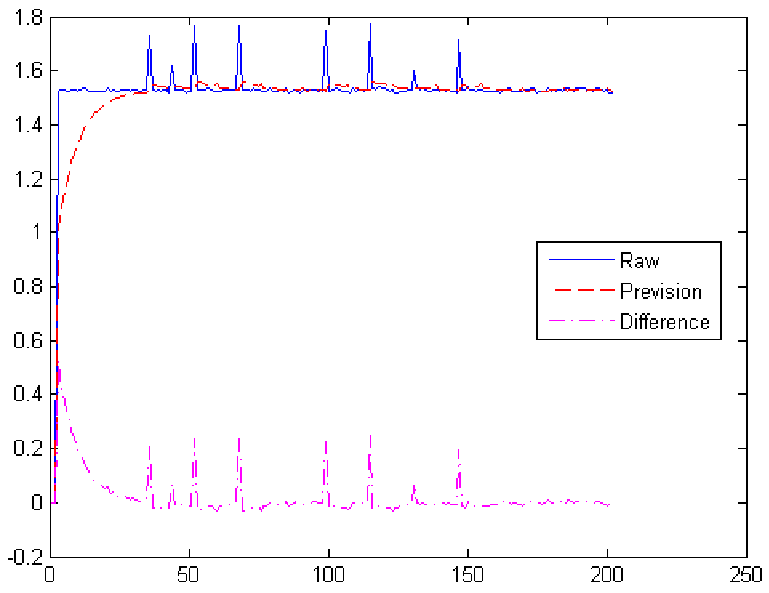

**Figure 5.6:** LMS response with a slow constant learning rate behavior, facing several obstacles in its path.

If the learning rate is set to lower values :

$$\mu = \frac{1}{1000\lambda_{max}}, \tag{5.2}$$

we get a different behavior, the mechanism slowly converges to the required values. Figure 5.6 displays such behavior and the mentioned problem is no longer present. However, there is a substantially slower convergence if the same constant learning rate is applied to the varying locomotion gaits problem. These results are presented in figure 5.7, the alterations in the locomotion gait are applied every 30 strides and the results are clearly visible in the raw data. The robot switches between normal walk, slow walk and trot, and the mechanism is required to answer these alterations in order to keep producing good expectations.

The second regression model is the Newton's method, which is a minimization algorithm applied here to the cost function's minimization. It usually does not include any learning rate, therefore it is naturally set to $\mu = 1$.

Figure 5.8 shows the behavior of such algorithm when applied to detecting obstacles. In these conditions the method is of little use, since the prevision values simply are the same as the raw, only with a stride's delay.

In order achieve a smoother convergence, the method's learning rate was defined to $\mu = 0.5$. Figure 5.9 shows the results for the obstacle detection evaluation scenario under these circumstances and it is clear that the obstacle detections are still absorbed by the prevision

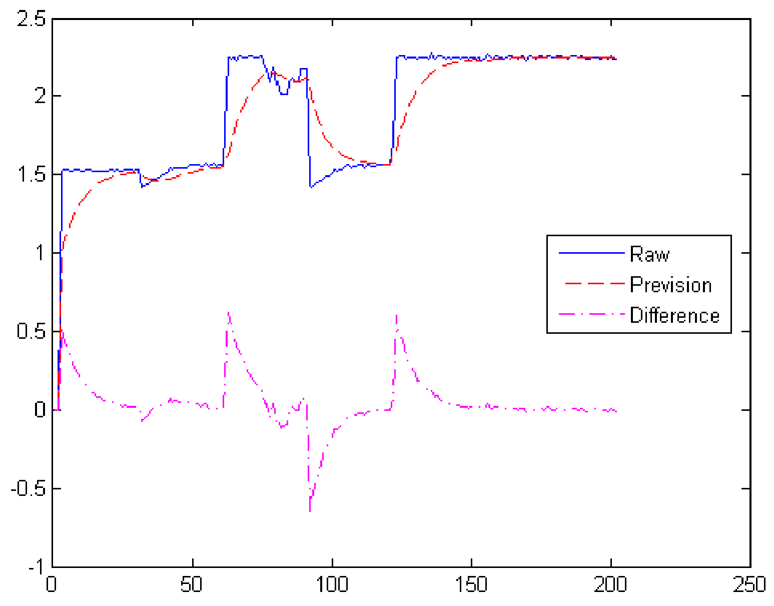

**Figure 5.7:** LMS response without adaptive learning rate behavior facing changes in the locomotion gait.

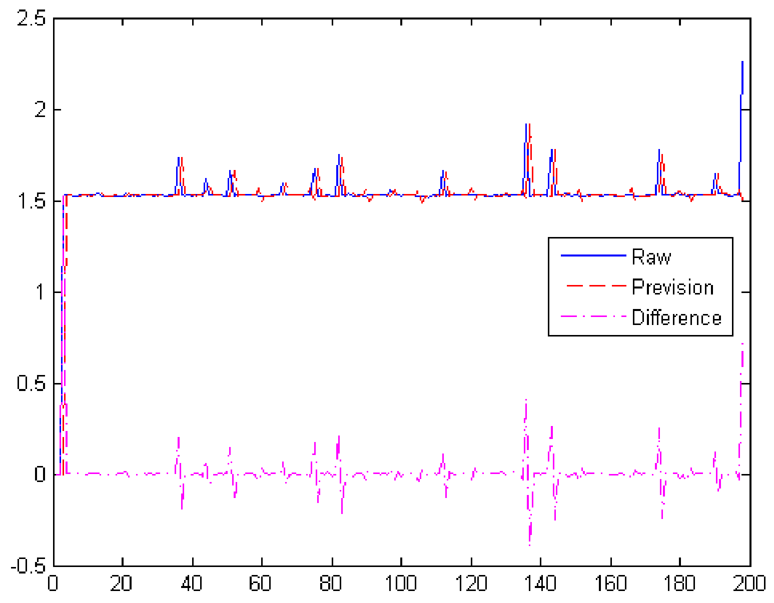

**Figure 5.8:** Newton method's previsions obtained in Bioloid without learning rate.

layer. In the changing activation pattern scenario the results were not good either, since the convergence became slower. If we reduce the learning rate even further, we might obtain a good result in the obstacle detection scenario, but we would get a much worse results in the changing locomotion pattern scenario (figure 5.10, displays the results when $\mu = 0.125$).

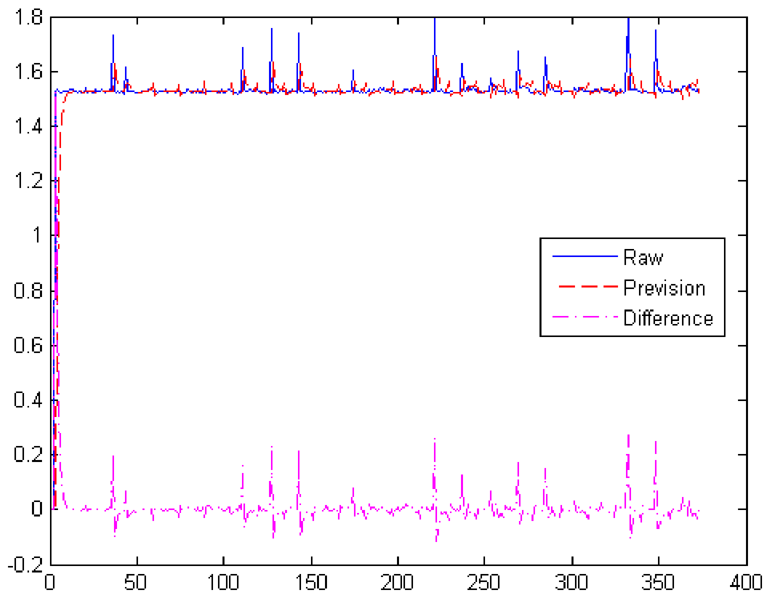

**Figure 5.9:** Newton's method results a static and low learning rate in Bioloid.

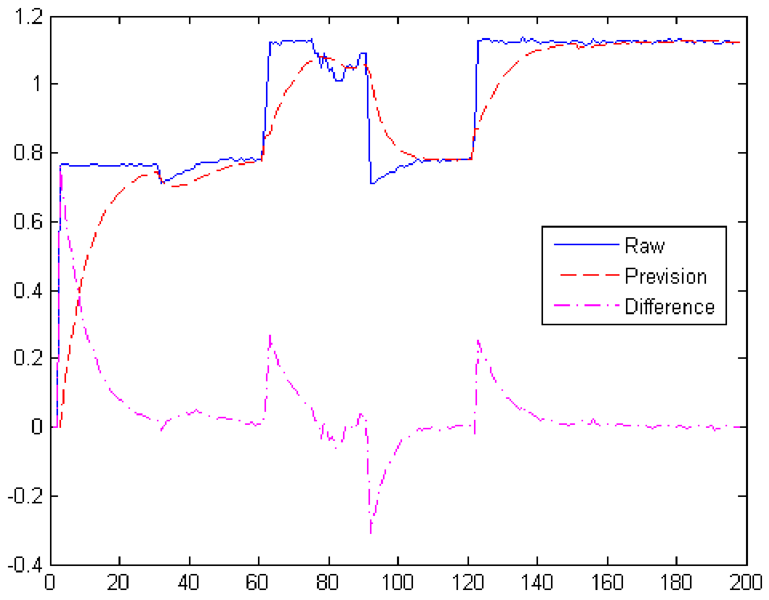

**Figure 5.10:** Newton's method behavior when facing changes in the locomotion pattern with a slow learning rate.

A static learning rate does not seem to meet the requirements pointed out before in chapter 3. So, the adaptive learning rate mechanism was defined in order to answer the learning requirements.

### Adaptive Learning Rate Results

The adaptive learning rate is defined according to a percentage of the optimal value, which poses a limit to the maximum learning performance. The LMS optimal learning rate is the one mentioned previously, the one used as static learning rate in the previous section.

The LMS method with the adaptive learning rate was first evaluated in its obstacle detection capability, figure 5.11 displays the results. The required goal, which was to be tolerant to the obstacle detection signals, is achieved, since the prevision layer does not absorb any. It simply keeps creating the usual expectations.

The results in the changing locomotion gait scenario with the adaptive learning rate are shown in figure 5.12. Again, the required performance was achieved since the robot increases the learning rate when it detects a persistent error, which denotes a change in the locomotion pattern. So, it quickly learns the new conditions (within 12 strides).

The optimal learning rate used in the Newton's method is set to $\eta = 1$ in order to attain a fast converge when the robot has low confidence in its expectations. When applied to the detection of obstacles, this method shows much better results using an adaptive learning rate rule, than it had when using a static learning rate. As is shown in figure 5.13, the system becomes tolerant to point activations and does not change its prevision layer.

When confronted to changes in the activation patterns, the results presented in 5.14 using the adaptive learning rate show much better results than those obtained with the static learning rate. The mechanism is capable of performing a fast convergence when so is required, quickly learning the new activation pattern (within 12 strides).

Both models were enhanced with the the adaptive learning rate mechanism. By using the confidence level the mechanism is able to evaluate its performance and define the learning rate according the mechanism's needs at that moment. The confidence level allows the system to take advantage of both the best, optimal learning rate, when there is the need to adapt, or a nearly null learning rate when the system knows it is performing well. Also, it allows the system to be tolerant to small and instantaneous errors, responding only when it detects a persistent error.

Comparing the results obtained with the LMS and the Newton's method when using the adaptive learning rate mechanism, both fulfill the required goals with very similar results. As for the computation complexity, it is higher for the Newton's method due to its need to compute both the hessian matrix and the gradient vector, in every iteration (in equation 3.16). Thus, the most adequate approach is the LMS.

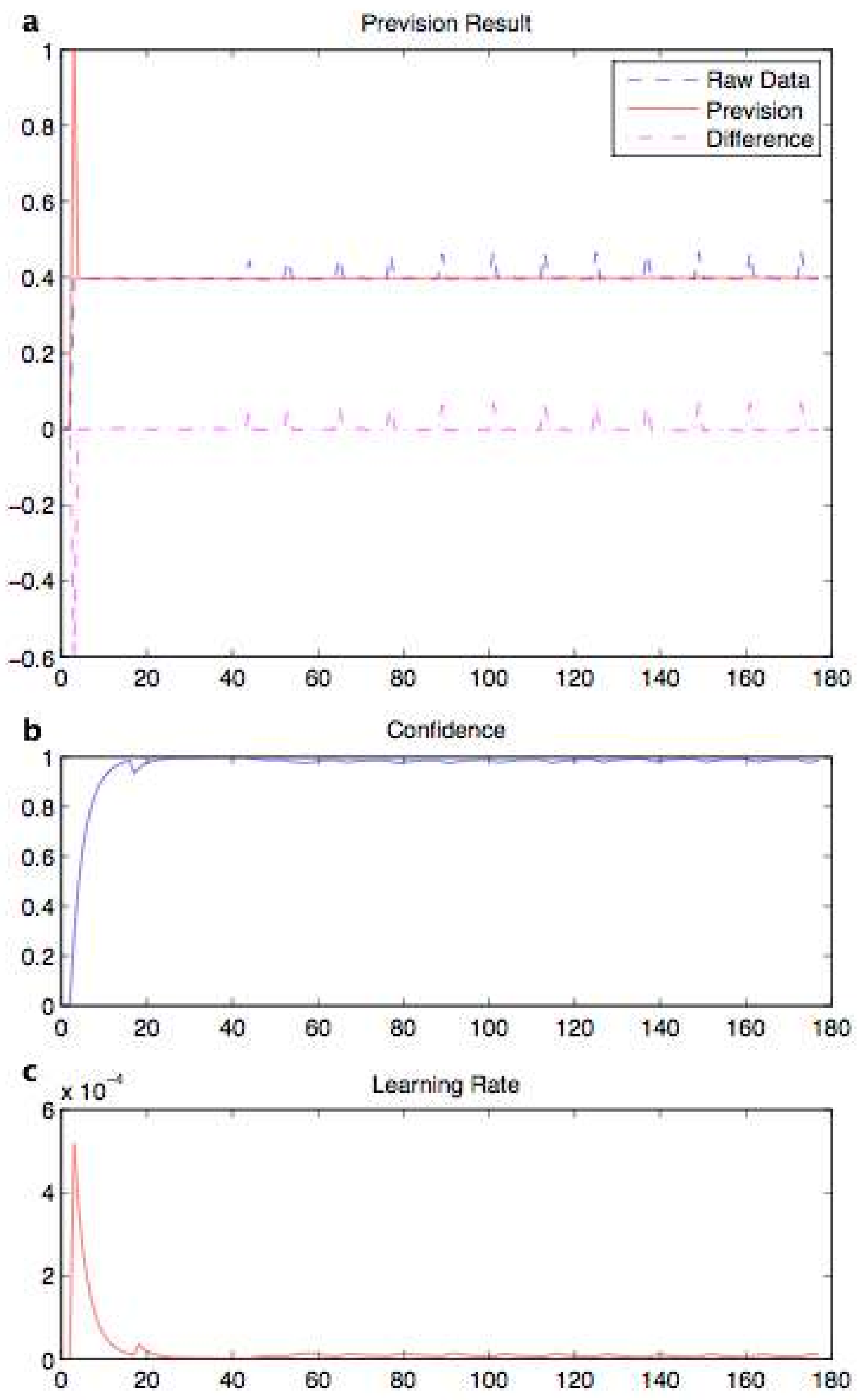

**Figure 5.11:** Results obtained in the Bioloid when the prevision layer tries to detect the obstacles. These were obtained with the LMS using an adaptive learning rate. In a we can see the error, the expectations and the raw data, b shows the confidence level and c the learning rate value.

### Filtering the noise from the Activations

The obstacle detections are no more than differences from the hypothesis created in the prevision layer and the values acquired from the environment. Noise is inevitable in the

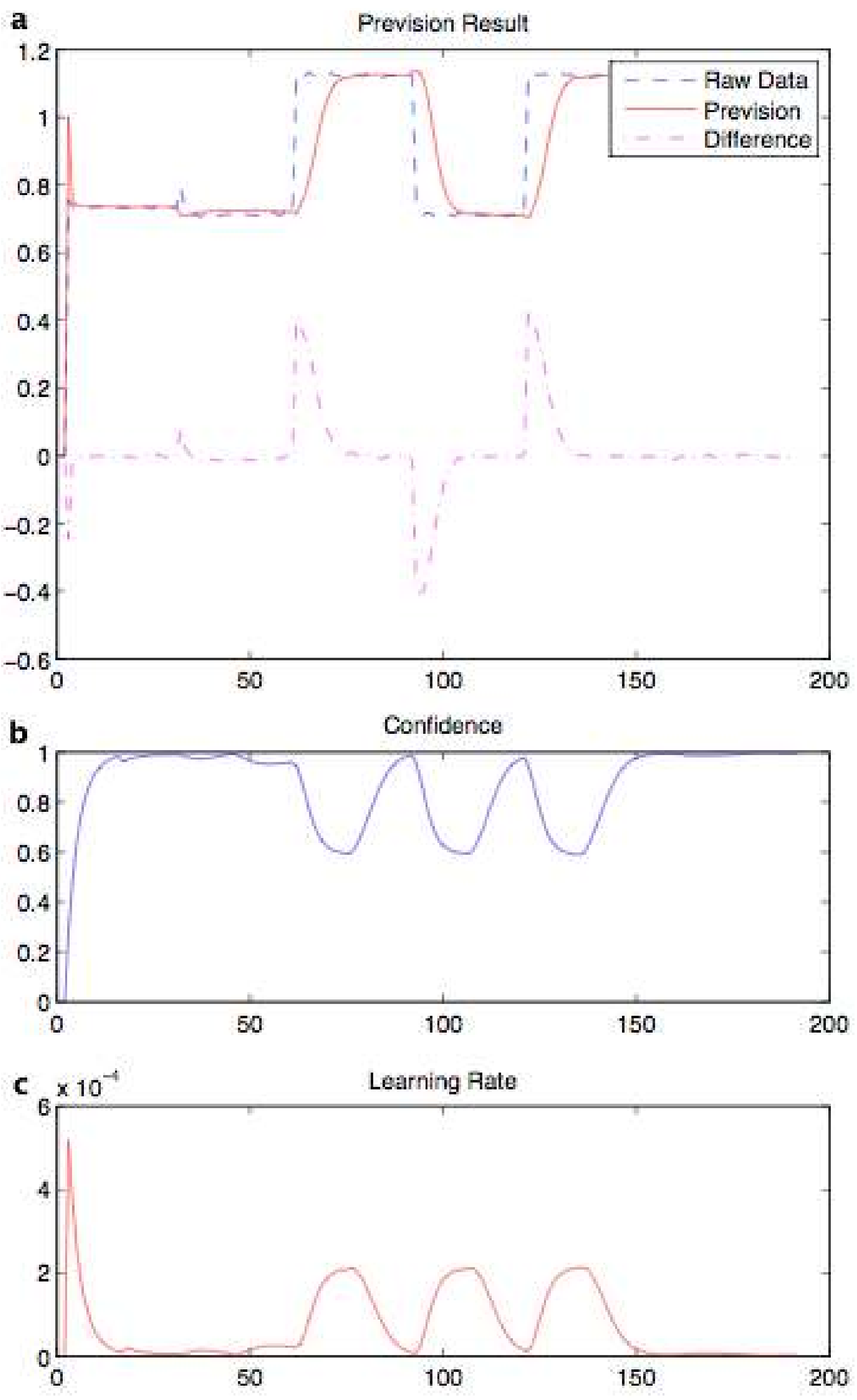

**Figure 5.12:** Results of LMS prevision and its ability to deal with changing locomotion patterns, using an adaptive learning rate in the Biloid platform. In a we have the mechanism's performance, b shows the confidence level and c the learning rate.

sensor readings, as well as other undesired features caused by the locomotion pattern (e.g. small variations). In order to distinguish the obstacle detections from all that information,

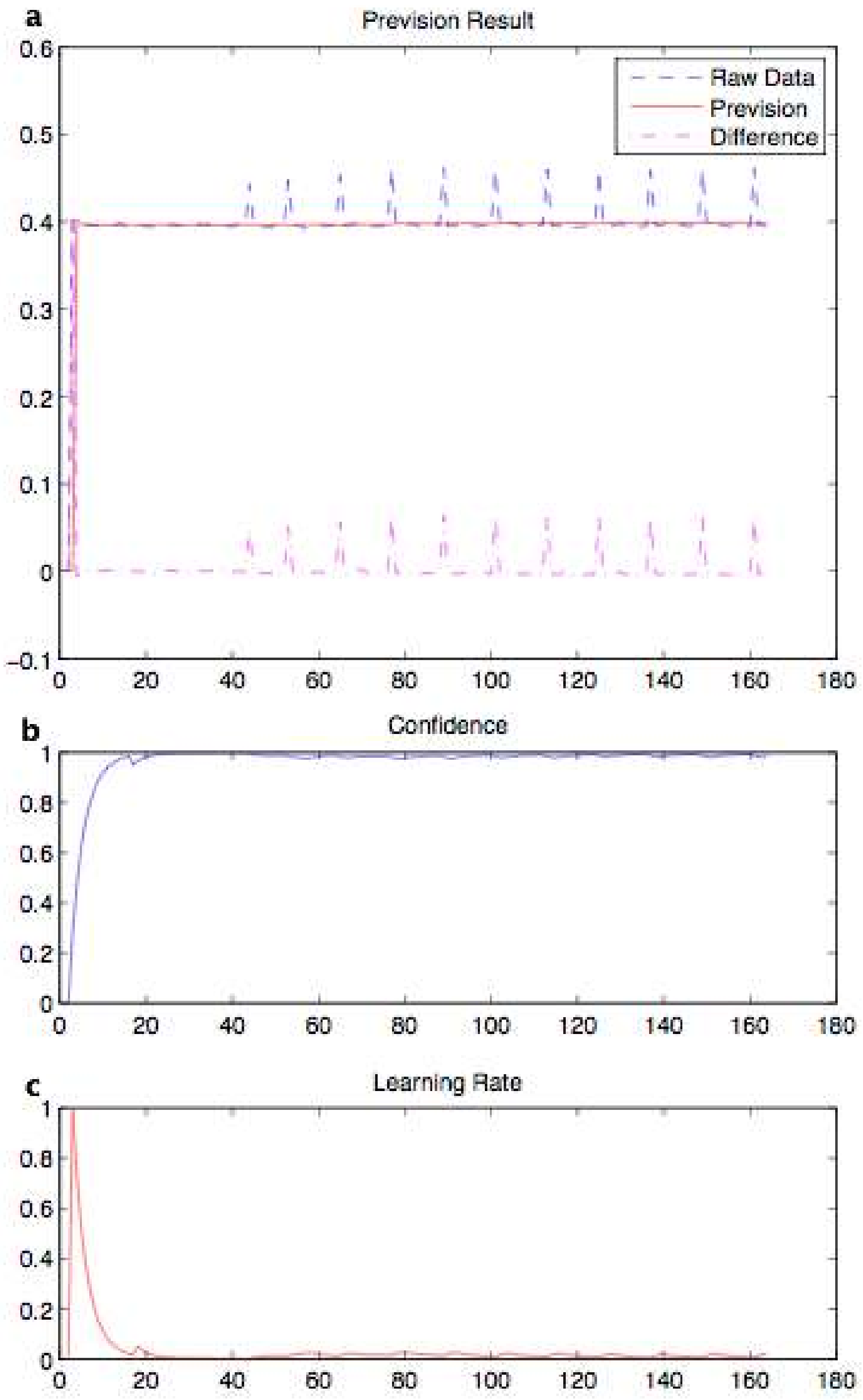

**Figure 5.13:** Newtons previsions applied to the detection of obstacles obstacle using an adaptive learning rate in the Bioloid robot. In a the raw, the expectations and the error are presented, b and c show the confidence and learning rate, respectively

the novelty layer applies a filtering process that includes a reputation variable called gain factor ($g$).

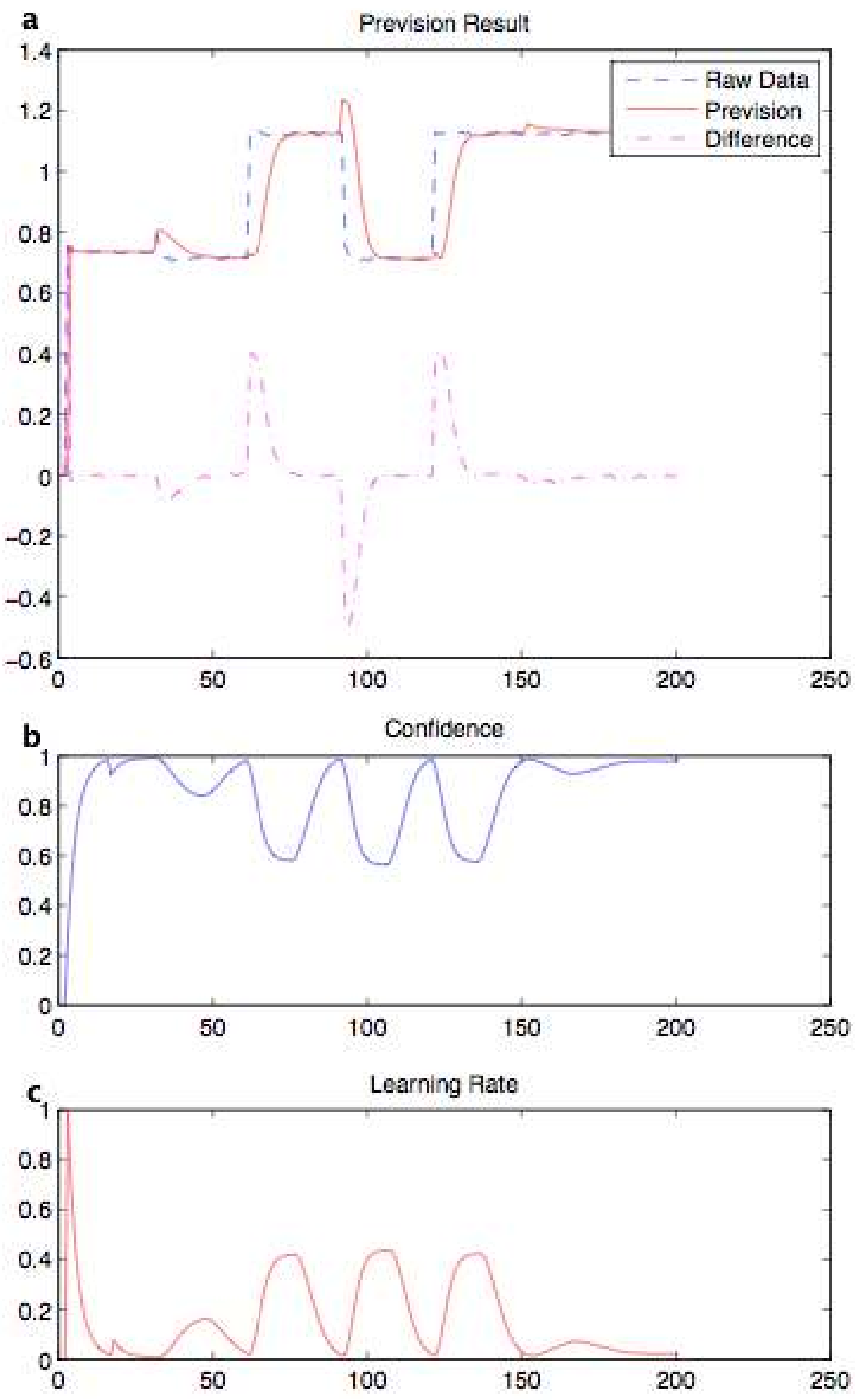

**Figure 5.14:** Newtons's method applied to deal with changing locomotion patterns, using an adaptive learning rate. Applied to the Bioloid robot. The confidence and the learning rate are displayed in b and c, respectively, and the performance is visible in a, where the error, the expectations and the raw data are present.

The confidence level presented before and a gaussian function will be evaluated in its ability to define a given cell's reputation. In order to prove the point of such mechanism, the range sensors will be defined with a noise value higher than usual, namely of 10% for each entry instead of 1%. Also, during this test the robot will execute several trials that will not include the stepping over the obstacle, it will only get close enough to detect the obstacle with all its sensors. At stride number 70 the gait will be changed from walk to trot in order to create more adverse conditions, thus, adequately evaluating this component.

The used regression model will be the LMS since it was proved to be the most adequate. Next, both approaches applied to the gain factor update mechanism will be tested. The gaussian function and the confidence mechanism.

Figure 5.15 presents two plots, the upper one shows the error and the filtered obstacle detections and bellow, one can see the gain factor variable evolution throughout the test. Around stride 70, the change in the gait is clear, since the error profile suffers some changes and some higher error values are generated. Here, the role of the filter is evident. After one single activation, the gain factor reacts and changes its value drastically, avoiding further unwanted activations of the novelty layer. Also, the obstacle detections clearly stands out, while the remaining error is flattened to 0.

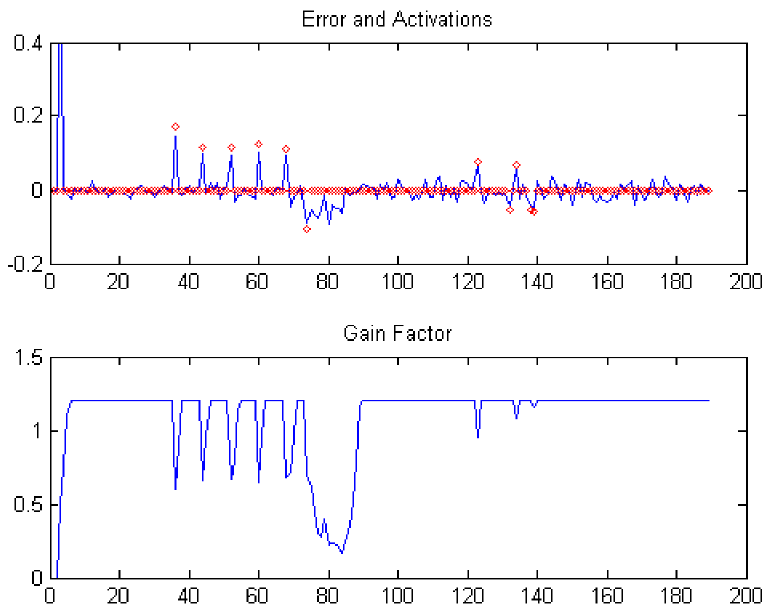

**Figure 5.15:** Results obtained with the gaussian function as feedback mechanism to the gain factor variable. Upper plot shows the error and the filtered output, the other one shows gain factor evolution.

The results obtained with the confidence mechanism, the one used before to change the learning rate, are shown in figure 5.16. In this case, the system is able to correctly filter the lesser noise values and output the correct obstacle detections, however, it was not able to deal with the noise generated during the gait alteration. Its reaction was too much delayed and several undesirable activations were passed on to the system. This consequence is actually a feature of the confidence mechanism, which was designed to deal with errors, but not to react until they become persistent and/or too strong. Therefore, its nature dos not apply to this specific problem, where a quick response to the error detection is required.

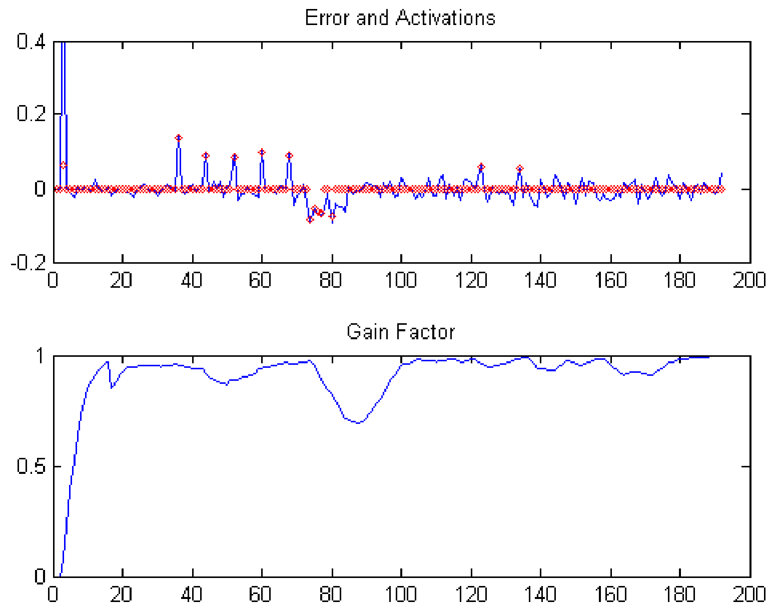

**Figure 5.16:** Results of the error filtering using the confidence mechanism to update the gain factor variable (presented in the lower plot). The error and the filtered activations are shown in the upper plot.

**AIBO Results in obstacle detection.**

The AIBO robot's results are very similar to those of the Bioloid. This is why the same detailed evaluation of each state will not be presented for this platform. Instead, a resumed version is described next.

There is one relevant difference, the Newton's method achieved a better performance in AIBO than the LMS, due to its optimal learning rate. The LMS learning rate in the AIBO is smaller than in the Bioloid. As it is smaller, the optimal convergence speed is slower. The Newton's method has the same optimal learning rate in both platforms ($\eta = 1$). The optimal learning rate used in the LMS is defined through the previously described process (equation 5.2), which one is based on the correlational matrix of the range sensors' values.

Figures 5.17 and 5.18 show the results with the LMS and the Newton's method, respectively. It is visible that the LMS takes a longer time to converge than the Newton's method.

The filtered output for the Newton's method produced with the AIBO robot is presented

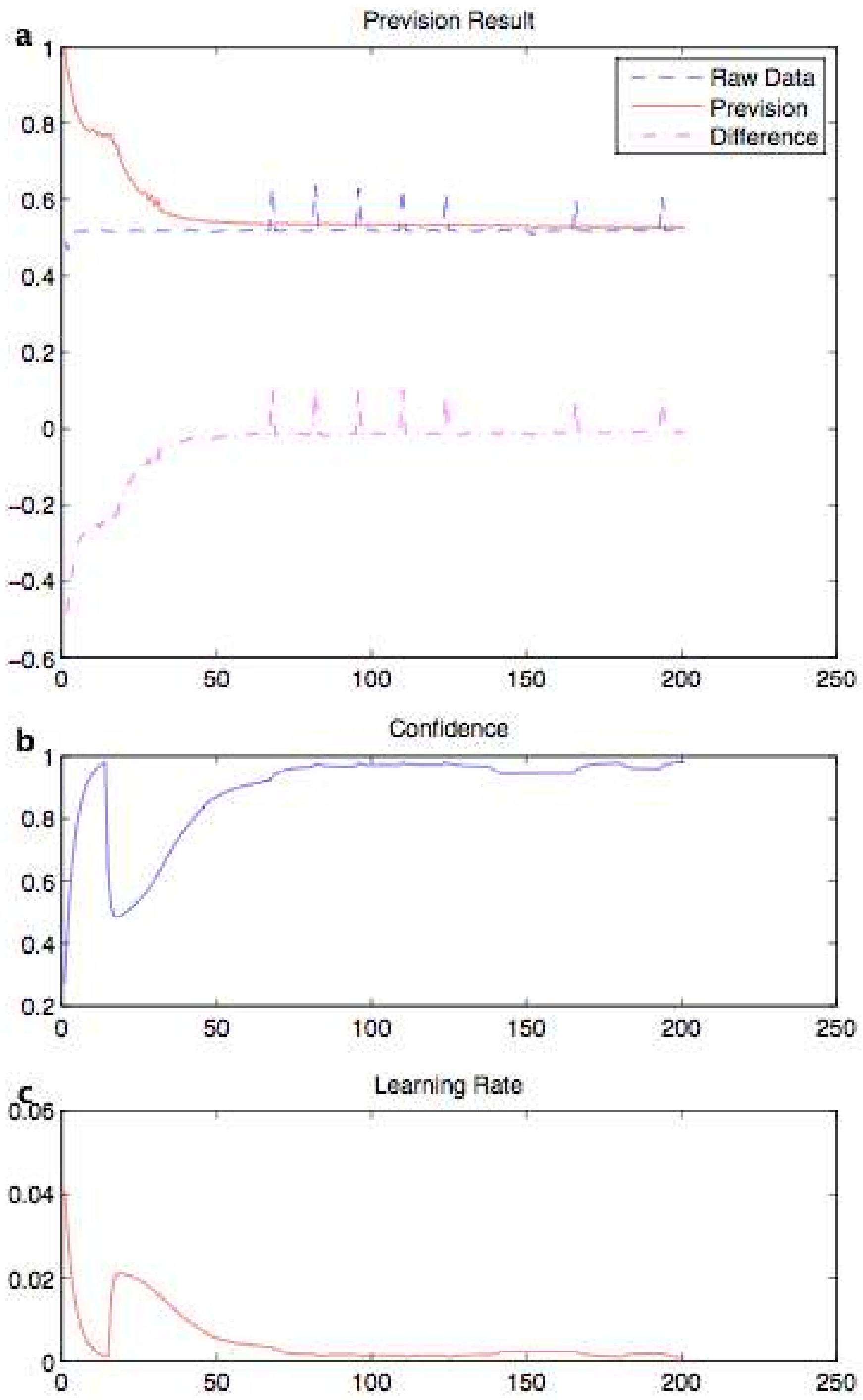

**Figure 5.17:** Prevision results obtained in the AIBO with the LMS. Shows the expectations, the raw data and the error (a), the confidence level (b) and the learning rate (c).

in figure 5.19. One can see that the obstacle detections are passed on to the second part of the mechanism, while the remaining noise is eliminated.

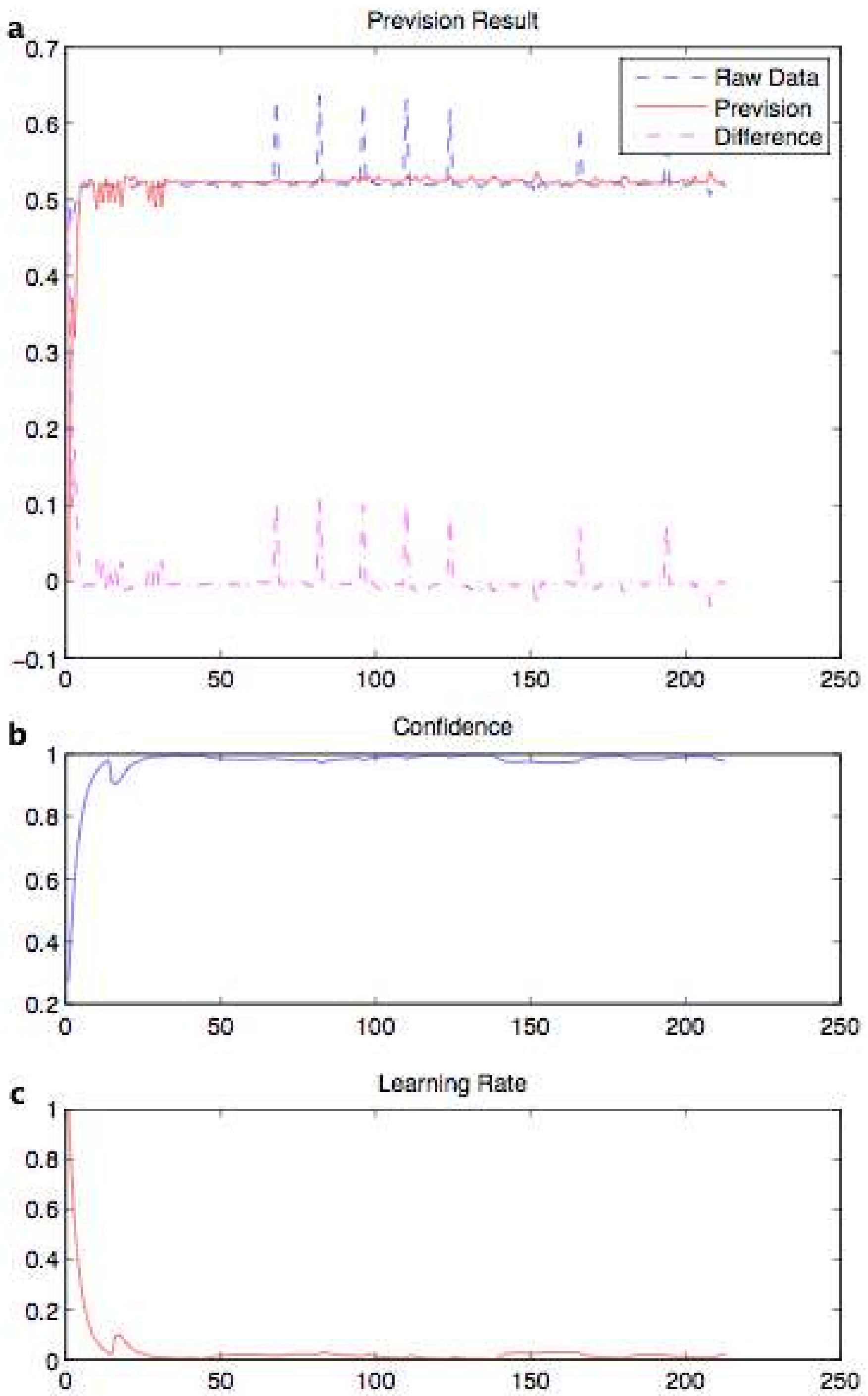

**Figure 5.18:** Results obtained with the Newton's method with the AIBO. It presents in (a) the raw data, the hypothesis and the error, and in (b) and (c), the confidence level and the learning rate, respectively.

## 5.2 Adaptive Locomotion Evaluation

The evaluation of this second part of the mechanism seeks to verify all the components in their specific goals, as well as the system's overall goal, which is to learn from experience

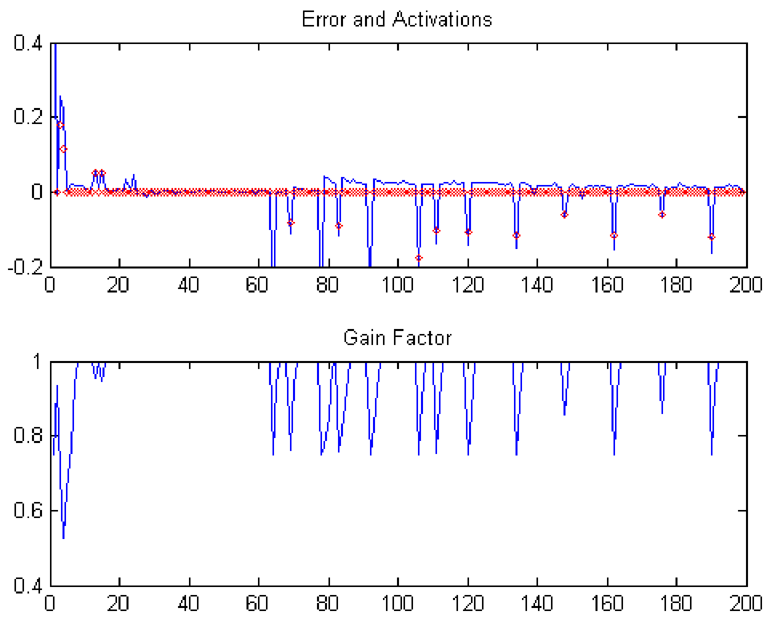

**Figure 5.19:** AIBO's performance, filtered error output as obstacle detections (upper plot) and the gain factor variable evolution.

how to change the locomotion in order to avoid the detected obstacles.

In this section, the main results will concern the Bioloid, since, as it was already pointed out before, the AIBO physical properties do not match the requirements to the attainment of the proposed goals. Its blunt and big paws make the stepping over the obstacle much too difficult, and its best gait, which is the trot, can hardly be properly executed.

Next, the results concerning each of the components of this second part are presented. Then, the whole mechanism is evaluated in its ultimate goal, to learn to avoid the obstacle by applying changes in the stride length.

### Activation Remains and Learning

The activation remains are an important part of the learning process, since these signals hold a memory of short duration that enables the robot to remember how did it approach the obstacle. This is then used to change the weight map when and if a learning signal is triggered, as it was explained before.

Whenever an obstacle is detected, an activation cell is activated alongside with a Short Term Memory (STM) cell. The value held by the STM cell is expected to last until a learning signal is triggered. The expected behavior to the STM cell is achieved through equation 3.28 and its displayed in figure 5.20.

There, despite the initial convergence error, all the activations have its value decayed in time, as expected.

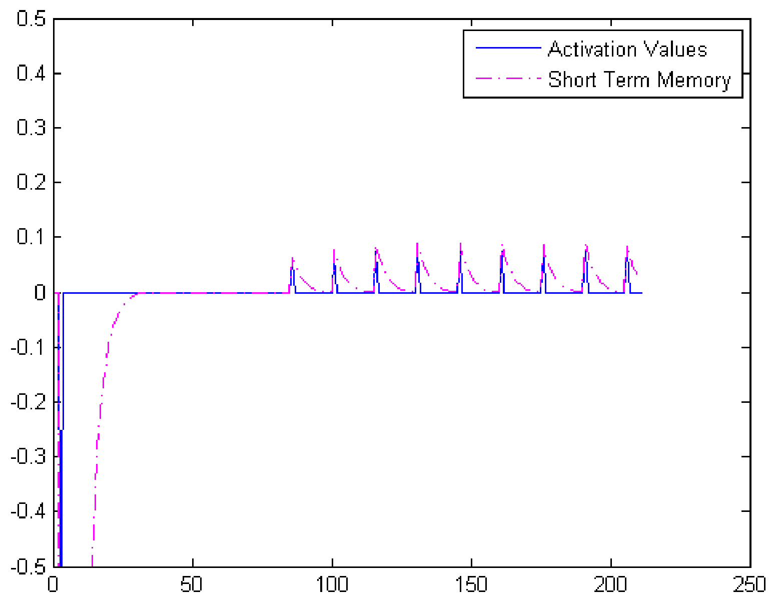

**Figure 5.20:** Short Term Memory cells behavior when activated.

The learning happens when the learning signal $\delta$ is triggered, as explained before, by changing the weight values according to the learning signal and its correspondent STM cell, which holds remains of the way the robot approached the obstacle.

Figure 5.21 shows the learning process in one cell. The lower plot presents the weight value changing according to the learning signals and the upper plot shows the STM values for the give cell and the triggered learning signals. One can see the amount of remaining activation at the moment of learning, which will define the weight change at that iteration.

### Stride Length Alteration

After learning, when the robot approaches the obstacle, the activated cells will be combined with the learnt weights in order to produce necessary alterations to the stride length. The combination of each cell is called a synapse. Figure 5.22 shows the synapse activation for the given cell on the lower plot, the middle and the upper plot show the weight value and the activation cells, respectively. The interaction between these values is clear since all change according to the weight value, also, after the last last weight change all the subsequent synapses becomes similar.

All activation and weight cells correspondent to a single stride phase - one for each sensor - are combined together in order to update the variable that changes the CPG *bl*. As previously mentioned in chapter 3, this variable might need to be amplified in order to actually change the CPG, which is required to transform the necessary stride change value,

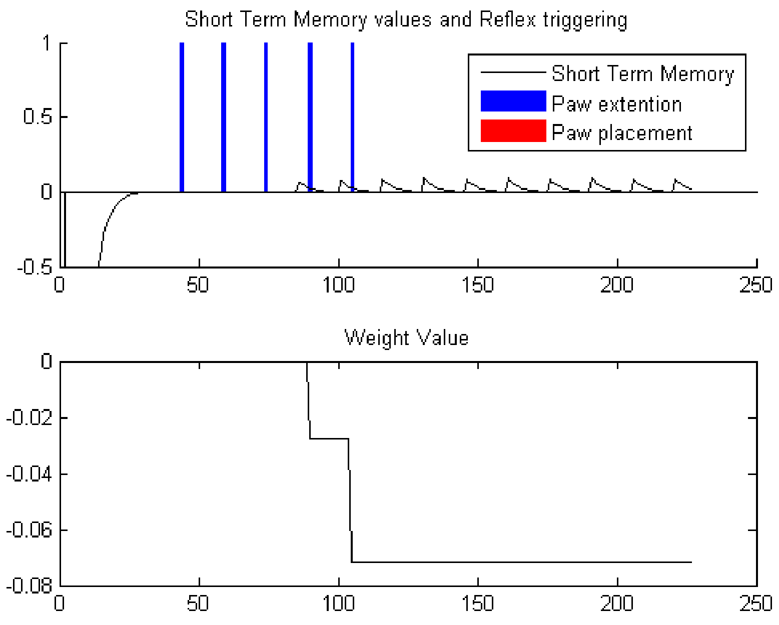

**Figure 5.21:** Weight update (lower plot) according to the Short term memory cells value and the learning signals (upper plot).

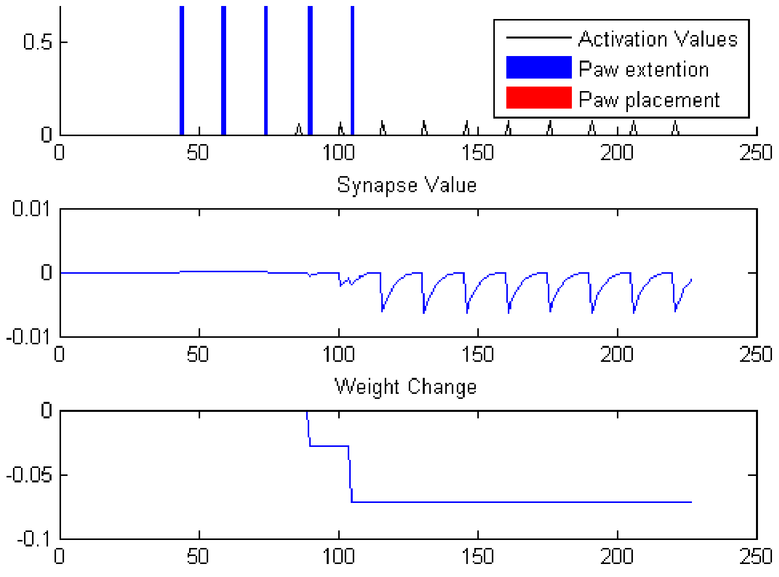

**Figure 5.22:** The synapses values (lower plot) according to the weight (middle plot) and the activation cells (upper plot).

computed in the mechanism value space, into a value that can actually change the required CPG's parameters, namely, the oscillation's amplitudes.

Figure 5.23 shows the alterations made to the stride length as system's output values (upper plot) which will be amplified and sent to the CPG. It also shows the measured stride length at each stride, displaying the system's response to the applied changes. The upper plot was not defined as a function of the strides because the changes to the CPG are computed at every iteration, so, it is computed as a function of time. However, both are scaled to the full duration of the tests, which allows a correct mach of each moment.

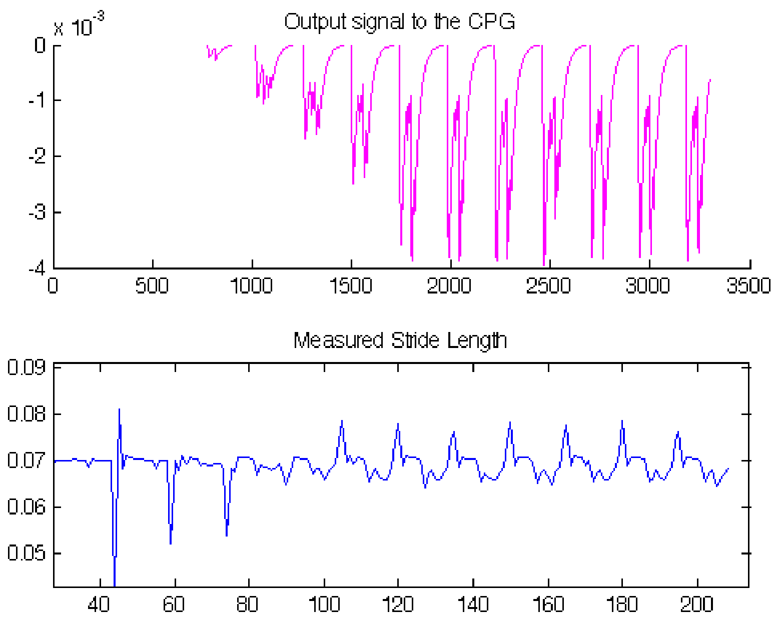

**Figure 5.23:** Stride length change (lower plot) according to mechanism output signal (upper plot).

The measured stride length plot displays a few features that require a deeper explanation. There are higher point activations that do not seem to make sense, since the mechanism output *bl* shown in the upper plot only produces stride length reduction signals. These increases are generated during the step over reflex activation, when the robot executes an increased stretch of its leg in order to overcome the obstacle. Also, at the beginning of the learning process, the robot produces some very low stride lengths, these are caused by collisions with the obstacle before the step over reflex execution. As the robot collides with the obstacle its paw will not be able to stretch as much as it would be necessary, thus, the stride length is reduced. After some learning iterations this is no longer verified, due to the robot's effort in avoiding to collide with the obstacle, the collisions become softer until they disappear. This shows that the mechanism works, since the stride length changes according to the knowledge acquired by the robot during the learning phase.

### Learning Results

Now that each element has been evaluated in its specific function, the results concerning the mechanism's goal will be presented.

In order for the system to be successful, it is required to fail by colliding with the obstacle during the step over reflex. When the robot approaches an obstacle, some cells are activated, forming a pattern that identifies that approach phase. The cells are activated according to the moments of the stride that each of the sensors detected the obstacle. An Activation Pattern (AP) can be regarded as a label that defines a way to approach the obstacle. Considering the previously defined trial concept, an AP is directly related to the different distances at which the robot is placed at the beginning of a trial. Or else, the distance to the next obstacle in the environment. Varying the distance will change the AP in the robot.

Figure 5.24 presents the results obtained when the robot is placed always at the same distance from the obstacle, meaning that it will always get the same AP. It is visible that there were only a few (5) learning iterations, which means that after those iterations, the robot learned to avoid the obstacle at that distance, when the AP is maintained.

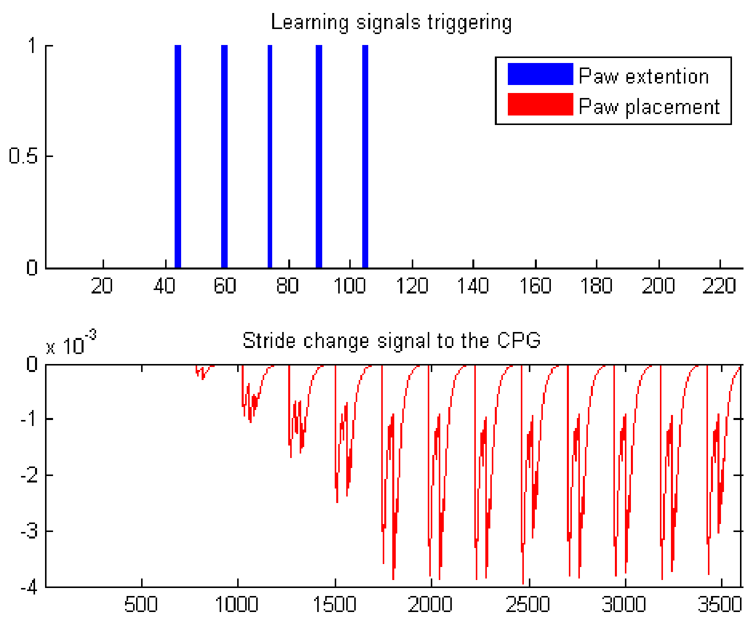

**Figure 5.24:** The robot test trying to learn to step over an obstacle always at the same distance.

However, if the distance changes, the robot will face different APs. Depending on the chosen pair of APs, the mechanism behaves differently. Let's consider a test where the robot learns to answer to a first AP and that after 10 trial, that position is changed, and the mechanism is confronted with a different one. Then, the distance to the obstacle is changed back to the first after another 10 trials and back again after another 10 trials.

Figure 5.26 shows the results for one these cases. It is visible that the mechanism is capable of learning to respond to different AP without forgetting any. After learning each one, it no longer triggered any learning signal.

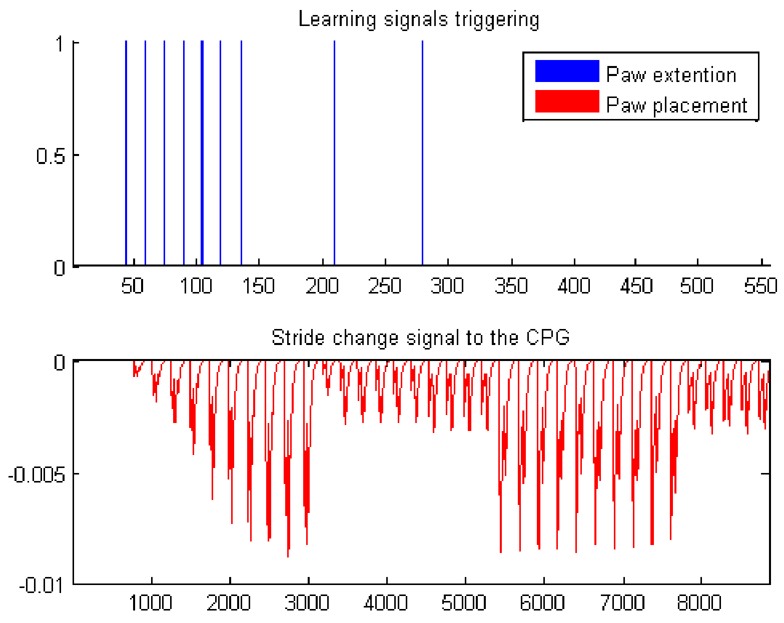

**Figure 5.25:** Ability for the system to learn to answer to a second activation patter without forgetting the first.

Another set of APs is used to execute the same test. In this case, there were required more than one learning period to learn one of the APs, in order to acquire the required knowledge and fulfill the goal. This suggested that both APs shared activation cells. I.e., the obstacle was detected during the same phase of the stride and at the same distance in both the APs. This seems to require the mechanism to run more iterations in order to achieve the correct balance between the responses to both the APs.

Since the locomotion is cyclic, and the stride length is nearly always the same, it is expected that if the robot part from two different distances that are about one stride length away from each other, the same activation pattern will occur. In order to evaluate such hypothesis the robot will first learn one AP by encountering the obstacle at a distance $d_1 = 1$, after 10 trials the distance will be changed to $d_1 + bl$, where $bl$ is the usually observed stride length 0.07. After another 10 trials, the distance will become $d_1 - bl$.

The results in figure 5.27 confirm this hypothesis, the robot was able to answer the three different APs only by learning the first. The lower plot reveals the system output variable $bl$ and it is clear that the result remains similar to all APs.

The full map, which enables the robot to answer all the possible APs, is the expected goal. As shown before, the APs are repeated if the position that define them are shifted

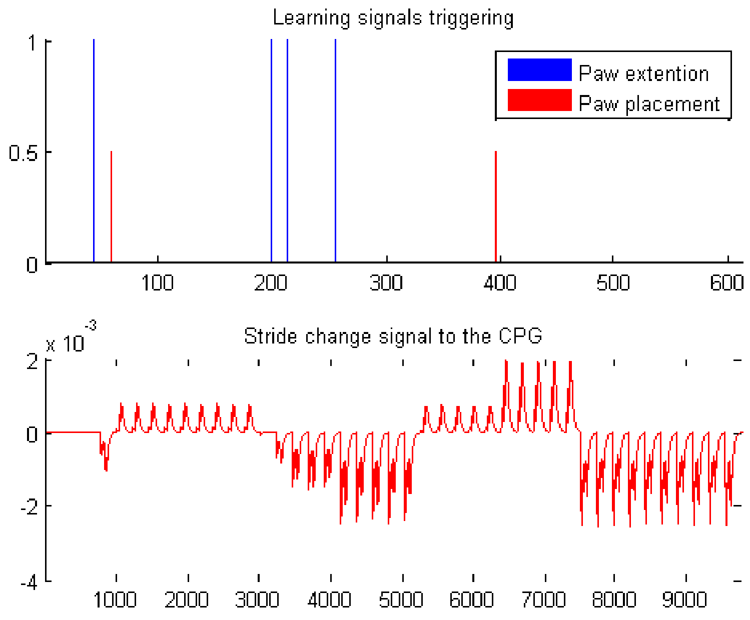

**Figure 5.26:** Ability for the system to learn to answer to a second activation pattern without forgetting the first. APs that share activation cells

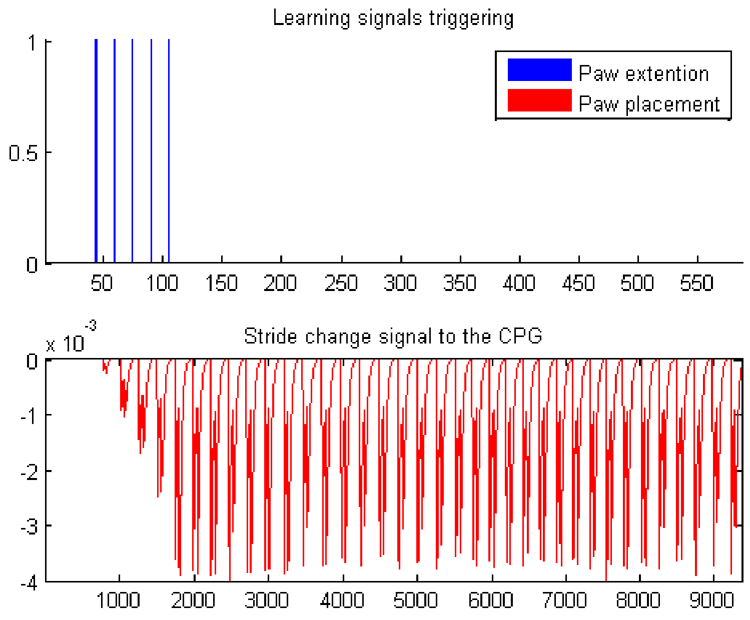

**Figure 5.27:** Evaluating the hypothesis of different distances to the obstacle can produce the same AP, when the distance bewteen them is about one stride length *bl*.

about multiples of *bl*. So, if the mechanism achieves the full learning by varying the starting position within the distance of its stride length $bl = 0.07$, it should be able to answer to any AP.

The first test will vary the starting position within the range $[st + bl, st + 2bl]$ (where $st$ is the base distance from the obstacle), considering 7 possible, uniformly distributed, positions.

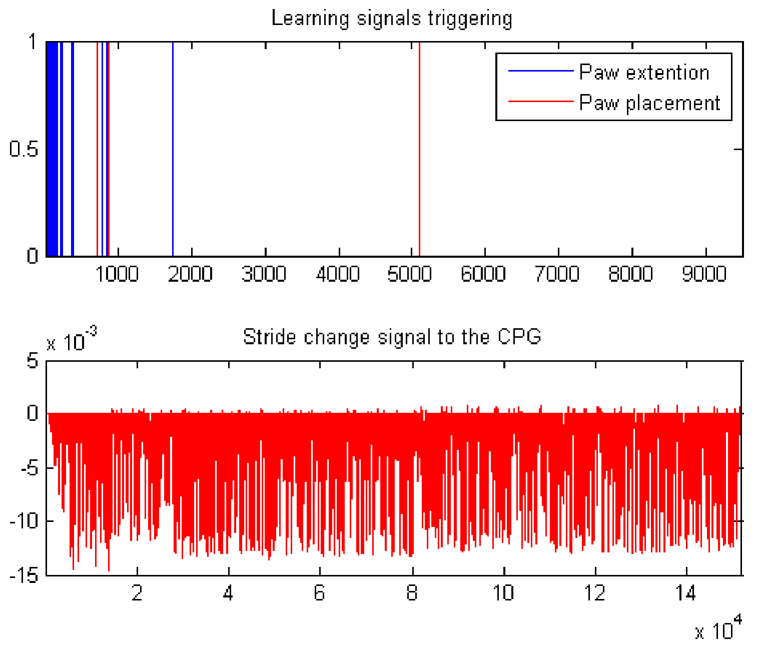

**Figure 5.28:** Learning results with 7 different, uniformly distributed AP. Upper plot shows the triggered learning signals and below, the changes in the system's output *bl*.

In figure 5.28 the results show that the mechanism acquired the necessary information to respond to all the 7 APs, thus, enabling the robot to avoid stumbling on the obstacle.

A second test used the double possible starting positions within the same range. Figure 5.29 displays the resulting activations. In this case, the mechanism was not able to fully learn the required map. The higher number of APs requires a much harder learning process as the number of common cells between different APs increases. So, the mechanism needs to find the balance between them all. This higher number of possible starting positions seems to be much too high for the system to learn the full map.

These results show that the mechanism is able to learn and to adapt the locomotion according to the detected obstacles. However, there is a limit to the number of AP that when crossed, the mechanism is able to avoid the obstacles, but eventually fails and needs to readapt. Nevertheless, the robot learns and avoids the obstacles using the acquired information, without any specific indication of what is an obstacle.

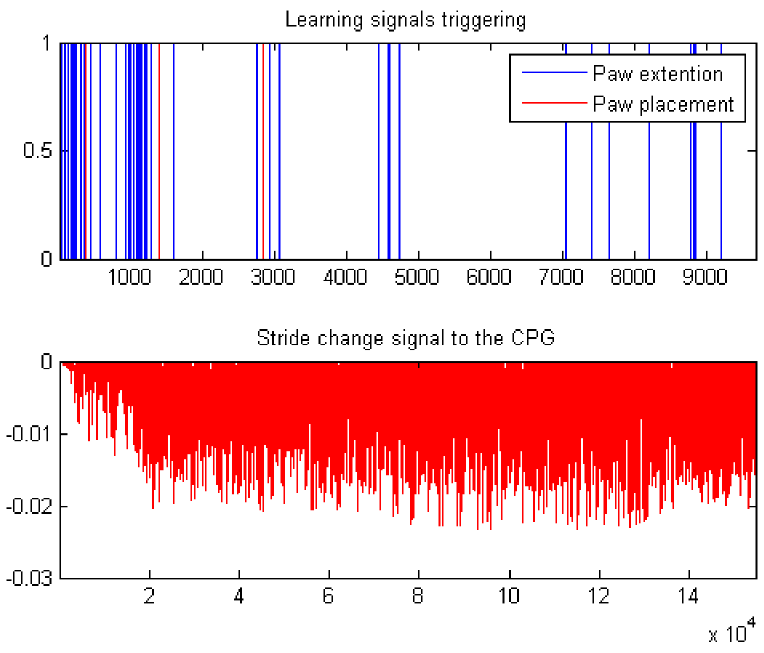

**Figure 5.29:** Results using 14 possible APs. Upper plot shows the triggered learning signals and below, the changes in the system's output *bl*.

# Chapter 6

# Conclusion and Discussion

The proposed mechanism requires several components in order to achieve its goal. The correct functioning of each one will determine the system's success. The previous chapter presented the results obtained and compared them to what was expected. Each component acheived the desired goals and, generally, the system fulfilled the required tasks. The robot was actually able to both detect and learn to avoid the obstacle.

Concerning the obstacle detection, the system achieved the best expectations, it was able to detect the obstacle and to adaptively learn to do so. The results showed that the forward internal model defined in the prevision layer obtained good expectation values after some learning iterations (some strides). This enabled the system to quickly generate reliable error signals (differences between the created expectations and the raw sensor readings), that when fed to the novelty layer made possible the correct discerning of obstacle detections among the noise and the remain error signals. Also, the novelty layer generated the desired behavior. The used reputation variable enables the system to avoid undesired activations, such as the ones that arise from changes in the locomotion. Thus, the obstacle detections depend on the error and the reliability of such error.

Another crucial point to the obstacle detection is the adaptive learning rate mechanism used to change the internal forward model's plasticity. It enables the system to differentiate the errors caused by the obstacle detections from those generated when changes in the locomotion occur. Thus, avoiding absorb the obstacle detections as part of the locomotion pattern and adapting to changes in the locomotion. Although the required behavior was successfully achieved, future work could further improve the mechanism's performance. Namely, into reducing the required time to readapt whenever a different pattern is detected. Results showed that the confidence level is reduced when this situation is detected, however, the confidence remains high ($\beta > 0.5$). So, the optimal learning rate is underused.

The methods used to implement the robot's perception, namely the detection of the obstacles, offers an important feature to design an adaptable and autonomous robot controller. The ability to distinguish the self caused perception changes from those that are externally generated, enables environment's perception and self awareness about the robot and its movements. Also, this enables the robot to evaluate its performance by comparing the obtained results to the expected values.

Learning to avoid the obstacle, the second proposed goal for this mechanism, was also successfully achieved. The robot learns to avoid the obstacle through experience, it tunes the weight map every time time it fails by colliding with the obstacle. As a consequence, it learns how should it change the locomotion in order to avoid such situations in the future, placing its paw optimally before executing the step over reflex.

The presented concept of Activation Pattern (AP) - defined by a set of activated cells during the obstacle approximation phase - is used to evaluate the different ways the robot can approach the obstacle. It was shown that when systematically faced with the same AP, the robot learns to answer it within a few iterations - or trials. Then, the robot was faced with two activation patterns interchangeably, with a learning period for each one. Here, we obtained different results when facing different pairs of APs. In some cases it learned the first, then the second, and the learning iterations stopped, which means that both AP were acquired by the map without forgetting any. On other cases, it required more than one learning period to learn one of them. This suggested that, on these cases, the two APs shared activation cells, i.e., the robot detected the obstacle at the same distance and during the same phase of the locomotion, when approaching the obstacle in both ways. That required the system to find a balance between these activation cells, or using different activated cells in the AP, to save the necessary alterations to the locomotion.

When facing several, randomly chosen, APs, the system's success seems to depend on the number of these APs. If the possible APs are only a few (7), the system seems to learn the full map even if it takes more than 350 learning iterations (trials), however it is able to acheive a reasonably good performance much earlier. As the number of APs increases - 14 or more -, it becomes harder for the system to fit all the solutions in the same weight map. This suggests that the number of shared cells is bigger, therefore the balance between all of them becomes a much complex problem.

Nevertheless, the robot did learn to change its stride length in order to optimally place its paw before triggering the step over reflex. Also, even though the robot eventually fails to step over the obstacle when the number of APs is too great, most of the trials are successful. This shows that the ultimate goal, learning to avoid the obstacles, was achieved. The robot learns by experience to detect and avoid an obstacle.

# Bibliography

[1] J. Albiez, W. Ilg, T. Luksch, K. Berns, and R. Dillmann. Learning a reactive posture control on the four-legged walking machine bisam. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 2, pages 999–1004. IEEE, 2001.

[2] G. Asuni, G. Teti, C. Laschi, E. Guglielmelli, and P. Dario. A bio-inspired sensory-motor neural model for a neuro-robotic manipulation platform. In *Advanced Robotics, 2005. ICAR '05. Proceedings., 12th International Conference on*, pages 607 –612, 2005.

[3] A. De Rengervé, Sofiane Boucenna, P. Andry, and P. Gaussier. Emergent imitative behavior on a robotic arm based on visuo-motor associative memories.

[4] F. Doshi, E. Brunskill, A. Shkolnik, T. Kollar, K. Rohanimanesh, R. Tedrake, and N. Roy. Collision detection in legged locomotion using supervised learning. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 317 –322, oct. 2007.

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

[6] James J. Gibson. *The Ecological Approach to Visual Perception*, volume 40. Houghton Mifflin, 1979.

[7] R. Held. Sensory deprivation: Facts in search of a theory. exposure-history as a factor in maintaining stability of perception and coordination. *Journal of Nervous and Mental Disease*, 1961.

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

[9] W. Ilg, J. Albiez, H. Jedele, K. Berns, and R. Dillmann. Adaptive periodic movement control for the four legged walking machine bisam. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 3, pages 2354–2359. IEEE, 1999.

[10] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal. Fast, robust quadruped locomotion over challenging terrain. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2665–2670. IEEE, 2010.

[11] H. Kimura and Y. Fukuoka. Adaptive dynamic walking of a quadruped robot on irregular terrain by using neural system model. In *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 2, pages 979 –984 vol.2, 2000.

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

[13] N. Kohl and P. Stone. Machine learning for fast quadrupedal locomotion. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 611–616. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004.

[14] M. Anthony Lewis and Lucia S. Simó. Certain principles of biomorphic robots. *Auton. Robots*, 11(3):221–226, 2001.

[15] M.A. Lewis. Detecting surface features during locomotion using optic flow. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 1, pages 305 – 310 vol.1, 2002.

[16] M.A. Lewis and G.A. Bekey. Gait adaptation in a quadruped robot. *Autonomous robots*, 12(3):301–312, 2002.

[17] M.A. Lewis and L.S. Simo. Elegant stepping: A model of visually triggered gait adaptation. *Connection Science*, 11(3):331–344, 1999.

[18] M. Lopes and J. Santos-Victor. Learning sensory-motor maps for redundant robots. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2670 –2676, oct. 2006.

[19] K.F. MacDorman. Responding to affordances: learning and projecting a sensorimotor mapping. volume 4, pages 3253 –3259 vol.4, 2000.

[20] P. Manoonpong and F. Wörgötter. Efference copies in neural control of dynamic biped walking. *Robotics and Autonomous Systems*, 57(11):1140–1153, 2009.

[21] Poramate Manoonpong, F. Pasemann, and J. Fischer. Modular neural control for a reactive behavior of walking machines. pages 403 – 408, jun. 2005.

[22] V. Matos and C.P. Santos. Omnidirectional locomotion in a quadruped robot: A cpg-based approach. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3392–3397. IEEE.

[23] Takeshi Mori, Yutaka Nakamura, Masa-Aki Sato, and Shin Ishii. Reinforcement learning for a cpg-driven biped robot. In *AAAI'04: Proceedings of the 19th national conference on Artifical intelligence*, pages 623–630. AAAI Press, 2004.

[24] L. Righetti and A.J. Ijspeert. Pattern generators with sensory feedback for the control of quadruped locomotion. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 819 –824, May 2008.

[25] R. Saegusa, G. Metta, and G. Sandini. Active learning for multiple sensorimotor coordination based on state confidence. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2598 –2603, oct. 2009.

[26] C.P. Santos, M. Oliveira, A.M.A.C. Rocha, and L. Costa. Head motion stabilization during quadruped robot locomotion: Combining dynamical systems and a genetic algorithm. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 2294–2299. IEEE.

[27] G. Schram, F. X. van der Linden, B. J. A. Krose, and F. C. A. Groen. Visual tracking of moving objects using a neural network controller. *Robotics and Autonomous Systems*, 18(3):293 – 299, 1996. IAS-4 Conference.

[28] J. Schröder-Schetelig, P. Manoonpong, and F. Wörgötter. Using efference copy and a forward internal model for adaptive biped walking. *Autonomous Robots*, pages 1–10, 2010.

[29] Claude F. Touzet. Modeling and simulation of elementary robot behaviors using associative memories. In InTech, editor, *International Journal of Advanced Robotic Systems*, volume ISSN: 1729-8806, 2008.

[30] Duc Trong Tran, Ig Moo Koo, Gia Loc Vo, Se-gon Roh, Sangdeok Park, Hyungpil Moon, and Hyouk Ryeol Choi. A new method in modeling central pattern generators to control quadruped walking robots. In *IROS'09: Proceedings of the 2009 IEEE/RSJ*

*international conference on Intelligent robots and systems*, pages 129–134, Piscataway, NJ, USA, 2009. IEEE Press.

[31] Erich von Holst and Horst Mittelstaedt. Das Reafferenzprinzip. *Naturwissenschaften*, 37(20):464–476, January 1950.

[32] D.M. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7-8):1317–1329, 1998.