A Heterosynaptic Spiking Neural System for the Development of Autonomous Agents

Thesis

A Heterosynaptic Spiking Neural System for the Development of Autonomous Agents

Cristian Jimenez Romero

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Science, Technology, Engineering and Mathematics

The Open University

Supervisors: Prof. Jeffrey H. Johnson and Dr Jane M. Bromley

25-August-2016

Abstract

Artificial neural systems for computation were first proposed three quarters of a century ago and the concepts developed by the pioneers still shape the field today. The first generation of neural systems was developed in the nineteen forties in the context of analogue electronics and the theoretical research in logic and mathematics that led to the first digital computers in nineteen forties and fifties. The second generation of neural systems implemented on digital computers was born in the nineteen fifties and great progress was made in the subsequent half century with neural networks being applied to many problems in pattern recognition and machine learning. Through this history there has been an interplay between biologically inspired neural systems and their implementation by engineers on digital machines. This thesis concerns the third generation of neural networks, Spiking Neural Networks, which is making possible the creation of new kinds of brain inspired computing architectures that offer the potential to increase the level of realism and sophistication in terms of autonomous machine behaviour and cognitive computing. This thesis presents the development and demonstration of a new theoretical architecture for third generation neural systems, the Integrate-and-Fire based Spiking Neural Model with extended Neuro-modulated Spike Timing Dependent Plasticity capabilities. This proposed architecture overcomes the limitation of the homosynaptic architecture underlying existing implementations of spiking neural networks that it lacks a natural spike timing dependent plasticity regulation mechanism, and this results in 'run away' dynamics. To overcome this ad hoc procedures have been implemented to overcome the 'run away' dynamics that emerge from the use of spike timing dependent plasticity among other hebbian-based plasticity rules. The new heterosynaptic architecture presented, explicitly abstracts the modulation of complex biochemical mechanisms into a simplified mechanism that is suitable for the engineering of artificial systems with low computational complexity. Neurons work by receiving input signals from other neurons through synapses. The difference between homosynaptic and heterosynaptic plasticity is that, in the former the change in the properties of a synapse (e.g. synaptic efficacy) depends on the point to point activity in either of the sending and receiving neurons, in contrast for heterosynaptic plasticity the change in the properties of a synapse can be elicited by neurons that are not necessary presynaptic or postsynaptic to the synapse in question. The new architecture is tested by a number of implementations in simulated and real environments. This includes experiments with a simulation environment implemented in Netlogo, and an implementation using Lego Mindstorms as the physical robot platform. These experiments demonstrate the problems with the traditional Spike timing dependent plasticity homosynaptic architecture and how the new heterosynaptic approach can overcome them. It is concluded that the new theoretical architecture provides a natural, theoretically sound, and practical new direction for research into the role of modulatory neural systems applied to spiking neural networks.

Acknowledgements

First of all, I would like to thank my thesis supervisors Professor Jeff Johnson and Dr. Jane Bromley who during my entire PhD journey have supported me, not only academically and technically but also as a friendly presence that often helped me keep going ahead during difficult times.

I also want to express my gratitude to the Open University for giving me the opportunity to further develop my academic career by providing me with the academic and financial support necessary for the completion of all the work behind this thesis.

I would like to thank my soul mate, my lovely parents and my wonderful sister for always believing in me and for understanding the reasons why I could not be with them as much as I wished.

I dedicate this thesis to God, and express my gratitude for allowing me to appreciate the divinity of the world through the lenses of science.

Acronyms

ANN	Artificial Neural Networks	Page 14
CNS	Central Nervous Systemm	Page 18
EC	Equilibrium Concentration	Page 67
EM	Excitability Modulation	Page 46
EMS	Excitability Modulatory Substance	Page 40
EPSP	Excitatory Postsynaptic Potential	Page 8
I&F	Integrate and Fire Model	Page 8
IPSP	Inhibitory Postsynaptic Potential	Page 8
LI&F	Leaky Integrate and Fire model	Page 19
LTD	Long Term Depression	Page 25
LTP	Long Term Potentiation	Page 25
MS	Modulatory Substance	Page 71
NM	Neuro-modulatory Neuron	Page 40
PD	Pulse Driver Neuron	Page 40
PM	Plasticity Modulation	Page 40
PMS	Plasticity Modulatory Substance	Page 39
PSP	Postsynaptic potential	Page 16
SNN	Spiking Neural Networks	Page 8
SP	Sub-process	Page 54
SRM	Spike Response Model	Page 19
STDP	Spike-Timing-Dependent Plasticity	Page 8

Contents

Chapter 1

Introduction

The emergence of Spiking Neural Networks (SNN) as a control mechanism for robots and autonomous self-adaptive systems and the development of other novel machine-learning technologies makes it possible to create brain-inspired computational architectures with the potential to increase the level of sophistication in terms of autonomous machine behavior and cognitive computing.

Artificial neural systems for computation were first proposed as computational systems by McCulloch and Pitts in 1943 [1]. In the nineteen forties there was great interest in the modelling of neural systems, including the mathematical theory developed by von Neumann [2]. A particularly important innovation was the learning theory proposed by Hebb [3]. This first generation of neural systems was developed in the nineteen forties in the context of analogue electronics and the theoretical research in logic and mathematics that led to the first digital computers in nineteen forties and fifties. The second generation of neural systems was implemented on digital computers as they began to be available in the nineteen fifties. Great progress was made over the next half century with neural networks being applied to many problems in pattern recognition and machine learning. Throughout this history there has been an interplay between biologically inspired neural systems and their implementation by engineers on digital machines. For example, in the 1980s Aleksander et al. [4] developed the Wisard pattern recognition system that could recognise faces using a biologically inspired associative memory. In contrast, the multilayer perceptron based on non-biological mathematical principles proved to be a powerful engineering solution to a wide range of classification tasks.

The third generation of neural networks, Spiking Neural Networks, which makes possible the realisation of new kinds of brain inspired computing architectures. These have the potential to increase the level of realism and sophistication of neural systems in terms of autonomous learning behaviour and cognitive computing.

The details of all this are given in the following chapters. Put simply, neurons work by receiving input signals from other neurons through joins or synapses (Figure 2.1). If sufficient pulses are received in a given time the potential of the neuron increases sufficiently for it to fire and transmit an output pulse to other neurons.

This thesis presents a completely new theoretical architecture for third generation neural systems, the Integrate-and-Fire based Spiking Neural Model with extended Neuromodulated Spike Timing Dependent Plasticity. This system has capabilities that enable it to be used in autonomous agents and robots where learning and adaptation to the environment must occur without human intervention. Beyond developing the theory of this new system, the thesis demonstrates that it can be implemented in both simulated and real robotic systems.

This proposed architecture overcomes the limitation of the homosynaptic architecture of existing implementations of spiking neural networks, namely that they lack a natural spike timing dependent plasticity regulation mechanism, and this results in 'run away' or 'out of control' dynamics in their implementation of the hebbian-based plasticity rules. The ad hoc procedures implemented to overcome these problems have had limited success, and the conventional theory has become fragmented and lacking in coherence and generality. The difference between homosynaptic and heterosynaptic plasticity is that, in the former the change in the properties of a synapse (e.g. synaptic efficacy) depends on the point to point activity in either of the sending and receiving neurons, in contrast for heterosynaptic plasticity the change in the properties of a synapse can be elicited by neurons that are not necessary presynaptic or postsynaptic to the synapse in question. The new heterosynaptic architecture presented here explicitly abstracts the modulation of complex biochemical mechanisms into a simplified mechanism that is suitable for the engineering of artificial neural systems with low computational complexity. The new architecture is tested by a number of implementations as simulated and real environments. This includes experiments

with a simulation environment implemented in Netlogo, and an implementation using Lego Mindstorms as the physical robot platform. These experiments demonstrate the problems with the traditional Spike timing dependent plasticity homosynaptic architecture and how the new heterosynaptic approach can overcome them.

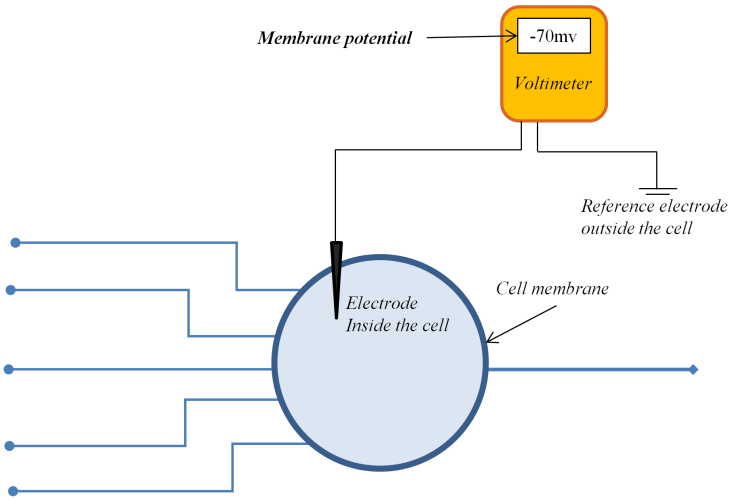
It is concluded that the new theoretical architecture provides a natural, theoretically sound, and practical new direction for research into the role of modulatory neural systems in spiking neural networks and their application in machine learning and autonomous control.

1.1 Motivation and research contribution

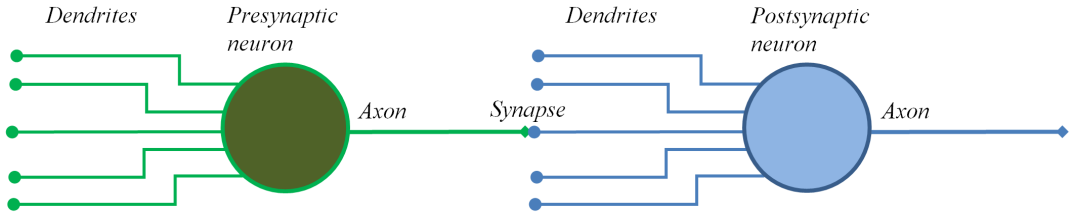
Compared to well established artificial neural architectures such as multilayer perceptrons, the dynamics of Spiking Neural Networks are based locally on the 'membrane potential' of each neuron, where this is the potential difference between the electrical charge inside the neuron and the electrical charge of its surrounding environment (Figure 1.1(a)). Apart from neurons responding individually according to changes in their membrane potential, spiking neural networks are dynamic in the patterns of discrete electrical 'spikes' produced through time as neurons fire, and in the time-dependent nature of the membrane potential. Understanding the global behaviour of spiking neural networks is a research goal, and this thesis suggests a new approach for the interactions between pre-synaptic and post-synaptic neurons (Figure 1.1(b)) by making them heterosynaptic.

The standard Spike Timing Dependent Plasticity rule (STDP rule) is an implementation of Hebbian learning that describes the evolution of the synaptic efficacy between a pre and a post-synaptic neurons in terms of the relative timings of the incoming excitatory postsynaptic potentials (EPSP) and inhibitory postsynaptic potentials (IPSP) elicited by the presynaptic neuron and the generated action potential in the postsynaptic neuron. This mechanism allows the postsynaptic neuron to recognize the incoming pulses as causal or acausal of its activation and consequently reinforce or weaken the corresponding synapses, respectively.

In terms of Neuro-engineering, STDP has demonstrated to be successfully implemented as a learning mechanism in autonomous systems in both simulated and real environments



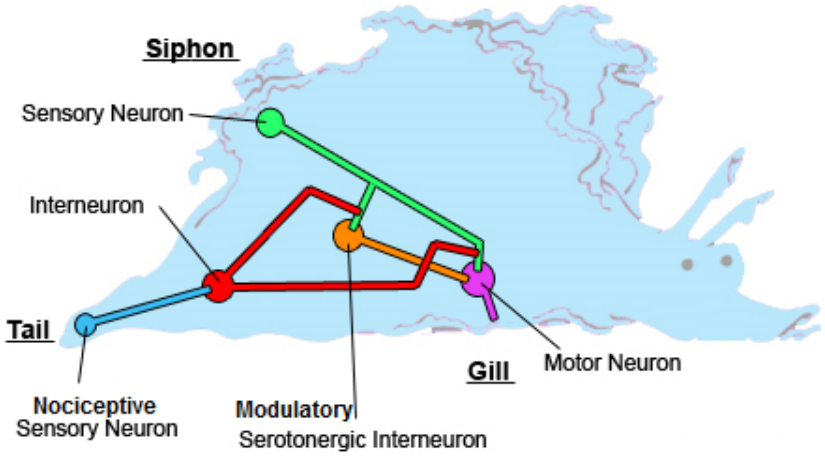
(a) Membrane potential: voltimeter showing the potential difference (-70mv) when one of its electrodes is inside the neuron and the other outside of it.



(b) Presynaptic and postsynaptic neurons: the neuron shown in green extends its axon to a dendrite of the blue neuron. Thus the green neuron is presynaptic to the blue one.

Figure 1.1

(*e.g.*[5, 6, 7, 8]). STDP has been used as the underlying neural learning mechanism for reinforcement learning and classical conditioning in bio-inspired robots. However, in spite of its success, STDP has the disadvantage that the learning mechanism does not contain any information about the nature of the incoming pulses [9], *i.e.* a pure STDP-based neural system cannot differentiate between pulses from several presynaptic neurons apart from their arriving times and synaptic efficacies. In contrast, in biological neurons, neurotransmitter systems as well as other neurochemicals allow the differentiation and modulation of different types of information and the way neurons behave under certain conditions. For instance, as observed in the sea slug Aplysia Californica (see figure 1.2 below), the activation of a nociceptive pathway triggers the activation of modulatory interneurons releasing serotonin which enhances the synaptic plasticity between sensory neurons (siphon and upper mantle) and motor-neurons (gill withdrawal reflex) [10, 11, 12]. This facilitation of synaptic plasticity between sensory and motor neurons results into what is called *sensitization*, characterized by a strengthen response of motoneurons to stimulation at sensory neurons.



http://upload.wikimedia.org/wikipedia/commons/thumb/c/cc/Aplysia-1.jpg/440px-Aplysia-1.jpg

Figure 1.2: Modulatory serotonergic interneurons (shown in orange) in the Aplysia californica enhance synaptic plasticity in motor neurons (shown in lila) leading to stronger motor responses when stimuli is applied to sensory neurons (shown in green) e.g. touching the Siphon.

The Aplysia Californica is a well studied case of heterosynaptic modulation (e.g. [10, 11, 12]) in invertebrates. Here, plasticity modulatory signals are transmitted by serotonergic interneurons to motor neurons. These modulatory signals increase temporarily the plasticity of their target neurons. Therefore, the synaptic efficacy of the input synapses carrying pulses during the transient modulatory activity will have a stronger potentiation or depression at the modulated neurons. This biological mechanism of neural modulation has served as inspiration for the development of this work and is explained in detail in the following chapters.

Through the modulation of synaptic transmission and plasticity, a neural system acquires the ability to react in different ways according to the originating type and site of stimulation. The stimulation can be *exogenous* (*i.e.* activation of a sensory neuron or a nociceptive pathway by an external stimulus) and *endogenous* (*i.e.* (1) Homeostatic plasticity [13, 14, 15] emerging from the neural activity. (2) Activation of interoceptors by visceral homeostatic mechanisms). Thus, the modulation of synaptic plasticity plays a significant role in the discrimination and prioritization of the neural activity. For instance, stimuli associated with pain or other threatening conditions may elicit higher synaptic plasticity and neural activation (EPSP/IPSP) than non-nociceptive (or not threatening) input stimuli.

Several mechanisms of neuromodulation in both short and long term plasticity have

been studied (*e.g.* [10, 12, 16]), especially in invertebrates given their relatively lower neural complexity when compared to vertebrates. In recent years the understanding of these mechanisms is improving through the development of new theoretical and experimental models. However, most of the complex emergent dynamics resulting from the interaction among the constituent parts of a hetero-synaptical plasticity system (including presynaptic, postsynaptic, inter-neurons and neurotransmitters) are still not fully understood.

In an artificial spiking neural network system without embedded neuro modulatory capabilities, some parts of the learning process have to be separately adjusted through the addition of artifacts (*i.e.* modifying the learning rate after certain set conditions or normalizing the synaptic weights) in order to help the system to converge to the desired solution. While it serves to the purpose of finding a solution to a specific problem it has the drawback that the system is limited to the restrictions imposed by the set parameters.

The modulation of neural behavior through heterosynaptic plasticity mechanisms in biological systems involves complex interactions between neurochemicals and neural dynamics. These interactions allow the nervous system to adapt rapidly to the ever-changing environment while preventing the out-of-control runaway dynamics that emerge during ongoing Hebbian plasticity [17]. This makes plasticity modulation not only a desired feature but a necessary one in artificial neural systems that are aimed to show self adaptation characteristics.

1.2 Methodological Overview

This thesis proposes a system inspired by the biological literature [10, 11, 12] for the modulation of neuronal excitability and synaptic plasticity in spiking neural networks. The modulation mechanisms involved in the proposed system are performed through a heterosynaptic architecture where artificial modulatory neurons induce changes in the dynamics of other neurons in the network.

This thesis starts with the description of the biological and artificial theoretical background that motivates and underlies the development of the proposed system. The description of the system is done in 2 parts: (1) The first part describes how the system is built and how it works by defining and explaining its different building blocks (e.g. neurons, synapses,

modulatory substances) and their corresponding dynamics. (2) The second part of this work proceeds with the implementation of the proposed system in different experimental set-ups including a simulation scenario and the use in real hardware using a robotic platform.

The development of the proposed system has the objective to develop a new theory and architecture of spiking neural networks that

- abstracts the biological complexity while recreates the characteristics of neuromodulated plasticity and excitability of real neural systems
- overcomes the problem of chaotic dynamics and associated non-generalisable *ad-hoc* procedures in previous spiking neural systems
- can be demonstrated to work in practical applications involving the control of autonomous agents and self-adapting systems.

1.3 Research Hypothesis

The Research Hypothesis of this thesis is:

The conventional homosynaptic neural model is unsatisfactory but

- (i) can be extended to heterosynaptic models that explicitly represent chemical regulation of the electrical spiking dynamics that
- (ii) overcome the weaknesses of the homosynaptic model, and
- (iii) can be implemented in practical systems

This hypothesis is developed through a new theoretical architecture and its implementation in simulated and real robots.

1.4 Organization of the Thesis

This thesis starts with Chapter 2 by introducing the theoretical background of Spiking Neural Networks and summarizes the way in which the different dynamics of biological

neurons are recreated in the artificial models. Chapter 3 deals with the current literature regarding hebbian learning, neuromodulation and heterosynaptic plasticity in biological neurons and its modeling and implementation in artificial systems. Chapter 4 describes the theory and design of the proposed novel SNN system with plasticity and efficacy modulation capabilities. This sets the groundwork for the experimental setup described in the Chapters 6 and 7. Chapter 5 describes a strategy for the implementation in software of the proposed SNN system. This strategy includes the description of data structures, processes and their interactions for the representation of the system. Chapters 6 further explores the theoretical and practical applications of the system by comparing two neural circuits applied to a virtual insect brain: traditional non modulated neural dynamics versus Plasticity and Efficacy modulated dynamics. The experimental setup of both neural circuits is described and the results are summarized. Chapter 7 validates the proposed system through the implementation of a neural circuit based on visual pattern recognition and associative learning that is able to control the navigation of a robot. Chapter 8 summarizes the research work presented in this thesis and considers further theoretical and practical applications of the presented system.

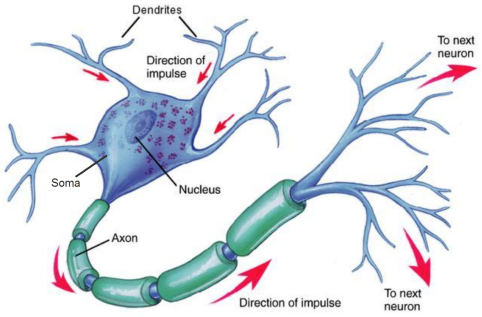
Chapter 2

Artificial Spiking Neurons

This chapter provides an overview of the biological neuron, and of first and second generation artificial neural networks. It then introduces and describes the characteristics of Spiking Neural Networks which are the underlying bio-inspired computational mechanisms used for the proposed systems and experimental investigations within this thesis.

2.1 The Biological Neuron

Before studying in depth the third generation ANNs, it is worth briefly summarizing the biological mechanisms taking place in the communication process between real neurons. Biological neurons (see figure 2.1) use short and sudden increases in their membrane voltage to trigger their activation or firing (action potential) and consequently send out electro-chemical signals, also known as spike trains or pulses, which ultimately represent 'information' sent out by neurons and provide action responses to incoming stimuli. When the spike has been triggered by a sudden increase in voltage (*i.e.* a threshold potential has been reached), the action potential occurring in the cell body of the neuron (or soma) will then traverse down the axon of the neuron, the axon being a long signal carrier that branches out and terminates in the synapse(s).



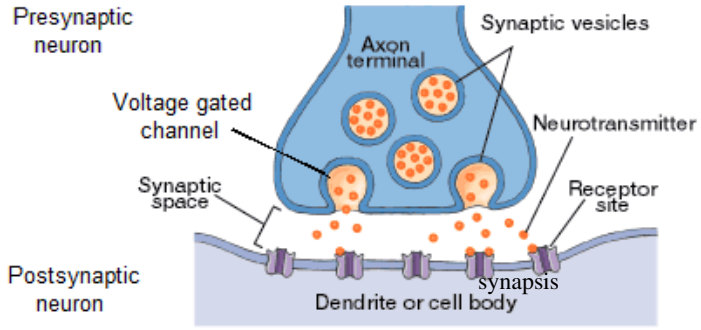
synapsis

https://www.studyblue.com/notes/note/n/biological-foundations-neuron-communication-/deck/1025438

Figure 2.1: A biological neuron

On the presynaptic side of the synapse (the side of the firing neuron) voltage-gated channels (see figure 2.2) open in reaction to the generated action potential, consequently allowing the release of neurotransmitter molecules in the synaptic cleft (the gap between the pre- and the postsynaptic endings). In order for the transmission of information to occur, the released neurotransmitter molecules have to reach their matching receptors on the postsynaptic ending of the gap which can be at any of the following: (1) the dendrites (axodendritic synapse); (2) the soma (axosomatic synapse); or (3) the axon (axoaxonic synapse) of the receiving neuron(s). The incoming neurotransmitters elicit changes in the membrane potential (voltage) of the postsynaptic neuron. This change or perturbation of voltage also known as postsynaptic potential (PSP) can either be positive and excitatory (EPSP) or negative and inhibitory (IPSP). In the human brain a single neuron will receive

postsynaptic potentials from thousands of synapses and when the sum of all these potentials reaches a given threshold value again the neuron will send out a new spike or pulse down its axon. It is also worth noting that after a spike has been sent the neuron will enter a short time period (generally between 1ms and 10ms) in which it cannot send spikes again, this is known as the *refractory period*.



https://ehumanbiofield.wikispaces.com/file/view/F02_04.gif/41855553/F02_04.gif

Figure 2.2: The synapse

2.2 Artificial Neural Networks

Artificial Neural Networks (ANN) consist of a simplified simulation of the computational features observed in biological neural systems, using mathematical models that are recreated by artificial mechanisms such as algorithms implemented in software, VLSI and digital logic circuits. In other words, ANN are simplified computerized neurons, which ideally will replicate the biological dynamics of real neurons as closely as possible whilst still being practical from a computational perspective.

The main objective of the ANN is to build machines capable of showing abilities similar to those observed in the brain, including the following: learning, fault tolerance, generalization of knowledge, self-organization, flexibility in handling fuzzy information and parallel processing [18]. In addition, from a biological point of view, ANNs (especially third generation models) which are accurate enough to describe the neurons in animal brains are not only valid computational models which have found a variety of applications, they have also aided the research for understanding brain functions and the information processing mechanisms in biological nerve cells.

Needless say, the modelling of the neural activity that takes place in the human brain has proven an incredibly challenging as well as fascinating task. There are millions and millions of neurons interconnected with each other and sending output signals to each other in order to process incoming input signals and decide on an appropriate response action. The road to attempt to replicate through computer modelling the neural activity of the human brain started several decades ago and is certainly no where near its end, however big steps have been taken towards more powerful and biologically accurate models which are able to incorporate appropriate responses to given stimuli in artificial agents: a key component of future artificial intelligence.

The first generation of ANN consists of a simple model by which a neuron sends out a binary 'high' signal if the sum of its weighted incoming signals rises above a given threshold value. Neurons modelled in such way have found several applications including multi-layer perceptron and Hopfield nets. However, there are several limitations with the first generation models, above all the fact that they only give binary outputs.

ANN of second generation are suitable for analogue input and output as they replace the threshold (step) function to compute their output signals with a continuous activation function (*e.g.* sigmoid or hyperbolic tangent). Neurons modelled through continuous activation functions have been applied in feed-forward and recurrent ANN. Neurons of the first two generations do not model the neuron based on its individual pulses but calculate a "rate coding" for the neuron instead. Basically the output signals from a neuron typically lie between 0 and 1, which is a normalized firing rate or frequency for that neuron, within a given time period. This is the rate coding, where a higher rate of firing correlates with a higher output signal. In other words the rate coding for a neuron is an averaging of its spikes over a time window, which can therefore be calculated by computing a sufficient number of iterations. Second generation ANN are more biologically realistic as the continuous activation functions that characterize them are better able to model the base firing-rate (the intermediate frequency of pulsing) of real neurons. However they still hold the limitation that the 'answer' (or response action) of the network of neurons to the input values can only be known after a sufficient number of computer iterations are run, hence the temporal

dimension of communication cannot be incorporated.

The third generation of ANN (also known as spiking or pulsed neural networks) raises once again the level of realism by incorporating the spatial-temporal response in the neural communication dynamics, which represents a huge step closer to how real (biological) neurons work. This is possible as the third generation on ANN supersedes the concept of rate coding by modelling the actual individual spikes of a neuron.

2.3 Spiking Neural Networks

The third generation of neural networks is mainly based on Threshold-Fire models (*e.g.*, Integrate and Fire model [19] [20]) and the pulsed neuron model of Hodgkin-Huxley (1952)[21]. The latter was proposed in order to incorporate the neurobiological properties of nerve cells and also to describe the generation and propagation of the action potential and explain its main properties from a mathematical point of view in terms of differential equations. Spiking Neurons, also known as *pulse-generating neurons*, are mainly known for their ability to encode information in the form of pulses or spikes over time. This neuron model, in a similar fashion to its predecessors of the first and second generation, is able to process multiple input signals (stimuli) from other neurons, and when the membrane potential of a stimulated (post-synaptic) neuron reaches a given threshold, its output generates an electric potential (pulse) or a train of pulses.

As stated above, the key conceptual feature of third generation ANN is the fact that they model the individual pulses or spikes of neurons and thus take a huge leap closer to approximating the biologically complexity of real neurons. Networks of spiking neurons are much more powerful than the previous ANN, which were characterized by networks that were not spiking, since the firing rate could only be averaged over a given period of time. With Spiking Neural Networks (SNN) the modelling of individual spikes enables computer models - and the artificial agents operated by those models - to incorporate a spatial-temporal dimension into their behaviour, a huge step closer to the communication and behaviour of artificially intelligent robots. However, by modelling individual spikes, SNN require a much more accurate understanding and implementation of the synaptic plasticity, *i.e.*, the way two neurons actually communicate and associate with each other,

which forms the basis of most models of learning and development of ANN. A brief overview of the different types of SNN models is presented in Section 2.6.

2.4 Rate and Pulse Codes

The representation of information based on pulses or spikes can occur in different ways taking into account the following factors: the average rate at which pulses are generated; the specific firing time (the delay between spikes is used to represent information); groups of pulses in a neuronal population and the synchronization of pulses between neurons (phase locking). Based on experimental evidence from the study of biological neurons, two generalized information coding mechanisms have been proposed in SNN, known as 'rate code' and 'pulse code' [22]. The rate code refers to the representation of information according to the average number of spikes generated in a given time interval. Gerstner and Kistler (2002) [22] differentiate between three types of average: average over time, average through several experimental repetitions, and average over a neuronal population. On the other hand, the pulse code uses the time between each spike to represent the information that is transmitted from one neuron to another. In the case of biological neural networks, there is some controversy regarding the type of code used for signalling between neurons. Signalling through the rate code has been observed in laboratory since the beginning of the 20th century (*e.g.* stretch receptor in a muscle spindle [23], touch receptor in the leech [24]). However, there are sufficient arguments to believe that the rate code mechanism does not offer sufficient computational speed required for processing information in the central nervous system (CNS), where certain perceptual (*e.g.* facial recognition [25]) and complex cognitive processes require a much faster reaction time than the duration involved in computing the average of a set of spikes in a period of time.

2.5 The Spatio-temporal Dimension of SNN

As mentioned above, a neuron is activated when its membrane potential reaches a certain depolarization or firing threshold. This principle is modelled by all the three generations of ANN. However, in the ANN of first and second generation the activation of a neuron

is modelled through computer iterations each representing an independent and complete potential scenario in which a given firing rate (represented through the connection weights) is computed. The computed firing rate may or may not be high enough to reach the activation threshold of the neuron. By contrast with ANN of third generation, the evolution of the membrane potential over time is modelled as a continuous process in which neurons can be stimulated and may reach the firing state at any time. That is, there is no state in which all inputs are first computed and then compared against the activation threshold; the activation function can be triggered at any time as a result of the continuous summation of the continuously incoming spikes.

The summation of the spikes can occur either in space or time. The Spatial summation occurs when two or more incoming spikes reach the activation area of the post-synaptic neuron (known as axon hillock in biological neurons) at almost the same time, jointly affecting the neuron membrane potential. Temporal summation occurs when the spikes arrive at different times but close enough so that the group of sub-threshold spikes (*i.e.* EPSP) contributes to the generation of the same action potential. The generation of an action potential is considered as an *all-or-nothing* activation mechanism, meaning that this will only be triggered once the membrane potential reaches a certain threshold because of the depolarization elicited by the incoming spikes.

2.6 SNN Models

This Section presents a brief overview of the different types of SNN models.

2.6.1 Threshold-Fire models

Threshold-Fire models are based on the temporal and spatial summation of incoming synaptic potentials in the membrane potential of the postsynaptic neuron. If the membrane potential reaches a firing threshold, then the neuron is set to fire (an action potential is triggered) and the membrane potential is reset to its resting value. Within the most used and well known Threshold-Fire models are the Leaky-Integrate-and-Fire (LI&F) and the Spike-Response model (SRM) [26][20]. The dynamics of Threshold-Fire models are much

simpler than other third-generation ANN models. For instance the LI&F uses a single linear differential equation to describe the evolution of the membrane potential, while in the SRM the membrane potential at time t is calculated as an integral over the past. This makes it possible to implement SNN in computing applications of practical use without compromising the performance of the system due to high processing requirements.
Given that the system proposed in this thesis is based on the LI&F model, a detailed explanation of the formulae describing the LI&F model, as well as a comparison with the proposed system, are included in Appendix A.

2.6.2 Conductance-Based models

Conductance based models are based on the simulation of the behaviour of ionic channels in biological nerve cells. The activation/deactivation of these channels (opening and closing) determines the conductance of the cell membrane and consequently of the membrane potential of the cell, influencing the triggering of an action potential when reaching a threshold value. The mechanism of activation by ionic currents is described by a set of differential equations. The complexity of these varies according to the number of ion channels which are taken into account. The most well known conductance based model is the one proposed by Hodgkin and Huxley in 1952[21], which was awarded with the Nobel Prize in Physiology or Medicine in 1963.

Hodgkin-Huxley model

It is known that the cell membrane has a given electrical capacitance and that it separates solutions of different ionic concentrations. Consequently, there is an electric potential difference (voltage) between the inside and outside of the cell: The concentration of potassium is greater inside than outside the cell. The opposite occurs with sodium, where the concentration is greater outside than inside the cell. Hodgkin and Huxley demonstrated that sodium and potassium make important contributions to the ionic currents. They predicted and proved that the initiation and propagation of the action potential depends on the external concentration of sodium. The Hodgkin-Huxley model is based on the idea that the electrical properties of a nerve cell membrane can be modelled by an equivalent

electrical circuit. By using Kirchoff's Laws, the behaviour of the equivalent electric circuit can be described by a set of differential equations for the total current flowing through the cell membrane [21].

2.6.3 Compartmental models

In a similar way to Conductance-Based models, compartmental models take into account the behaviour of ion channels to determine the conductance of the cell membrane. However, these models extend the level of detail in the simulation of biological neurons, by taking into account the structure of the dendritic-tree and calculating the propagation of each postsynaptic potential from the receiving dendrite to the Axon Hillock. The system of equations describing this model is more complex than in conductance models and requires high computing capacity in order to carry out a simulation. Given their level of accuracy in the simulation of biological neurons, compartmental models are more suitable for simulations of brain processes, nerve cell biology and related research in the field of neuroscience. However, given the mathematical complexity involved in calculating the current potential transmitted through each ion channel, it is necessary to use specialized hardware in order to implement these SNN models.

2.7 Summary

Inspired by the behaviour of biological neurons and their connections (synapses) the research on neural networks has been a very important and long-standing field within computer science. This chapter has provided an overview of the biological neuron and artificial neural networks of first and second generation. In addition, various third generation (Spiking Neural Networks) models have been described. This constitutes the beginning of the theoretical framework required for the investigation carried out in the next chapters.

Chapter 3

SNN: Learning Mechanisms and Neuromodulation in Artificial Neural Systems

This chapter introduces the concepts of hebbian-based plasticity in biological and artificial neural systems. It then describes one of the key implementations of this concept, known as the spike timing dependent plasticity rule. A description of architectures for associative learning is then presented, which constitutes the foundation of the experimental investigations carried out.

3.1 The Hebb Rule: a theory of synaptic learning

In his book titled "The Organization of Behaviour" (1949)[27], Donald Hebb postulated a theory of synaptic plasticity widely known today as the Hebb rule or hebbian learning. In his theory, Hebb describes how the synaptic strength or efficacy between a presynaptic neuron A and a postsynaptic neuron B is strengthened if the firing activity of A contributes to the firing activity of B. In other words, if the spikes transmitted by A tend to activate or are temporarily close to (preceding) the firing of B, then the efficiency (strength) of A activating B is increased.

Since its postulate, the Hebb rule became a new paradigm for theories of learning in both biological and artificial neural systems where synaptic plasticity is driven by the joint activation between pre- and postsynaptic neurons [28, 29].

From a machine learning perspective, hebbian learning is an algorithm that changes the weights of the connections between interconnected artificial neurons. The change that is applied to the weight of the link (or pairwise connection) between a sending and a receiving neuron, is computed according to a given interval in which both neurons need to be activated. The agent and robot described in chapters 6 and 7 of this thesis are examples of the implementation of hebbian learning in autonomous systems.

The hebb rule is considered an unsupervised learning mechanism given that the system does not explicitly receive any feedback indicating whether or not the resulting synaptic change is useful [30, 31, 9]. Instead, the connectivity strength (synaptic efficacy) in a neural network implementing hebbian learning, will emerge according to the coincidences between the firing patterns of the neurons that are interconnected *i.e.* the resulting synaptic efficacy will depend on the statistical correlations between neurons [32, 31, 9].

3.2 Spike Timing Dependent Plasticity: An implementation of hebbian learning for SNN

Spike Timing Dependent Plasticity (STDP) [20, 33, 34] is considered a temporally precise implementation of hebbian learning for synaptic plasticity [9]. It has been hypothesised

that STDP is one of the underlying mechanisms for memory formation and that it plays a role in the refinement and consolidation of weight dynamics and structure in neural circuits during brain development [35, 36].

In this thesis, the STDP model proposed by Gerstner et al. [20] has been used as the underlying plasticity mechanism for the proposed heterosynaptic system described in chapter 4 and is also at the core of the implementation of the experimental neural circuits in chapters 5 and 6. For this reason, this chapter focuses on the STDP rule and the learning mechanisms associated with it.

In STDP, the synaptic efficacy is adjusted according to the relative timing of the incoming presynaptic spikes and the action potential triggered at the postsynaptic neuron [20, 32, 37, 38, 39, 40].

This can be expressed as follows:

1. The pre-synaptic spikes that arrive shortly before (within a given range or learning window) the post-synaptic neuron fires are considered as contributors to the depolarization of the post-synaptic neuron. Consequently, these spikes reinforce the efficacy (weights in terms of artificial neurons) of their respective synapses. This is known as *pre-before-post-timing*.

2. The pre-synaptic spikes that arrive shortly after (within a given range or learning window) the post-synaptic neuron fires are not considered as contributors to the action potential of the post-synaptic neuron. Consequently, these spikes weaken the efficacy of their respective synapses. This is known as *pre-after-post timing*

The following formula [20] describes the weight change $\Delta\omega_j$ of a synapse through the STDP model for pre-synaptic and post-synaptic neurons represented with j and i respectively. The arrival times of the pre-synaptic spikes at the post-synaptic neuron are indicated by t_j^f where $f = 1, 2, 3, ...N$ enumerates the pre-synaptic spikes. t_i^n with $n = 1, 2, 3, ...N$ counts the firing times of the post-synaptic neuron i:

$$\Delta w_j = \sum_{f=1}^{N} \sum_{n=1}^{N} W(t_i^n - t_j^f) \tag{3.1}$$

Let $\Delta t = t_i^n - t_j^f$ .The connection weight resulting from the combination of a pre-synaptic spike with a post-synaptic action potential is given by the function [20, 34, 37]:

$$W(\Delta t) = \begin{cases} A_+ \exp(-\Delta t/\tau_+), \text{ if } \Delta t > 0 \\ -A_- \exp(\quad \Delta t/\tau_-), \text{ if } \Delta t < 0 \end{cases} \tag{3.2}$$

The parameters A_+ and A_- indicate the amplitude of the potentiation and depression of the synaptic weights respectively. τ_+ and τ_- are the time constants that describe the exponential shape of the learning window.

The form of the learning window between neurons i and j, w_{ij}, not only determines whether the time difference between a presynaptic and a postsynaptic pulse results in potentiation or depression of the synapse but also the amplitude of its change Δw_{ij}. Figure 3.1 illustrates a STDP learning window showing the time intervals in which long term potentiation (LTP) and long term depression (LTD) [1] occur.

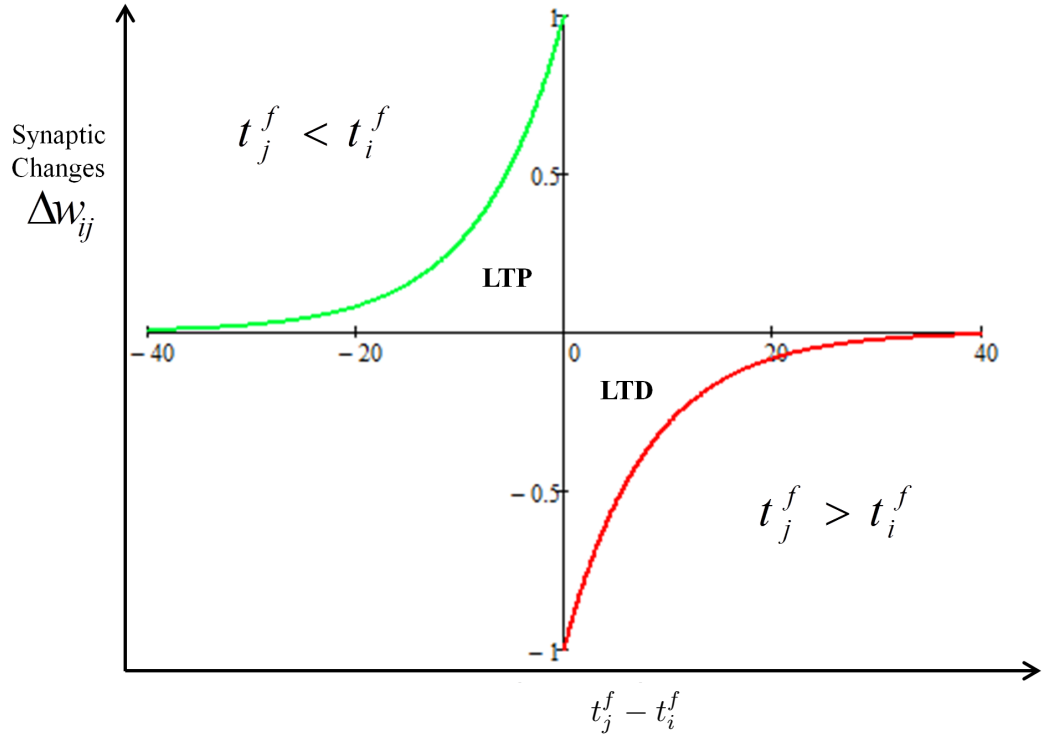


Figure 3.1: Learning window with symmetrical LTP and LTD intervals.

[1] LTP and LTD: are terms used in neuroscience to refer to the persistent increase and decrease of the synaptic strength (synaptic weight) respectively.

As seen in Figure 3.1 at the LTP interval, the presynaptic spike at time t_j^f precedes a postsynaptic action potential at time t_i^f. The opposite occurs at the LTD interval where t_j^f arrives after t_i^f. In both LTP and LTD cases, the time difference between the presynaptic and postsynaptic spikes must be within the intervals set by the parameters of the learning function.

3.3 STDP Applied at the Network Level Using Associative and Classical Conditioning Learning

There is increasing research (*e.g.*, [41, 42, 43, 44, 45]) demonstrating that the third generation of artificial neural networks is emerging as a potential computational tool to control autonomous systems which exhibit intelligent behaviour in terms of learning and adaptation to the environment.

Experimental results have demonstrated that different types of associative learning including operand and classical conditioning (*i.e.* Pavlovian extinction, partial conditioning, inhibitory conditioning) can be implemented successfully using SNN. [41, 46]. Given that the experimental setups presented in the following chapters of this thesis are based on associative learning and STDP, this Section will introduce the building blocks necessary for the implementation of the experiments.

3.3.1 Associative Learning Based on STDP

Associative learning is understood as a learning process by which a stimulus is associated with another. In terms of classical conditioning, learning can be described as the association or pairing of a conditioned or neutral stimulus with an unconditioned (innate response) stimulus, where association means that the neutral or conditioned stimulus acquires the ability to elicit the same response or behaviour produced by the unconditioned stimulus.

The pairing of two unrelated stimuli usually occurs by repeatedly presenting the neutral stimulus shortly before the unconditioned stimulus that elicits the innate response. When

talking about classical conditioning [47] in animals, the word 'shortly' refers to a time interval of a few seconds (or in some cases a couple of minutes). On the other hand, at the cellular level and in terms of STDP, the association of stimuli encoded as synaptic spikes occurs in short milliseconds intervals [20].

The simplest form of associative learning at the spike level occurs pair wise between a pre- and a postsynaptic neuron. This is illustrated below in Figure 3.2:

(a) (b)

Figure 3.2: a) The spike emitted by the presynaptic neuron j arrives at the synapse at time t_j^f. b) A spike is generated at the postsynaptic neuron i at time t_i^f .

Taking the spikes depicted in Figure 3.2, the function $W(t_i^f - t_j^f)$ determines whether the spike at t_j^f contributed or not to the generation of the postsynaptic spike t_i^f. According to formula 3.2 in this chapter, if $(t_j^f < t_i^f)$ the weight change of presynaptic neuron j will be positive and relative to the difference between t_j^f and t_i^f. In contrast, if $t_j^f > t_i^f$ the weight or efficacy of the presynaptic neuron j will be decreased.

3.3.2 Associative Topologies for Classical Conditioning

In order to create a neural circuit of Spiking neurons that allows the association of an innate response to a neutral stimulus, it is necessary to have at least the following elements:

1. A receptor or sensory input for the unconditioned stimulus U.

2. A receptor or sensory input for the conditioned or neutral stimulus C.

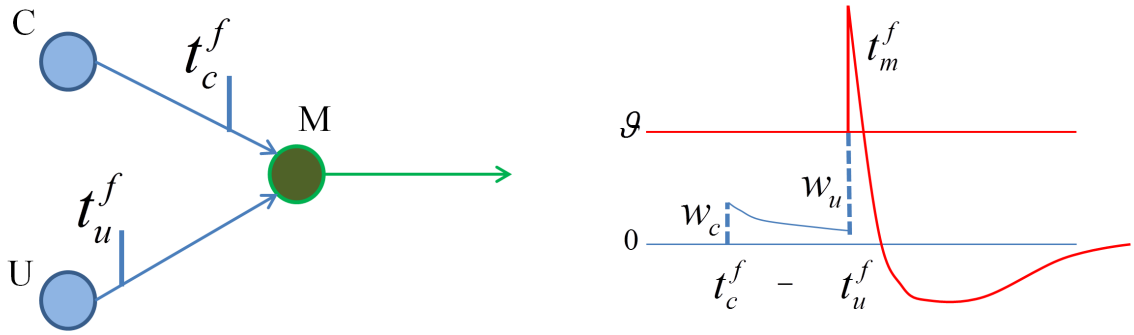
3. The motoneuron or actuator, which is activated by the unconditioned stimulus M.

For U the unconditioned stimulus must be able to elicit an immediate reflex-response (action potential) in the postsynaptic motoneuron. Thus the synapse efficacy of the presynaptic neuron U (unconditioned input neuron) must be greater or equal to the activation

threshold Th_m of the motoneuron M, *i.e.* $(W_u \geq Th_m)$, in order to elicit a postsynaptic action potential with a single presynaptic spike.

For C the conditioned stimulus must be able to elicit a PSP (postsynaptic potential) in the postsynaptic motoneuron M. Thus a synapse between the presynaptic neuron C (conditioned input neuron) and the postsynaptic motoneuron M must exist.

Given the elements U, C and M, the following topology illustrated in Figure 3.3 could be used for a simple associative neural circuit:

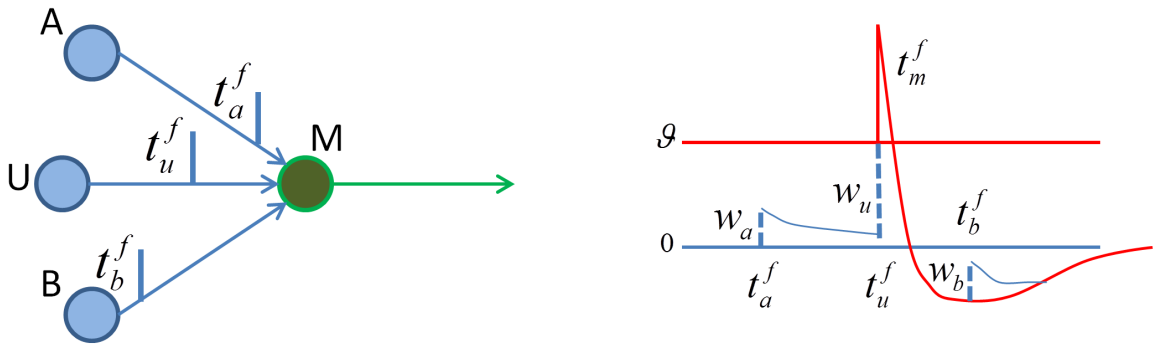


(a) (b)

Figure 3.3: a) Spikes emitted by input neurons C and U reaching the synapse with postsynaptic motoneuron M at time t_c^f and t_u^f respectively. b) The spike emitted by C elicits an EPSP (excitatory postsynaptic potential) of amplitude w_c (left dashed line) at time t_c^f. At time t_u^f the spike emitted by U elicits an EPSP of amplitude w_u (right dashed line) that reaches the threshold ϑ triggering an action potential (thick line) at the postsynaptic Motoneuron M.

The neural circuit in Figure 3.3(a) illustrates the two input neurons C and U each transmitting a pulse to postsynaptic neuron M. As shown in 3.3(b) the unconditioned stimulus transmitted by U triggers an action potential at time t_m^f shortly after the EPSP elicited by C at time t_c^f. According to eq. 3, $t_m^f > t_c^f$ (EPSP elicited by C preceding spike at M) with $\Delta t > 0$ the synaptic efficacy between C and M, *i.e.* (Δw_c), would be increased relative to the difference $t_c^f < t_m^f$ - and the parameters τ_+, τ_-, $+A$, $-A$ set in eq. 3, which represents the learning interval or learning window [20, 34, 37] of STDP. Given that the STDP learning window allows both LTP and LTD, the simple topology illustrated in Figure 3.3(a), can be extended giving it the ability to associate stimuli (either as causal or acausal) from multiple input neurons with an unconditioned or innate response.

The following topology illustrated in Figure 3.4 includes three input neurons A, B and U. Neurons A and B receive input from two different neutral stimuli, while U receive input from a unconditioned stimulus.



(a) (b)

Figure 3.4: a) Spikes emitted by input neurons A, U and B reaching the synapse with postsynaptic motoneuron M at time t_a^f, t_u^f and t_b^f respectively. b) The spike emitted by A elicits an EPSP of amplitude w_a, which is followed a few milliseconds later $(t_u^f - t_a^f)$ by an action potential triggered by U at time t_u^f. The pulse emitted by B arrives shortly after the action potential in M at time t_b^f.

The circuit in Figure 3.4(a) is able to associate two neutral input stimuli with Motoneuron M. As shown in Figure 3.4(b) the unconditioned stimulus transmitted by U triggers an action potential at time t_m^f shortly after the EPSP elicited by A at time t_a^f. Following eq.(3) with $t_m^f > t_a^f$ (EPSP elicited by A preceding spike at M) with $\Delta t > 0$ the synaptic efficacy between A and M, *i.e.* (Δw_a), is potentiated in terms of $t_a^f - t_m^f$. On the other hand, the EPSP elicited by B at time t_b^f occurs during the relative refractory period that followed the action potential triggered in M. Since $t_b^f - t_m^f$ is on the LTD side of the learning window, the synaptic efficacy between B and M, *i.e.* (Δw_b), is depressed in terms of $t_b^f - t_m^f$. In Figure 3.4(b), if the incoming spikes in M continue following the same timing pattern, in the long term the synaptic efficacy of w_a and w_b would reach their maximum and minimum values respectively (values boundaries depending on the implementation).

3.4 Characteristics and limitations of STDP

As seen so far, the traditional STDP plasticity rule provides an interesting bio-inspired mechanism for unsupervised learning in neural circuits. The use of STDP as an underlying plasticity mechanism has been extensively and successfully tested in several experimental setups [20, 32, 37, 48]; demonstrating that with STDP, a neuron is able to discriminate through the time dimension between the real contributors to its activation and the noisy neurons whose firing patterns are not synchronized or correlated with its own activation. In other words, STDP allows a neuron to decide which presynaptic neurons are worth listening to and which ones should be given less priority or possibly be completely ignored [49]. Moreover, through the combination of associative network architectures as described above, a SNN system implementing STDP is able to create associations between multi-modal stimuli (*i.e.*, from different types of sensory inputs). This allows the engineering of sophisticated neural circuits that can be applied to control the behaviour of autonomous systems (as will be demonstrated in the next chapters).
Nevertheless, in spite of these desirable features, STDP also has some limitations that need to be considered when implementing neural circuits on a greater scale. This thesis will focus on two known issues found in STDP based systems:

1. Runaway dynamics.

2. Timing as single source of spike information.

3.4.1 Runaway dynamics

Out of control or runaway dynamics [17] refer to the positive feedback loop that emerges from the mechanisms that rule the induction and amplification of synaptic changes in hebbian based plasticity. As already explained, in order for the induction of synaptic change to take place, it is required that the presynaptic spikes arrive shortly before the activation of the postsynaptic neuron. These presynaptic spikes elicite excitatory postsynaptic potentials that keep adding to the postsynaptic membrane potential as they arrive. Therefore, the presynaptic spikes coming from a synapse with stronger synaptic efficacy will have a higher probability of bringing the membrane potential of the postsynaptic neuron to reach

its firing threshold and make it fire. Consequently, the already stronger synapse that contributed to the activation will be reinforced and hence its probability of activating again the postsynaptic neuron is increased. Over time, the synaptic efficacy (or weight) will reach its maximum value or ceiling.

A similar situation occurs with the long term depression of synapses. The synapses with lower synaptic efficacy have a lower probability of activating a postsynaptic neuron given that the elicited postsynaptic spikes are mostly not large enough to make a significant contribution to the postsynaptic membrane potential and to elicite a postsynaptic action potential. Therefore, over time the already weaker synapses tend to be depressed to a minimum or floor value as their spikes do not have the chance to activate the postsynaptic neuron.

One of the critical issues with the runaway dynamics in hebbian plasticity is that not only does it affect the system at the level of individual synapses, but it creates chains or cascades affecting the entire neural circuit [50]. For instance, as it is observed in some of the experimental setups carried out in this thesis, in neural circuits implementing associative and classical conditioning architectures, the stronger over-potentiated synapses originating from conditioned sensory (neutral) neurons gain the ability to activate their associated motoneurons (behaviour) acting as unconditioned sensory neurons. Whilst this is an expected behaviour in a conditioned system, the issue arises when the conditioned neurons in addition to acting as stronger activators of motor behaviour acquire the ability to reinforce other neutral sensory neurons creating new conditioning behaviour between neutral stimuli.

As described so far, runaway dynamics may render the behaviour of a neural circuit unpredictable or unstable. Moreover, it may have a negative impact on the computational abilities of the system [17].

3.4.2 Timing as single source of spike information

As mentioned before, a neuron implementing STDP is able to identify the coincidences between its own firing time and the firing time from presynaptic neurons. However, apart from the timing of the spikes there is no other information about the event that triggered

the firing of the incoming spike or about the neuron that originated the incoming spike. For instance, the standard STDP implementation neglects any information regarding reward, success, punishment and novelty [9].

This limitation is not only specific to STDP but it is a general characteristic of hebbian-based plasticity systems. The reason for this is that the canonical hebbian postulate is solely focused on the relationship between pre- and post-synaptic activity whilst neuromodulation is completely out of the scope of its implementation.

From a neuro-engineering perspective, having a plasticity system that is solely based on timing information brings along some technical challenges that arise when designing and implementing an artificial neural system. One of these key challenges is the undesirable reinforcement of a synapse resulting from the hebbian coactivation between two temporarily coinciding spikes. This is an issue that emerges from the lack of a feedback signal in response to the synaptic change (*e.g.* reward, punishment, error). The missing feedback in response to the change in the system is an inherent characteristic of unsupervised learning systems that in certain cases requires complex workarounds in order to guide or regulate the learning of the system.

3.5 Homosynaptic Plasticity and the Need of a Heterosynaptic Approach

The STDP dynamics and the architecture of neural circuits described so far, are based on a type of synaptic plasticity known as homosynaptic plasticity. This form of synaptic learning is the most widely observed in biological systems and has been extensively studied in biological and artificial models.

In homosynaptic plasticity the changes that occur in the synapse between a presynaptic neuron and a postsynaptic neuron are induced by the firing activity of the presynaptic neuron only. In other words, if a postsynaptic neuron fires, only the synapses that were active (*i.e.* transmitting spikes from a presynaptic neuron) shortly before (inducing hebbian potentiation) or after the activation (inducing hebbian depression), will be changed.

Therefore, this type of synaptic modification is also known as input specific plasticity. Homosynaptic plasticity by definition is related to hebbian learning and is considered to be the neural basis for associative learning and memory formation in the nervous system [17]. As previously described, there are issues associated with hebbian learning systems which are inherent to homosynaptic plasticity. These issues pose questions about the mechanisms used by the nervous system to maintain stability of neuronal activity and to regulate the formation and consolidation of memories without the degradation caused by runaway dynamics [50, 51, 52]. Heterosynaptic plasticity has been identified as a potential mechanism for the homeostasis of hebbian plasticity through the normalization and regulation of synaptic changes [53, 54, 55] and for its ability to prevent runaway dynamics [56, 17]. Heterosynaptic plasticity also known as Heterosynaptic modulation [57, 52] refers to the changes in the synaptic behaviour between a presynaptic neuron and its postsynaptic target induced by another (modulatory) neuron.

In contrast to the homosynaptic mechanism where plasticity results from the joint spiking activity between the presynaptic and the postsynaptic neurons, the heterosynaptic form does not necessarily require the activity of both presynaptic and postsynaptic neurons in order to trigger changes in a synapse. Heterosynaptic modulation has different characteristics depending on: (1) The site of action that the modulatory signal is targeting (*i.e.* pre- or postsynaptic), (2) The neurotransmitters that serve as modulatory signal and (3) the effect of the modulatory signal on the target site [57].

1. At the presynaptic side, modulation occurs when the modulatory signal reaches the presynaptic neuron affecting the dynamics underlying neurotransmitter release in response to action potentials [57]. Early experimental evidence has demonstrated that heterosynaptic modulation is involved in presynaptic inhibition (*i.e.* reduction of the amount of released neurotransmitter) in invertebrates via Gabaergic pathways [58] and Dopaminergic pathways combined with FMRF [59], as well as in vertebrates via the regulation of voltage-dependent calcium channels [60] and through GABA and opioid receptors in nociceptive pathways in the spinal cord [57]. Heterosynaptic plasticity at the presynaptic side is also considered to be one of the cellular mechanisms underlying synaptic facilitation (*i.e.* sensitization and dishabituation) in Aplysia Californica [61] via serotonergic pathways [62, 63].

2. At the postsynaptic side, heterosynaptic modulation occurs when the modulatory signal reaches the postsynaptic neuron affecting its mechanisms of response to presynaptic spiking activity [57]. Postsynaptic modulation of hebbian LTP (via noradrenergic pathways [64] and LTD (via Muscarinic receptors) has been observed in some neuronal populations in the hippocampus [65] and in the neocortex [66]. Heterosynaptic modulation of postsynaptic neurons is also the mechanism underlying the implementation of the system presented in this thesis. This will be introduced in the next Section.

3.6 Modelling SNN with Heterosynaptic Activity

Advances in neuroscience are making it possible to overcome the difficulties that arise when studying the dynamics of pre- and postsynaptic spiking activity combined with the effects of different types of neuromodulators on synaptic plasticity. Along with increasing experimentation in the biological field, theoretical and mathematical models of neuro modulated hebbian plasticity are being hypothesized. The majority of these theoretical plasticity models and rules make use of modulatory signals to represent reward or in some models, novelty or surprise. The use of rewards as a mechanism of regulated hebbian plasticity [67, 68, 69] is characterized by the implementation of a success or reward signal which is triggered on the completion of an action (*e.g.* finding food) during a trial. In reward based systems the reward signal always arrives with a certain delay after a given success action has been completed (*e.g.* Reward-modulated STDP or R-STDP [70]). This affects the plasticity in the synapses where neural activity occurred before the triggering of the reward signal. Because of the delay between the signal and the preceding spiking activity, this modulatory approach has the difficulty involved in keeping a record or tagging of the synaptic activity (for time intervals that can sometimes last several milliseconds) which must be processed after the subsequent reward signal has arrived. Solutions to overcome the difficulties associated with the retroactive modulation [71] mechanism also called 'the distal reward problem' [70] have been proposed in several studies (*e.g.* Sutton

and Barto (1998) [72], Baxter and Bartlett (2001)[73], izhikevich (2007)[70]) using different hypothetical and experimental approaches.

On the other hand, in systems where learning is driven by novelty [74], the synaptic change is modulated by the introduction of a novelty signal that triggers the activation of the hebbian mechanism at the time when the signal arrives. In contrast to the reward based approach, the novelty or surprise signal does not impose the restriction that is has to always arrive after the input activity that caused the hebbian activation. Instead, the signal works as a novelty-gated learning mechanism that is sensitive to the timings between the hebbian activation and the arrival of the novelty signal which makes the plasticity mechanism effective.

The use of reward and novelty signals as a modulation mechanism of hebbian plasticity has been inspired and corroborated by research in the nervous system where experimental evidence has shown the association of neuromodulators (*e.g.* dopamine, acetylcholine, serotonine and noradrenaline) with the reward-related reinforcement of actions and the creation of new memories [75, 76, 77, 78, 79, 80]. However, despite the fact that several neurotransmitters have been associated with different neuromodulatory functions through experimental observations, at this stage it is not possible to map specific one-to-one functionalities to each neurotransmitter given that their elicited behaviour may vary (and the same neurotransmitter can sometimes even have opposite effects) between different types of neurons and synapses. Therefore, a standard unified computational model of neuromodulated hebbian plasticity is still not available in the current state of the literature [9].

3.7 A Novel SNN system with embedded Heterosynaptic capabilities

The system proposed in this thesis presents a novel approach for the modulation of two mechanisms which are at the core of the dynamics of spiking neural networks: (1) modulation of synaptic plasticity and (2) Modulation of neuron excitability. In contrast to reward and novelty based learning models, the mechanism described in this research

does not explicitly involve the use of reward or novelty signals for the regulation of the plasticity process. Instead, the neuron excitability as well as the amplitude of change in the synapses is determined by individual and local neuronal-homoeostatic processes that are manipulated through the introduction of non-spiking modulatory neurons. The modulatory neurons in this system are capable of inducing changes in the behaviour of their target neurons through the transmission of modulatory signals. Whilst the hypothetical modulatory signals do not induce any change in the membrane potential of the target neurons, they tell the affected neurons how to respond to incoming presynaptic spikes and to what extent hebbian plasticity is effective in the synapses that obey (have affinity) to the modulation process. The novel system is heterosynaptic by definition because the mechanisms of excitability and plasticity in a given synapse are affected by a modulatory neuron that is not at either presynaptic or postsynaptic end of the affected synapse. The next chapter will describe this proposed system in much deeper detail.

In summary, the novelty of the proposed system includes the implementation of the following features:

- A modulatory factor representing a Plasticity Modulatory Substance that is individual (concentration varies from neuron to neuron) and self-regulated (homeostatic). The hypothetical and non-specific substance (not related to a specific neurotransmitter and neuromodulator) controls the amplitude of change in induced hebbian plasticity (STDP).

- A modulatory factor representing an Excitability Modulatory Substance that is also individual and self-regulated. This hypothetical and non-specific substance controls the amplitude of induced postsynaptic potentials.

- Selective modulation of synapses: The modulation of plasticity and excitability in target synapses is subject to their affinity with the corresponding modulatory substance.

- Modulatory neurons with embedded membrane dynamics (as regular spiking neurons) that send plasticity and excitability modulatory signals.

The applicability of the proposed system is demonstrated by two experiments which have been carried out and are described in chapters 6 and 7.

3.8 Summary

This chapter started with the introduction of the Hebb postulate of synaptic plasticity better known as 'Hebbian learning' or 'Hebb rule'. The Hebb rule was described as a learning mechanism in biological neural systems that determines the change of efficacy in a synapse. In computational terms the Hebb rule was described as an unsupervised learning mechanism for the change of the connectivity strength in artificial neural networks.
Consecutively, the STDP rule was introduced as an important model of synaptic plasticity that is based on Hebbian learning. The implementation of STDP was described along with some of its applications in artificial systems (e.g. Spiking Neural Networks used in autonomous systems). This chapter also described some of the issues that arise when STDP is used as the underlying learning mechanism in SNN. Most of the issues mentioned in this chapter were related to the 'traditional' homosynaptic neural architecture and the emergent runaway dynamics which are an inherent property from the underlying hebbian learning mechanism.
Given the afore mentioned, an overview of the approaches to overcome runaway dynamics in STDP was presented. This included the use of reward and novelty signals as an extended mechanism to control the learning activity in SNN. Finally, the heterosynaptic approach of neuro-modulatory activity developed in this thesis was introduced as a novel bio-inspired solution to overcome the weaknesses of hebbian learning and homosynaptic neural architectures.

Chapter 4

Theory and Design of a Spiking Neural System with Heterosynaptic Capabilities

This chapter develops the theory for the proposed mechanisms of Heterosynaptic modulation of synaptic plasticity and neural excitability in Spiking Neural Networks. The first part of this chapter briefly describes the dynamics of both plasticity and excitability modulation. The second part focuses on plasticity modulation starting with the introduction and implementation of an extended 3-factor rule for hebbian learning. This is followed by the description of the mechanisms of up and down modulation of plasticity using modulatory signals. The third part of this chapter describes the mechanism of excitability modulation starting with the introduction of a new excitatory factor that is embedded in the proposed neuron model. This is followed by the description of the mechanisms of up and down modulation of neural excitability using modulatory signals. The fourth part of this chapter describes the neuron's self-regulation or homeostatic processes for both plasticity and excitability modulation. The fifth and final part describes the types of synapses used by the artificial neurons to communicate with each other.

As already seen in the previous chapter, not only does heterosynaptic plasticity contribute to the homeostasis of neural activity in the nervous system, it also enhances its computational capabilities by extending its dynamics beyond the neuron-to-neuron or input specific activation which characterizes the most observed and studied form: Homosynaptic plasticity. The properties that emerge from the synergy between homosynaptic and heterosynaptic plasticity (for instance, modulation of synaptic change in hebbian-type plasticity, discrimination and prioritization of input stimuli) deserve further theoretical and experimental research [81]. The system presented in this chapter describes an artificial spiking neural network with heterosynaptic plasticity and heterosynaptic mediated neural excitability. This system is not aimed at creating an accurate model to reproduce the dynamics observed in biological systems. Instead, the proposed model aims at simulating some of the mechanisms involved in heterosynaptic plasticity in order to take advantage of the emerging dynamics and use them in the design and implementation of autonomous systems.

4.1 Introduction

This thesis describes a novel heterosynaptic modulation system in Spiking Neural Networks (SNN). The proposed modulation system is applied to two mechanisms of neural interaction, namely:

1. Hebbian based synaptic plasticity.

2. Neuronal excitability.

1. Synaptic Plasticity refers to the change in the strength or efficacy of a synapse as a result of neuronal activity in the presynaptic or the postsynaptic ends of the synapse. In hebbian-based plasticity the change in the synaptic strength is caused by the joint spiking activity between a presynaptic and a postsynaptic neuron. The process that regulates the amount of change in synaptic efficacy by means of an external input (*i.e.* a modulatory signal that is not originated from either end of the affected synapse) is known as modulation of synaptic plasticity. Not only does modulation of synaptic plasticity affect the amplitude

of change in synaptic efficacy, it may also elicit or inhibit plasticity in the synapse of the target neurons.

In the proposed hebbian-based synaptic plasticity system, each single neuron has an internal and independent plasticity modulatory factor which represents a hypothetical Plasticity Modulatory Substance (PMS) in the cell environment. The concentration or value of PMS at a given time determines the amplitude of the synaptic change as a result of the hebbian coactivation elicited by presynaptic and postsynaptic spike activity. The modulatory effect is subject to the affinity of the carrier synapse with the modulatory substance. For instance, if synaptic plasticity is triggered in a synapse with affinity to the PMS, then the amplitude of change of synaptic strength will be modulated by the current PMS concentration in the postsynaptic neuron. By contrast, if there is no affinity on the carrier synapse, the elicited synaptic plasticity is computed without the PMS modulatory factor.

The PMS in each individual neuron has its own equilibrium concentration or value. Therefore some neurons may have a high plasticity factor while other neurons may have a much lower or nil plasticity factor. The property that allows this system to be considered heterosynaptic is the presence of modulatory neurons that are able to increase or decrease the concentration of PMS inside individual neurons. This change in the internal behaviour of the target neurons caused by an external signal is referred to as 'modulation'.

The modulation from a modulatory neuron to a target neuron occurs through synapses where the modulatory neuron is presynaptic to its target. Thus, modulation occurs as a point-to-point mechanism. In order to induce changes in the behaviour of the target neurons, modulatory neurons send signals through synapses with the corresponding target neurons. The signal sent by the modulatory neuron is binary and is subject to the axonal (synaptic) delay of the carrier synapse. Given that the transmitted signal is binary, the information about the amplitude and polarity (increasing or decreasing) of the modulatory signal, and its effect on the target neuron, depends on the synaptic strength of the modulatory carrier synapse. The concentration of PMS inside each neuron is regulated by a homeostatic process that aims to maintain the PMS at its equilibrium value (which varies from neuron to neuron). Consequently the modulatory effect described above is transient since the target neuron will eventually revert its PMS to the equilibrium concentration.

2. Neuronal excitability refers to the way in which a neuron behaves in response to an incoming spike. In other words, the excitability describes the effect that an incoming presynaptic spike has on the postsynaptic membrane potential. Similar to the process of synaptic plasticity described above, in the proposed neuron excitability system, each single neuron has an internal and independent excitability modulatory factor which represents a hypothetical Excitability Modulatory Substance (EMS) in the cell environment.
The concentration or value of EMS at a given time determines the amplitude of the excitatory or inhibitory postsynaptic potential (*i.e.* the perturbation caused in the membrane potential by an incoming spike). However, the modulatory effect is subject to the affinity of the carrier synapse with the modulatory substance. Therefore, if an incoming spike arrives from a carrier synapse with affinity to the EMS, then the amplitude of the incoming spike will be modulated by the current EMS concentration in the postsynaptic neuron. By contrast, if there is no affinity on the carrier synapse, the elicited postsynaptic spike is computed without the EMS modulatory factor.
The EMS in each individual neuron follows the same dynamics described above with regard to PMS in the context of hebbian based synaptic plasticity. This includes the modulation of the EMS in target neurons by presynaptic modulatory neurons. Again, the signal sent by the modulatory neuron is binary and is subject to the axonal (synaptic) delay of the carrier synapse. Given that the transmitted signal is binary, the information about the amplitude and polarity (increasing or decreasing) of the modulatory signal, and its effect on the target neuron, depends on the synaptic strength of the modulatory carrier synapse. Similarly to the homeostatic regulation of PMS, the concentration of EMS inside each neuron is also regulated by a homeostatic process that aims to maintain the EMS at its equilibrium value, which varies from neuron to neuron. Consequently, the modulatory effect is again transient, as the target neuron will eventually revert its EMS to the equilibrium concentration.

The following Sections in this chapters describe in detail the underlying technical background of the heterosynaptic system for both plasticity and excitability modulation.

4.2 Plasticity Modulation

Plasticity Modulation (PM) is used as a mechanism to attenuate or amplify the amplitude of synaptic plasticity resulting from the application of the STDP rule which computes the synaptic change according to the relative timing between a presynaptic input current and a postsynaptic action potential (see Section 3.2). The PM in each Pulse Driver (PD) neuron [1] is determined by its internal (local) concentration of the Plasticity Modulatory Substance (PMS). The concentration and decay rate of the PMS may be different in each PD neuron allowing different neuro modulatory characteristics between neural sub circuits. The effect of the PMS on the plasticity of a given synapse depends on whether or not that synapse has affinity to the PMS.

4.2.1 A 3-factor STDP rule

The STDP model presented in chapter 3 section 3.2 is illustrated below as a 2 factor rule [9] where the change of synaptic efficacy (weight) W'_{ij} depends on: (1) the arrival time of the presynaptic spikes t_j^f and (2) the firing time at the postsynaptic neuron t_i^f:

$$W'_{ij} = f_{STDP}(t_i^f, t_j^f) \tag{4.1}$$

This 2 factor rule was implemented in section 3.2 as a formula to compute the change of synaptic efficacy resulting from the pairing of presynaptic and postsynaptic firing times. The implementation of the 2 factor rule is illustrated below as f_{STDP} function:

$$f_{STDP}(t_i^f, t_j^f) = \begin{cases} A_+ \exp(-(t_i^f - t_j^f)/\tau_+), \text{ if } (t_i^f - t_j^f) > 0 \\ -A_- \exp(\quad (t_i^f - t_j^f)/\tau_-), \text{ if } (t_i^f - t_j^f) < 0 \end{cases} \tag{4.2}$$

As this thesis proposes the use of plasticity modulation through a heterosynaptic architecture, the 2 factor form of the STDP rule must be extended in order to support the new modulating factor in the computation of the efficacy or weight change.

[1] In order to avoid ambiguity, in this chapter the (regular) spike firing neurons are referred to as PD neurons while modulatory neurons are referred to as NM neurons.

The extension of STDP as a 3 factor rule [9] is performed through the incorporation of a plasticity modulatory factor PMS (see figures 4.1(b), 4.2(b) and formula 4.3 of this chapter). The complete implementation of the 3 factor STDP formula proposed for the heterosynaptic architecture in this thesis is presented below:

$$f_{STDP}(t_i^f, t_j^f, PMS) = \begin{cases} PMS.A_+ \exp(-(t_i^f - t_j^f)/\tau_+), \text{ if } (t_i^f - t_j^f) > 0 \\ PMS. - A_- \exp(\quad (t_i^f - t_j^f)/\tau_-), \text{ if } (t_i^f - t_j^f) < 0 \end{cases} \tag{4.3}$$

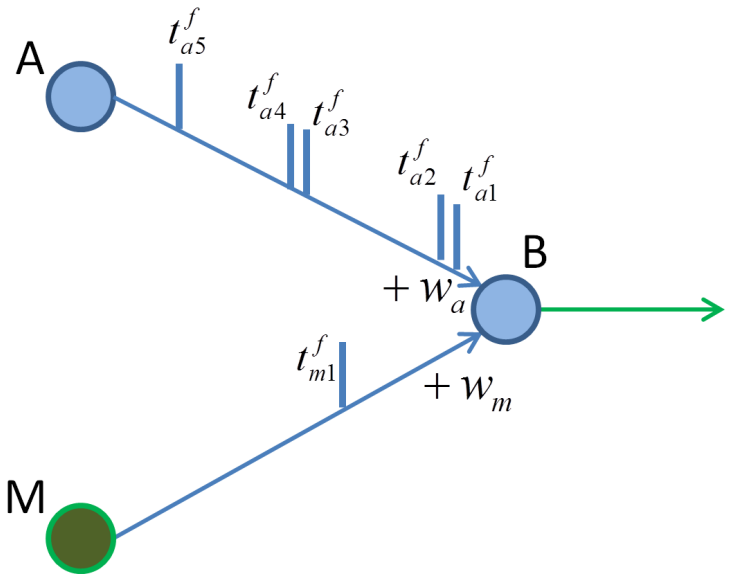
In formula 4.3 the extended function f_{STDP} receives 3 parameters: (1) the firing time of the postsynaptic neuron t_i^f; (2) the arriving time of the incoming presynaptic spike t_j^f and (3) the current concentration of the plasticity modulatory substance Pms. Compared to the original implementation of the STDP rule (formula 4.2) the only difference in the extended 3 factor implementation is the incorporation of PMS as a scaling factor for the exponential curve that describes the learning window (see figure 3.1). Thus, modulation of plasticity in the proposed artificial neural system occurs through the increase or decrease of the neuron embedded scaling factor PMS. Through plasticity modulation, the learning behavior of a PD neuron can be dynamically tuned through the firing activity of presynaptic Neuro-modulatory (NM) neurons. The mechanism of PMS modulation in a neuron by means of external modulatory signals is explained in the following sections of this chapter.

4.2.2 Up-modulation of PMS

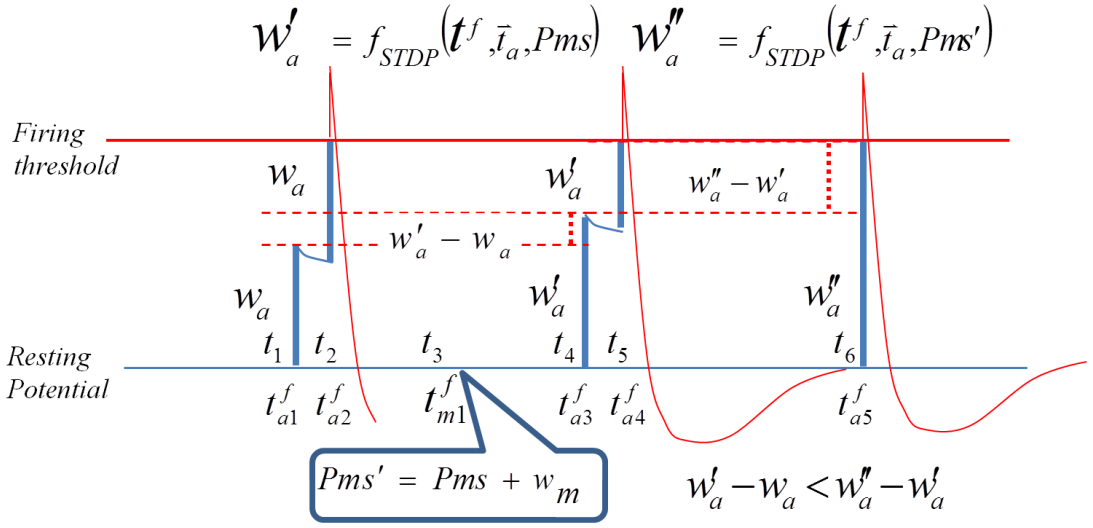
In Figure 4.1 (a) A spike train with 5 spikes is transmitted from presynaptic PD neuron A to postsynaptic PD neuron B. t_{a1}^f and t_{a5}^f represent the firing times of the first and the last spike respectively. $+W_a$ represents the synaptic efficacy between neurons A and B with the $+$ sign indicating that the synapse is excitatory.

Presynaptic PM neuron M transmits a modulatory signal at time t_{m1}^f. $+W_m$ represents the amplitude or efficacy of the signal with the $+$ sign indicating that the synapse is up-regulating the PMS at the postsynaptic neuron B.

Figure 4.1 (b) illustrates the behaviour of the membrane potential of postsynaptic neuron B during the arrival time of the presynaptic spike train sent by A. The first spike t_{a1}^{f} elicits an excitatory postsynaptic potential (EPSP) of amplitude W_a which corresponds to the actual efficacy of the carrier synapse.



(a)



(b)

Figure 4.1: Up-modulation of PMS in PD neuron B affecting its synaptic plasticity with presynaptic neuron A.

The EPSP slightly decays until the second spike at time t_{a2}^{f} arrives, producing once again an EPSP of amplitude W_a.

Through the second spike, the depolarization of the membrane potential reaches the firing threshold consequently triggering an action potential in B. The generation of the action potential activates the plasticity mechanism which involves the application of the STDP rule over the previous spikes that arrived at neuron B.

For the synapse between A and B the new efficacy W'_a is calculated by the 3-factor STDP function:

$$W'_a = f_{STDP}(t^f, \vec{t}_a, Pms) \tag{4.4}$$

The parameters of the formula are shown as follows: (1) the firing time in neuron B, t^f; (2) the arriving time of the previous incoming EPSPs (t^f_{a1} and t^f_{a2}) represented with $\vec{t}$; and (3) the current concentration of the plasticity modulatory substance PMS as Pms. The implementation of the 3-factor plasticity rule will be further explained in section 4.2.3 of this chapter.

The synaptic efficacy between A and B is updated with W'_a which has been potentiated with respect to the previous efficacy ($W'_a > W_a$). The signal transmitted at time T^f_{m1} by neuron M arrives at postsynaptic neuron B at time t_3. The concentration of PMS in B is increased according to the synaptic efficacy between M and B which is represented by W_m. As response to the incoming modulatory signal at time t_3 the increase of the PMS in neuron B is defined by:

$$PMS = PMS + W_m \tag{4.5}$$

PMS is the concentration of the plasticity modulatory substance in the target neuron (B) and is used as the modulatory factor in the implementation of the 3-factor STDP formula illustrated previously in 4.3. W_m represents the efficacy of the modulatory carrier synapse with the $+$ sign indicating that the weight is positive, thus increasing the concentration of PMS.

The third spike sent by A at time t^f_{a3} reaches the postsynaptic neuron B at time t_4 eliciting

an EPSP of amplitude W'_a. The difference between W'_a and W_a, *i.e.* $(W'_a - W_a)$, indicates the plasticity of the synapse that occurred during the triggering of the previous action potential. The EPSP slightly decays until the fourth spike sent by A at t^f_{a4} reaches B at time t_5, producing again an EPSP of amplitude W'_a which brings the membrane potential to reach the firing threshold, consequently triggering a new action potential in B .

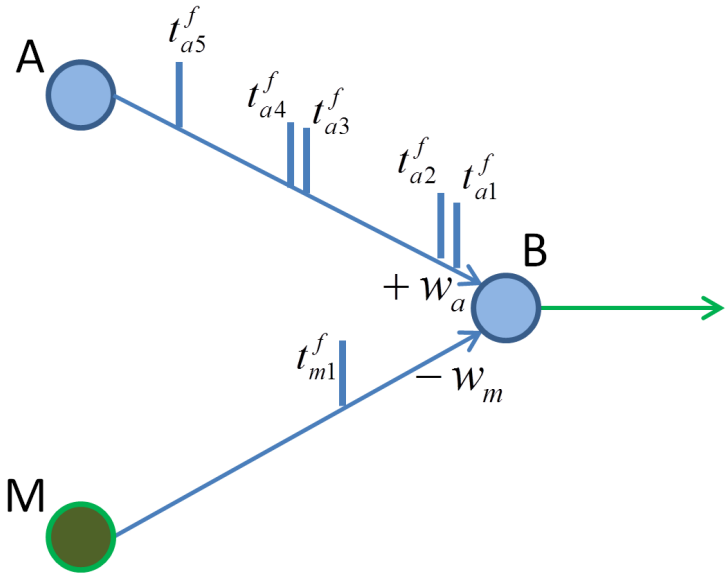
The STDP plasticity mechanism [2] is activated taking into account: (1) the new firing time, (2) the arriving time of the presynaptic spikes at times t_4 and t_5, (3) the current concentration of PMS which is now represented by PMS' [3] . The synaptic efficacy between A and B is updated with W''_a which again has been potentiated with respect to the previous efficacy $(W''_a > W'_a)$. As shown in Figure 4.1 (b) the amplitude of the EPSP at time t_6 (caused by the presynaptic spike sent by A at time t^f_{a5}) demonstrates that the change of efficacy from W'_a to W''_a is significantly higher than the change from W to W'_a, *i.e.* $(W'_a - W < W''_a - W'_a)$. The strong potentiation of W''_a through the up-regulation of plasticity in neuron B triggered by M, allowed the presynaptic neuron A to activate neuron B with a single spike.

[2] The membrane potential behaviour shown in Figure 4.1 (b) does not take into account the pre-after-post timing dynamics for the STDP rule in order to keep the illustration as simple as possible.

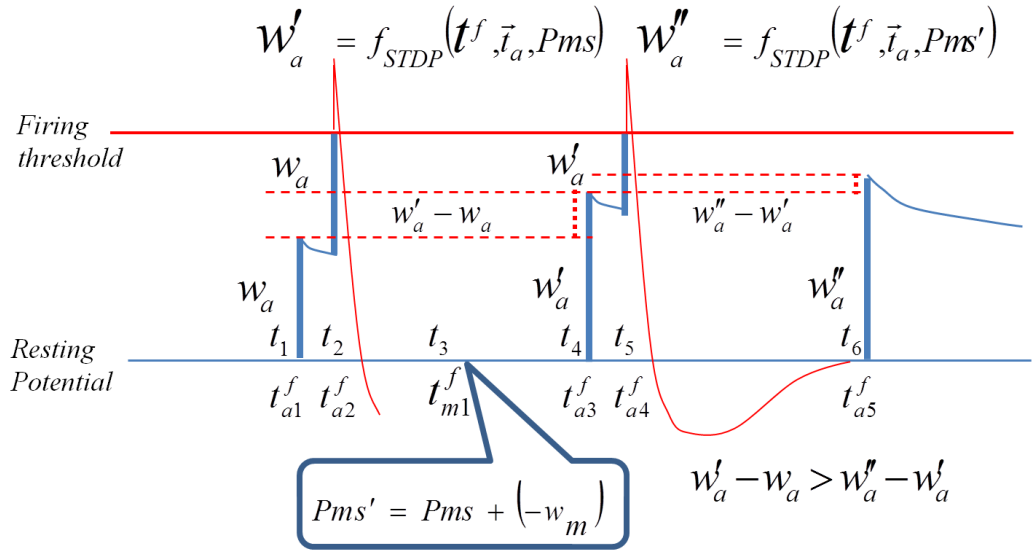
[3] Although PMS' describes the latest value after modulation activity, it does not take into account the process of homeostatic regulation which brings the modulatory substance towards its equilibrium concentration.

4.2.3 Down-modulation of PMS

In a similar way to Figure 4.1 (a), in Figure 4.2 (a) A spike train with five spikes is transmitted from presynaptic PD neuron A to postsynaptic PD neuron B. t_{a1}^f and t_{a5}^f represent the firing times of the first and the last spike respectively. $+W_a$ represents the synaptic efficacy between neurons A and B with the $+$ sign indicating that the synapse is excitatory.



(a)



(b)

Figure 4.2: Down-modulation of PMS in PD neuron B affecting its synaptic plasticity with presynaptic neuron A.

Presynaptic PM neuron M transmits a modulatory signal at time t^f_{m1}. $-W_m$ represents the amplitude or efficacy of the signal with the $-$ sign indicating that the synapse is down-regulating the PMS at the postsynaptic neuron B.

Figure 4.2 (b) illustrates the behaviour of the membrane potential of postsynaptic neuron B during the arrival time of the presynaptic spike train sent by A. The first spike t^f_{a1} elicits an excitatory postsynaptic potential (EPSP) of amplitude W_a which corresponds to the current efficacy of the carrier synapse. The EPSP slightly decays until the second spike fired at time t^f_{a2} arrives, producing once again an EPSP of amplitude W_a. Through the second spike, the depolarization of the membrane potential reaches the firing threshold consequently triggering an action potential in B. The generation of the action potential activates the plasticity mechanism which involves the application of the 3-factor STDP rule (see formula 4.3) over the previous spikes that reached neuron B.

For the synapse between A and B the new efficacy W'_a is calculated by the function f_{STDP} with the following parameters: (1) The neuron firing time t^f, (2) the arrival time of the previous incoming EPSPs (t^f_{a1} and t^f_{a2}) represented with $\vec{t}$. (3) the current concentration of the plasticity modulatory substance PMS. The synaptic efficacy between A and B is updated with W'_a which has been potentiated with respect to the previous efficacy $(W'_a > W_a)$. The signal transmitted at time t^f_{m1} by neuron M arrives at postsynaptic neuron B at time t_3. The concentration of PMS in B is decreased according to the synaptic efficacy between M and B which is represented by W_m:

$$PMS = PMS - W_m \tag{4.6}$$

W_m represents the efficacy of the modulatory carrier synapse with the $-$ sign indicating that the weight is negative, thus decreasing the concentration of PMS in the target neuron B. The third spike sent by A at time t^f_{a3} reaches the postsynaptic neuron B at time t_4 eliciting an EPSP of amplitude W'_a. The difference between W'_a and W_a, *i.e.* $(W'_a - W_a)$, indicates the plasticity of the synapse that occurred during the triggering of the previous action potential. The EPSP slightly decays until the fourth spike sent by A at time t^f_{a4} reaches B at time t_5, producing again an EPSP of amplitude W'_a which brings the membrane potential

to reach the firing threshold, consequently triggering a new action potential in B .

The STDP plasticity mechanism is activated taking into account: (1) the new firing time, (2) the arrival time of the presynaptic spikes at times t_4 and t_5, (3) the current concentration of PMS which is now represented by PMS' .

The synaptic efficacy between A and B is updated with W_a'' which again has been potentiated with respect to the previous efficacy $(W_a'' > W_a')$.

As shown in Figure 4.2 (b) the amplitude of the EPSP at time t_6 (caused by the presynaptic spike sent by A at time t_{a5}^f) demonstrates that the change of efficacy from W_a' to W_a'' is significantly lower than the change from W_a to W_a', *i.e.* $(W_a' - W_a > W_a'' - W_a')$. The lower potentiation of W_a'' through the down-modulation of plasticity in neuron B triggered by M, prevents the rapid growth of the synaptic efficacy from neuron A to B.

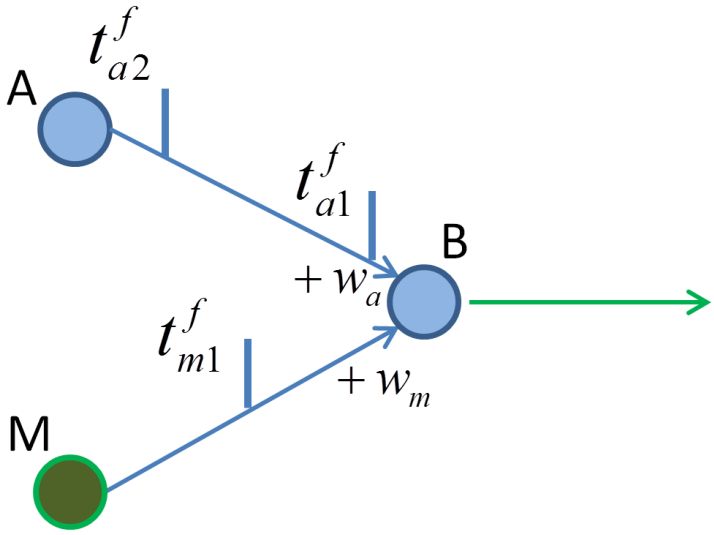
4.3 Excitability Modulation

Excitability Modulation (EM) is used as a mechanism to attenuate or amplify the amplitude of EPSPs and IPSPs originated from incoming presynaptic spikes. The EM in each PD neuron is determined by its internal (local) concentration of the EMS. Similarly to PMS, the decay rate of the EMS may be different on each PD neuron allowing different neuro modulatory characteristics between neural sub circuits. The up and down-modulation of the EMS in the postsynaptic neuron depends on the nature of the carrying synapse which can be either excitatory or inhibitory.

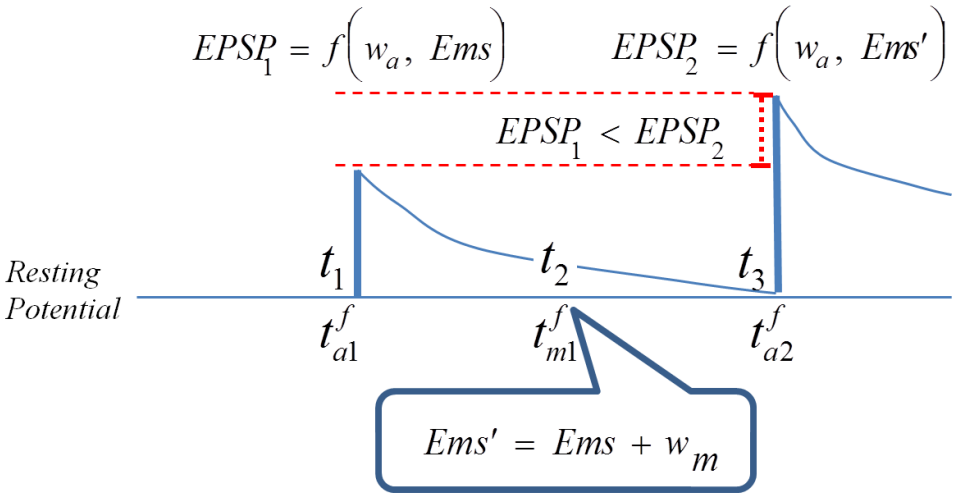
Through EM, the firing activity of a postsynaptic PD neuron can be dynamically tuned so that it can discriminate the incoming presynaptic inputs and react differently to each of them according to the affinity expressed by their carrying synapses.

Figures 4.3 and 4.4 illustrate the mechanisms of up and down-modulation of the EMS in a postsynaptic PD neuron (B) and its effect on the incoming synapse from a presynaptic neuron (A).

4.3.1 Up-modulation of EMS



(a)



(b)

Figure 4.3: Up-modulation of EMS in PD neuron B increasing the amplitude of EPSPs originated from presynaptic neuron A.

In Figure 4.3 (a) A spike train with two spikes is transmitted from presynaptic PD neuron A to postsynaptic PD neuron B. t^f_{a1} and t^f_{a2} represent the firing times of the first and the second spike respectively. $+W_a$ represents the synaptic efficacy between neurons A and B with the $+$ sign indicating that the synapse is excitatory.

Presynaptic EM neuron M transmits a modulatory signal at time t^f_{m1}. $+W_m$ represents the amplitude or efficacy of the signal with the $+$ sign indicating that the synapse is up-regulating the EMS at the postsynaptic neuron B. Figure 4.3 (b) illustrates the behaviour of the membrane potential of postsynaptic neuron B during the arrival time of the presynaptic spike train sent by A.

The first spike t^f_{a1} elicits an excitatory postsynaptic potential (EPSP) of amplitude $f(W_a, EMS)$ which takes into account two parameters: (1) the efficacy of the carrier synapse indicated by W_a. (2) The current concentration of EMS in the neuron.

The signal transmitted at time t^f_{m1} by neuron M arrives at postsynaptic neuron B at time t_2. The concentration of EMS in B is increased according to the synaptic efficacy between M and B which is represented by W_m. The new concentration of EMS is now represented by EMS'.

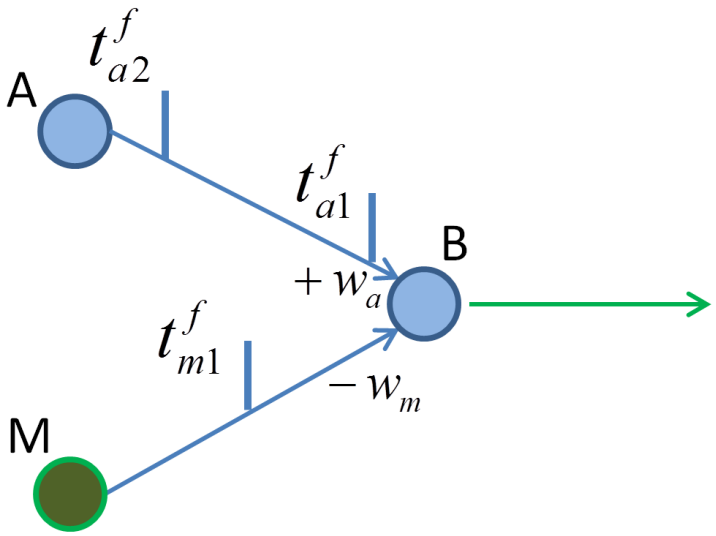
Shortly after the EPSP elicited at time t_1, the postsynaptic membrane potential decays almost reaching its resting potential until the second spike t^f_{a2} arrives at time t_3, eliciting once again an EPSP of amplitude $f(W_a, EMS')$. The difference between $EPSP1$ and $EPSP2$, where $(EPSP1 < EPSP2)$, indicates the increase in the synaptic efficacy after the modulatory signal received at time t^f_{m1}.

4.3.2 Down-modulation of EMS

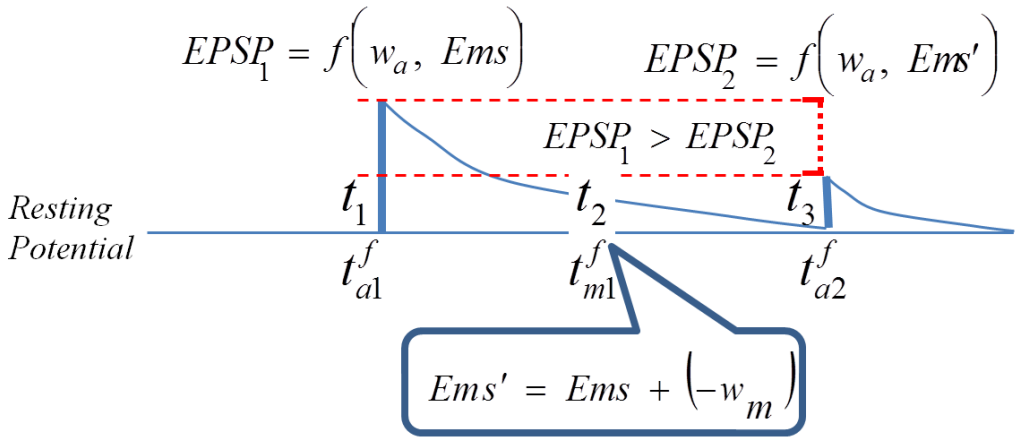
In Figure 4.4 (a) A spike train with two spikes is transmitted from presynaptic PD neuron A to postsynaptic PD neuron B. t^f_{a1} and t^f_{a2} represent the firing times of the first and the second spike respectively. $+W_a$ represents the synaptic efficacy between neurons A and B with the $+$ sign indicating that the synapse is excitatory.

Presynaptic EM neuron M transmits a modulatory signal at time t^f_{m1}. $-W_m$ represents the amplitude or efficacy of the signal with the $-$ sign indicating that the synapse is down-regulating the EMS at the postsynaptic neuron B.

Figure 4.4 (b) illustrates the behaviour of the membrane potential of postsynaptic neuron B during the arrival time of the presynaptic spike train sent by A.



(a)



(b)

Figure 4.4: Down-modulation of EMS in PD neuron B decreasing the amplitude of EPSPs originated from presynaptic neuron A.

The first spike t^f_{a1} elicits an excitatory postsynaptic potential (EPSP) of amplitude $f(W_a, EMS)$ which takes into account two parameters: (1) the efficacy of the carrier synapse indicated by W_a. (2) The actual concentration of EMS in the neuron. The signal transmitted at time t^f_{m1} by neuron M arrives at postsynaptic neuron B at time t_2. The concentration of EMS in B is decreased according to the synaptic efficacy between M and B which is represented by W_m.

The new concentration of EMS is now represented by EMS'. Shortly after the EPSP elicited at time t_1, the postsynaptic membrane potential decays almost reaching its resting potential until the second spike t_{a2}^f arrives at time t_3, eliciting once again an EPSP of amplitude $f(W_a, EMS')$. The difference between $EPSP1$ and $EPSP2$, where $(EPSP1 > EPSP2)$, indicates the decrease in the synaptic efficacy after the modulatory signal received at time t_{m1}^f.

4.4 PMS and EMS Homeostasis

In the proposed neural system the concentrations of the modulatory substances PMS and EMS have their corresponding equilibrium concentration (EC) values in each individual PD neuron. Thus, in the absence of incoming modulatory signals the equilibrium concentrations of PMS and EMS serve as default modulatory factors inside a PD neuron. Moreover, the role of the EC parameters is not just limited to serve as default modulatory factors but also to regulate the concentration of PMS and EMS inside each neuron. The artificial PD neuron aims to maintain both PMS and EMS at its corresponding equilibrium concentrations, thus resembling a biological homeostatic process where the concentration of a substance is up or down regulated in order to reach an stable equilibrium [82]. The homeostatic mechanism of both PMS and EMS is presented in the next sections of this chapter.

4.4.1 PMS Homeostasis

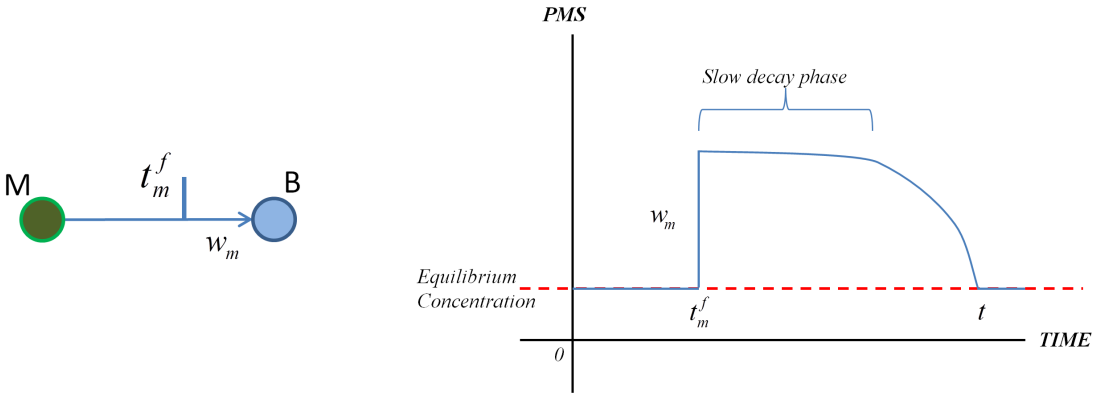
In a PD neuron, the mechanism of regulation of PMS works by verifying whether the concentration of PMS is different from its equilibrium concentration (EC). If the PMS concentration is higher or lower than the equilibrium value, the system proceeds to increase or decrease the PMS towards the EC value. This is implemented through the formula:

$$PMS = PMS + \Delta_{PMS} \tag{4.7}$$

Where Δ_{PMS} indicates the change of PMS towards EC. Δ_{PMS} is implemented as follow:

$$\Delta_{PMS} = \begin{cases} -A_{pms}\exp((t-t^f)/\tau_{pms}), \text{ if } PMS > EC & (a) \\ A_{pms}\exp((t-t^f)/\tau_{pms}), \text{ if } PMS < EC & (b) \\ 0, otherwise & (c) \end{cases} \tag{4.8}$$

In formula 4.8 above, A_{pms} and $-A_{pms}$ are the scaling increments of the PMS over time. t is the current time and t^f is the time of the last incoming plasticity-modulatory signal. τ_{pms} is a time constant which determines the shape of the exponential function that describes the evolution of PMS. Figure 4.5 illustrates with an example the behaviour of the PMS concentration and its mechanism of regulation or homeostasis:



(a) (b)

Figure 4.5: Behaviour of PMS in a PD neuron that receives a modulatory signal from an plasticity modulatory neuron.

In figure 4.5 (a) the presynaptic modulatory neuron M sends a modulatory signal of amplitude W_m that arrives at time t_m^f at the postsynaptic PD neuron B. (b) The behaviour of the PMS in neuron B shows that PMS is in its equilibrium concentration during the time interval $[\, t_0, t_m^f)$ until the incoming presynaptic signal arrives at t_m^f increasing the PMS concentration by an amplitude of W_m. As shown in (b) the increase of PMS is followed by a period of slow decay where the duration is determined by τ_{Pms}. The slow decay period serves to keep the modulatory effect at a relatively steady amplitude before entering in a

fast decay phase towards the EC. As shown above, the modulatory effect is transient, as the target neuron will eventually revert its PMS to the equilibrium concentration.

4.4.2 EMS Homeostasis

The homeostatic regulation of EMS occurs in a similar way to the mechanism shown before for PMS. There is also an equilibrium concentration (EC) which the PD neuron tries to maintain. Thus, if the EMS is higher or lower than the EC, the regulation mechanism will decrease or increase the EMS concentration towards its EC. This is implemented through the formula:

$$EMS = EMS + \Delta_{EMS} \tag{4.9}$$

Where Δ_{EMS} indicates the change of EMS towards EC. Δ_{EMS} is implemented as follow:

$$\Delta_{EMS} = \begin{cases} -A_{ems} \exp((t - t^f)/\tau_{ems}), \text{ if } EMS > EC & (a) \\ A_{ems} \exp((t - t^f)/\tau_{ems}), \text{ if } PMS < EC & (b) \\ 0, otherwise & (c) \end{cases} \tag{4.10}$$

In formula 4.10 above, A_{ems} and $-A_{ems}$ are the scaling increments of the EMS over time. t is the current time and t^f is the time of the last incoming excitability-modulatory signal. τ_{ems} is a time constant which determines the shape of the exponential function that describes the evolution of EMS. The behaviour of the EMS concentration curve is the same as the PMS curve shown in figure 4.5.

4.4.3 Synapses

In the proposed SNN system, the type of synapse determines the type of information that is transmitted between 2 neurons. A synapse can be a carrier for 2 types of information,

these can be either: (1) pulses or (2) modulatory signals.

Pulse carrier synapses

Pulse carrier synapses transmit pulses that elicit changes in the membrane potential of the receiving (postsynaptic) neuron. The postsynaptic potentials elicited by a pulse can be either excitatory (EPSP) or inhibitory (IPSP) depending of the efficacy of the synapse. Pulse carrier synapses may be sensitive (express affinity) to plasticity (PMS) and excitability (EMS) modulation. Synapses with affinity to PMS are highly dependent on the concentration of PMS in the postsynaptic neuron. In this case synaptic plasticty based on STDP evolves following the dynamics of a three factors rule [9] as described in formula 4.3 of chapter 4. Synapses without affinity to PMS do not depend on the concentrations of the PMS and their synaptic efficacy evolves following the traditional STDP two factor rule [9] described in formula 4.2 of chapter 4. In a similar way to PMS, synapses with affinity to EMS are highly dependent on the concentration of EMS in the postsynaptic neuron. Thus, incoming postsynaptic potentials are affected according to the concentration of EMS as described in section 4.3.

Modulatory synapses

Incoming synapses from modulatory neurons allow the increase or decrease of EMS and PMS concentrations in postsynaptic neurons. Incoming synapses from modulatory neurons may carry the signal to perform one of the following processes:

- Change of PMS in the postsynaptic neuron.
- Change of EMS in the postsynaptic neuron.

In the proposed system, an incoming synapse from a modulatory neuron will always carry one and the same type of signal. The amplitude of the increasing or decreasing effect of the signal depends on the efficacy of the incoming NM synapse in a similar way to an incoming synapse carrying a pulse.

4.5 Summary

This chapter has described the theory behind the proposed mechanisms of Heterosynaptic modulation of synaptic plasticity and neural excitability for SNN. In the first part of this chapter the dynamics of both plasticity and excitability modulation via a heterosynaptic architecture were described. The second part was devoted to the mechanism of plasticity modulation which started with the description of an extended 3-factor rule for hebbian learning. This was followed by the definition of the mechanisms of up and down modulation of plasticity through the use of modulatory signals. The next part of the chapter described the mechanism of excitability modulation which introduced a new built-in excitatory factor in the proposed neuron model. This was followed as well by the definition of their corresponding mechanisms of up and down modulation of neural excitability through modulatory signals. The fourth part of this chapter described the homeostatic processes in the neuron for both plasticity and excitability modulation mechanisms. The last part of this chapter defined the types of synapses used for the transmission of modulatory signals and regular pulses between neurons. The implementation of this theoretical system will be explored experimentally in the following chapters of this thesis.

Chapter 5

Implementation of the Proposed Spiking Neural System with Heterosynaptic Capabilities

This chapter suggests a set of data structures and algorithms for the implementation in software of the SNN system described in the preceding chapter. The architecture of the system is presented through simplified schematics and pseudocode that implement the behaviour of the components of the system.

5.1 Neurons

In order to allow a neural network to make use of heterosynaptic modulation, it requires the implementation of an architecture that includes both pulse-driver (regular) neurons and modulatory neurons. Therefore, in the proposed SNN model an artificial neuron will be implemented as either one of the following types:

- Pulse driver neuron.
- Neuro-Modulatory neuron.

5.1.1 Pulse driver neuron (PD)

A PD neuron is based on the classical dynamics of a threshold fire Spiking Neuron. The reaching of a threshold by the neuron's membrane potential produces a pulse (action potential) which is propagated to the postsynaptic neurons eliciting excitatory (EPSP) or inhibitory (IPSP) postsynaptic potentials, according to the efficacy of the synapses.

The implementation of the PD neuron proposed in this thesis is a simplification of Integrate-and-fire (I&F) models which recreate to some extent the phenomenological dynamics of neurons while abstracting the biophysical processes behind them. In the simplest terms, the implemented PD neuron assumes that the only inputs come from pulses of presynaptic neurons and there are no imposed external currents. The PD neuron also implements the embedded plasticity (PMS) and excitability (EMS) modulatory factors which are sensitive to external plasticity and excitability modulatory-signals respectively (as described in sections 4.2 and 4.3). As part of the modulatory mechanisms, the PD neuron implements as well the homeostatic processes (described in section 4.4) that regulate the concentrations of EMS and PMS.

5.1.2 Neuro-modulatory neuron (NM)

A NM neuron refers to an artificial neuron which transmits modulatory signals instead of electrical pulses. In terms of implementation, all the internal dynamics including the membrane potential, plasticity and homeostatic processes of a NM neuron are implemented

in the same way as for the PD neuron. However, the difference is in the outgoing signals that are transmitted as a result of an action potential or firing event. The type of transmitted information which can be either an electrical pulse or a modulating signal is indicated by the carrier synapse. Thus, a NM neuron is implemented as a PD neuron but with the difference imposed by the constraint that all its outgoing synapses are of modulatory type.

5.1.3 The underlying Threshold and Fire Model

In order to gain a better understanding of the differences between the implemented neuron model and the canonical I&F model, a comparison is shown below:

	Integrate and Fire Model (I&F)	Proposed
Membrane Potential	The canonical Integrate-and-Fire [19] represents the evolution of the neurone membrane potential through the time derivative of the Law of Capacitance: $I(t) = C_m \frac{dV_m(t)}{dt}$ 'Integrate' refers to the behaviour of the model when input currents are applied resulting in the increase of the membrane voltage until it reaches a set threshold which initiates a spike (fire event). The I & F model does not implement the decay of the membrane voltage towards its resting potential. Thus the membrane will keep a sub-threshold voltage indefinitely until new input currents make	The evolution of the membrane potential over time is described by the variable u. The behaviour of $u(t)$ depends on: (1) the machine state at time t, (2) the applied currents from incoming spikes and (3) the membrane potential leakiness (see below).

	the membrane cross the firing threshold.	
Leakiness	The decay or leakiness of the membrane potential is implemented as an extension of the I & F model: The Leaky-Integrate-and-Fire Model (LI & F) recreates the dynamics of a neuron by means of a current I flowing through the parallel connection of a resistor with a capacitor in an electrical circuit [26][20]. The current I splits in the resistor R and capacitor C, as follows: $I(t) = I_R + I_C = \frac{u(t)}{R} + C\frac{du}{dt}$ Where the voltage across the capacitor C is depicted with u and represents the neuron membrane potential. By introducing the membrane time constant $T_m = RC$, the above equation can be rewritten as: $T_m\frac{du}{dt} = -u(t) + RI(t)$ with T_m quantifying the rate at which u decays to its resting potential.	The decay of the membrane potential u is implemented as: $du = \frac{[v_{res}-u]}{\tau_m}$ if $v_{res} < u$ then $u = u - du$ else if $v_{res} > u$ then $u = u + du$ where v_{res} is the resting potential and τ_m is the membrane time constant.
Spike Initiation	The mechanism of spike initiation is established through a threshold	Same as I & F. Fixed firing threshold.

	condition: $u(t) = \theta$. Thus, if a given threshold θ is reached at $t = t^{(f)}$ then the neuron is said to fire a spike at time $t^{(f)}$.	
Action Potential	The form of the generated action potential is not described explicitly in the LI&F model [26]. Following the fire event, the potential is reset: $u_{reset} < \theta$. Then, when $t > t^{(f)}$ the dynamic behaviour continues as described by the membrane time constant T_m.	Same as I & F. During the generation of action potential the system initializes the absolute_refractory_period timer.
Refractoriness	The absolute refractory period is generally implemented by temporarily stopping the dynamics immediately after the threshold conditions have been reached. After the stop time the membrane potential dynamics start again with $u = u_{reset}$ where $u_{reset} < \theta$.	Same as I & F. The state of the system remains as absolute_refractory as long as the absolute_refractory_period timer is still alive.
Synapses	Following the framework of the I & F model, given a neuron i, its total input current is defined as the sum of all its incoming current pulses: $T_i(t) = \sum_j w_{ij} \sum_f \alpha(t - t_j^{(f)})$ Where $\alpha(t - t_j^{(f)})$ describes the time	Similarly to I & F, the total input current is also expressed as: $T_i(t) = \sum_j w_{ij} \sum_f \alpha(t - t_j^{(f)})$ However, in contrast with the I & F framework, in the proposed model

course from the presynaptic firing time $t(f)$ at neuron j and the arrival time t at postsynaptic neuron i. W_{ij} represents the synaptic weight or efficacy between neuron j and the postsynaptic neuron i. The postsynaptic current generated by an incoming spike depends on the elicited change in the conductance of the postsynaptic membrane [20].	the postsynaptic current only takes into account the efficacy W_{ij} of the synapses but not the conductance of the post-synaptic membrane.

From the comparison above it can be seen that similarly to the I&F model the neuronal system implemented in this thesis is also based on the threshold and fire framework, thus sharing similar mechanisms for the handling of the membrane potential, spike initiation, action potential and refractoriness. However, so far these characteristics can be considered part of the 'essential' neuronal dynamics in SNN. The extended modulatory capabilities presented in the previous chapter will be implemented on top of the underlying threshold and fire model. The following section of this chapter will start by describing the implementation of the 'essential' characteristics followed by a proposed process architecture with the corresponding algorithms for the implementation of the whole SNN system with extended modulatory capabilities.

5.1.4 A 2-state Clock-driven Neuron

The artificial neurons are simulated according to the clock driven SNN implementation by Jimenez *et al.* [83]. Following this approach, the time in the simulation is represented by an iteration-counter which increases at discrete steps or ticks. The value of the time step or tick determines the time granularity of the simulation and consequently the time dynamics of the neurons and their internal mechanisms.

For the simulation of the neuronal dynamics, a similar approach to Upegui *et al.* [84] has been adopted, modelling the artificial neuron as a finite-state machine where the states transitions depend mainly on a variable representing the membrane potential of the cell. This has the advantage of reducing the consumption of computational resources given that most of the internal processes of the artificial neuron are implemented in either one of the two possible operational states of the system: 'open' and 'absolute-refractory'. The processes or internal mechanisms that take place within the 2 states are: (1) membrane potential leakiness, (2) resting potential, (3) spike threshold, (4) excitatory and inhibitory postsynaptic response and (5) refractoriness.

In the open-state the neuron is receptive to input pulses coming from presynaptic neurons. The amplitude of postsynaptic potentials elicited by presynaptic pulses is given by the function $psp(w_{ij})$ (see Figure 5.1) where w_{ij} is the synaptic efficacy between presynaptic neuron j and postsynaptic neuron i . The membrane perturbations reported by $psp(w_{ij})$ are added (excitatory postsynaptic potential EPSP) or subtracted (inhibitory postsynaptic potential IPSP) to the actual value of the membrane potential u.

If the neuron firing threshold θ is not reached by u, then u begins to decay (see *decay*(u) in Figure 5.1) towards a fixed resting potential r_p. On the other hand, as it occurs in other Integrate-and-Fire implementations, if the membrane potential reaches a given threshold, an action potential or spiking process is initiated. In the presented model, when u reaches the firing threshold θ, this triggers a state transition from the open to the absolute-refractory state. During the latter, u is set to a fixed refractory potential value a_v (see Figure 5.2) and all incoming presynaptic pulses are neglected by u.

When the absolute-refractory state is initiated, an iteration counter i_c is set to a value n_r representing the number of iterations during which the absolutely-refractory state continues. Once the n_r iterations are completed, a state transition to the open-state is triggered by the condition $i_c = 0$.

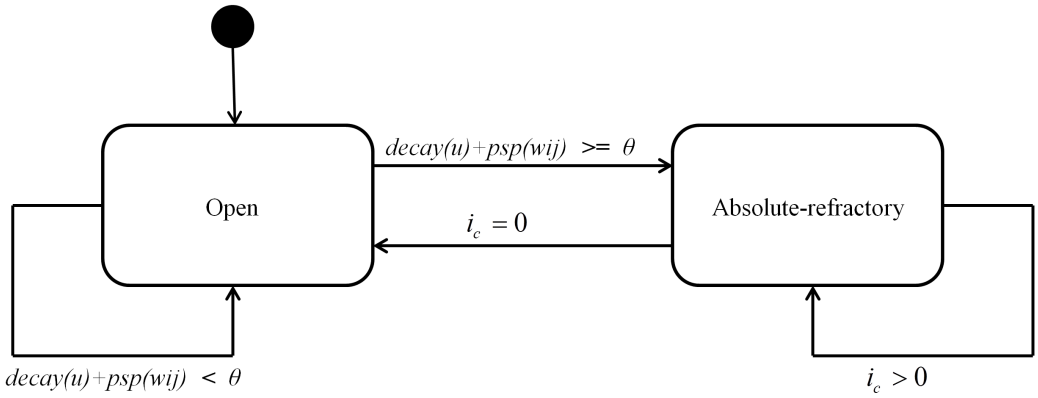


Figure 5.1: Model state transition represented with a Harel state chart.

As shown in Fig. 5.1 the membrane dynamics of the simulated neuron are encapsulated within the two states Open and Absolute-refractory, while the entire states transition depends on the two variables u and i_c corresponding to the membrane potential and refractory period counter respectively.

Figure 5.2 illustrates the behaviour of the membrane potential in response to incoming presynaptic spikes according to the simulation approach explained above:

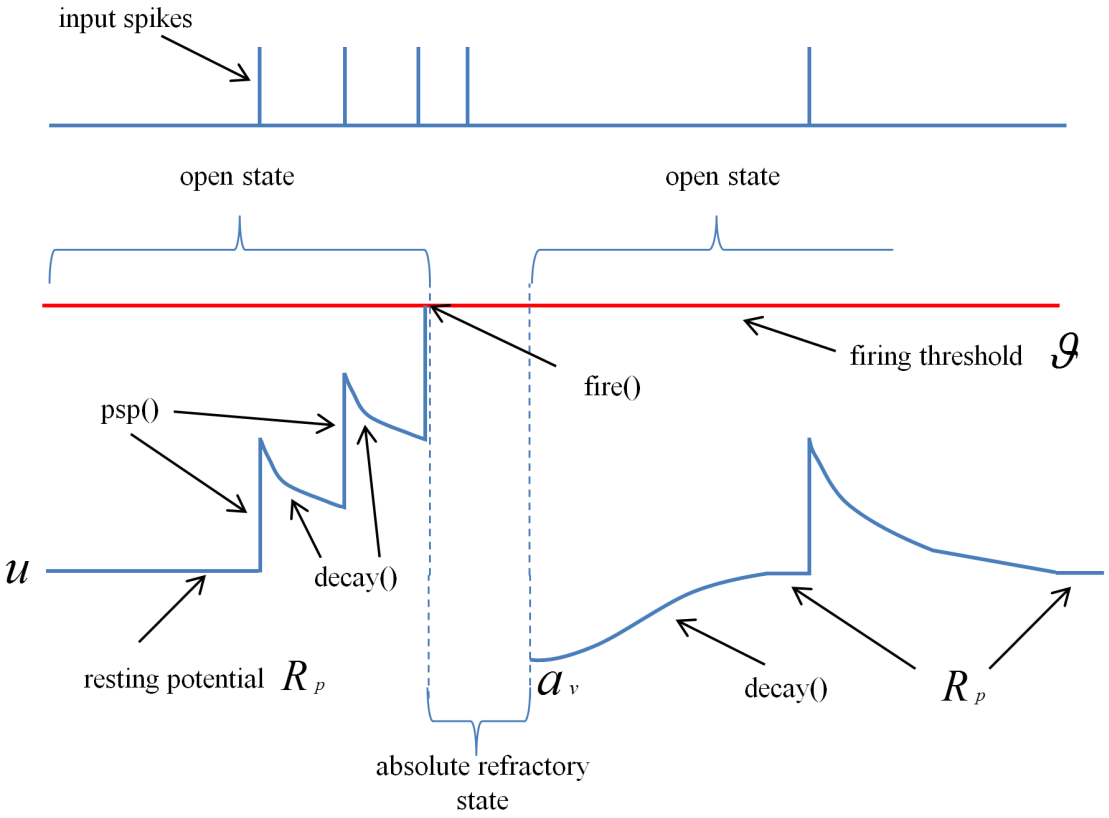


Figure 5.2: Modeling of the membrane potential in the implemented SNN model

5.1.5 Data Structures for Simulation, Neuron and Synapse Objects

So far it has been said that the threshold-and-fire dynamics of the artificial neuron presented in this thesis will be implemented based on 2 operational states (open and absolute refractory). The next step towards implementation involves the definition of the data structures which represent the main entities that build up a SNN system. These entities include: the simulation, neurons and synapses. This thesis adopted an Object-oriented programming (OOP) approach given that the use of classes provides an easier and more understandable way [85] to represent the above mentioned entities as objects that encapsulate their own attributes and behaviours.

The description of these entities as OOP classes is presented through UML class diagrams which are non specific to any particular programming language. Figure 5.3 below illustrates the relationship among the 3 main classes Simulation, Neuron and Synapse:

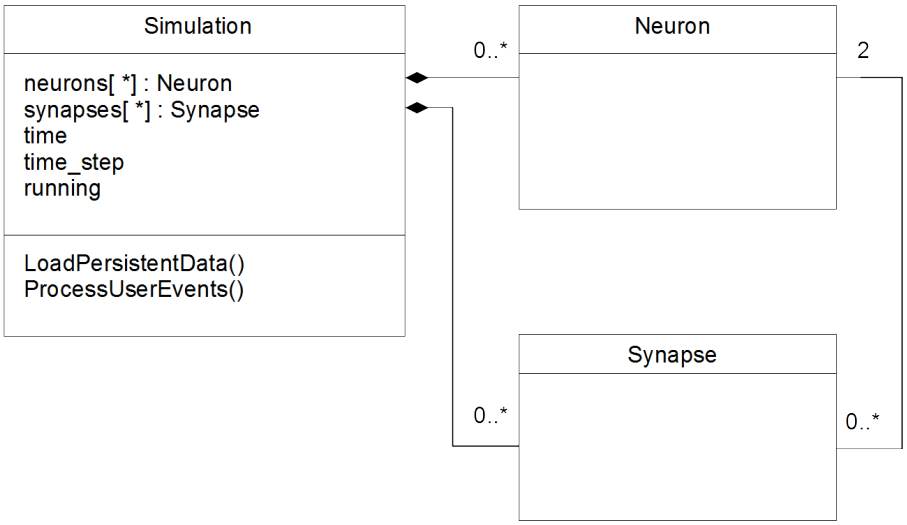


Figure 5.3: The 3 main entities: Simulation, Neuron and Synapse.

In figure 5.3, following the UML 2.x specification, the classes are depicted as boxes with 2 or 3 compartments depending on the level of detail at which the system is being described. The top compartment indicates the name of the class, the middle one shows the attributes, and the third one lists the class operations or behaviours.

The class 'Simulation' contains the parameters required to carry out the execution of the SNN simulation. These include a 'running' or execution flag, the elapsed simulation time represented by 'time' and the time step 'time_step' (or the time granularity) at which the clock driven simulation increases the 'time' attribute. As will be shown in the next sections of this chapter, the time and time_step attributes are used by the instantiated Neuron and Synapse objects as reference time for their internal dynamics.

Simulation also contains the attributes 'neurons' and 'synapses' where both attributes are followed by $[*]$ indicating multiplicity (a collection of elements) of unspecified length. A multiplicity may refer to a string, array, list or other collections of data, depending on the programming language and implementation approach. The multiplicity is followed by ':' and the name of a class or type which indicates the data type of the attribute. In this case the collections 'neurons' and 'synapses' are of type Neuron and Synapse respectively. Figure 5.3 illustrates the composition relationship between classes. The arrows pointing with a black diamond shape indicate that both Neuron and Synapse are 'part of' Simulation. The '0..*' at the other end of the arrows indicate the multiplicity of both attributes going from 0 to an unspecified number.

Figure 5.3 also shows a relationship between classes Neuron and Synapse. The type of this relationship is not explicit at this stage, however the numbers at each other end of the connecting line indicates that a Neuron may be associated with an unspecified number of Synapse objects while a Synapse may be associated with only 2 Neuron objects. For the sake of simplicity, the attributes and behaviours of Neuron and Synapse have been omitted in figure 5.3. A more detailed description of the composition and relationships of classes Neuron and Synapse is show next in figure 5.4.

The classes Neuron and Synapse are shown in figure 5.4 including all its attributes and operations. At the same time, from the class diagram it can be observed that some of the attributes are classes (or defined data structures) themselves.

In the class Neuron, the attributes shown in black correspond to the parameters of the fundamental threshold-and-fire SNN system described before. These attributes are: (1) 'state' which indicates the current machine state of the neuron, (2) 'membrane_potential' which was previously described in figures 5.1 and 5.2 as 'u', (3) 'resting_potential' which

indicates the membrane potential when the neuron is at rest (R_p in figure 5.2), (4) 'absolute_refractory_potential' which corresponds to the membrane potential of the neuron during its absolute refractory state, (5) the 'firing_threshold' which was described as 'θ' in figures 5.1 and 5.2, (6) the 'refractory_counter' which indicates the elapsed time of the absolute-refractory period (i_c in figure 5.2) and (7) 'last_firing_time' which stores the time of the neuron's last firing event.

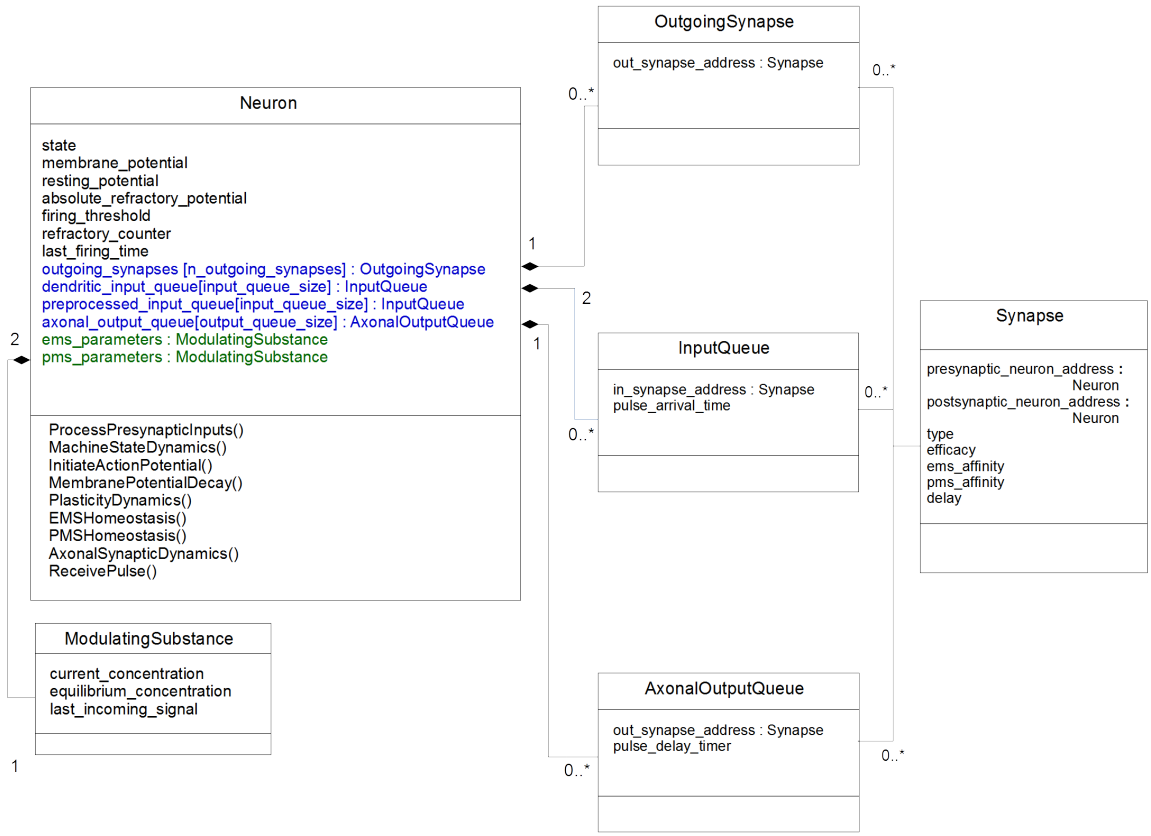


Figure 5.4: Neuron and Synapse data structures.

The next group of attributes in class Neuron is depicted in figure 5.4 with blue font. These attributes are: (1) 'outgoing_synapses' which is a collection of objects of the type 'OutgoingSynapse'. The later is used as a data structure to store the address of a 'Synapse' object. Thus 'outgoing_synapses' is a collection of data that contains the addresses of all the outgoing synapses in an artificial neuron. (2) 'dendritic_input_queue' is a collection of objects of the type 'InputQueue'. At the same time, 'InputQueue' is a data structure with 2 attributes: 'in_synapse_address' and 'pulse_arrival_time'. The first, serves to store the address of a 'Synapse' object. The second attribute stores time in the given time units used

by the implementation. Thus, 'dendritic_input_queue' is used to store the information from the presynaptic incoming pulses. This information includes the address of the pulse carrier synapse and its arrival time. (3) 'preprocessed_input_queue' is also a collection of objects of the type 'InputQueue'. However, it is used to store the incoming pulses where the plasticity rule has been applied. This will be further explained in the next sections of this chapter. (4) 'axonal_output_queue' is a collection of objects of the type 'AxonalOutputQueue'. The later is a data structure with 2 attributes: 'out_synapse_address' and 'pulse_delay_timer'. The first attribute stores the address of a 'Synapse' object while the second one stores a value representing time units. Thus, 'axonal_output_queue' is used to store the information (synapse address and scheduled time) of the outgoing pulses that are scheduled to be transmitted to postsynaptic neurons.

The third group of attributes depicted in green contain the data structures for the modulatory mechanisms of the extended neuron model. Both attributes 'ems_parameters' and 'pms_parameters' are of type 'ModulatingSubstance'. The later data structure contains the following parameters: (1) 'current_concentration' which represents the current concentration of the modulatory substance (modulatory factor), (2) 'equilibrium_concentration' indicates the homeostatic concentration value of the modulatory substance. (3) 'last_incoming_signal' stores the arrival time of the last incoming modulatory signal.

The class 'Synapse' is also shown above in figure 5.4. This class is used to represent the synapses in the SNN. The attributes: (1) 'presynaptic_neuron_address' and (2) 'postsynaptic_neuron_address' are both of data type 'Neuron'. These two attributes store the memory address of the presynaptic and postsynaptic neurons respectively. In other words a synapse knows the address of the neuron at each one of its ends.(3) The attribute 'type' indicates the nature of the carried information, whether it is a pulse, an efficacy modulating signal or a plasticity modulating signal. (4) 'efficacy' (or weight) represents the strength at which the presynaptic neuron can affect the postsynaptic (receiving) neuron. (5) Attributes 'ems_affinity' and (6) 'pms_affinity' indicate whether the respective mechanisms of excitability modulation and plasticity modulation are applicable to pulses carried by the synapse. (7) The attribute 'delay' represents the time that a pulse or modulatory signal

needs to travel through the synapse before reaching the postsynaptic neuron.

The relationships depicted in figure 5.4 between 'Synapse' and the three data structures 'OutgoingSynapse', 'InputQueue' and 'AxonalOutputQueue' indicates that a 'Synapse' object can be referenced in one 'OutgoingSynapse', multiple 'InputQueue' and multiple 'AxonalOutputQueue' elements. The way in which 'Neuron' and 'Synapse' objects interact with each other in order to behave as a network will be described in the next section.

5.1.6 Memory Organization and Interaction between Neuron and Synapse objects

As illustrated in figure 5.3, all the neuron entities in the simulation are contained in the data structure 'neurons' which is a collection of instantiated objects of type 'Neuron'. this collection can be either an array or list of objects where each individual object can be accessed through its corresponding memory address. Similarly, all the synapse entities in the simulation are contained in the collection 'synapses'. This, again, can be an array or list of instantiated 'Synapse' objects where each object can be accessed through its corresponding memory address.

As occur in biological as well as in artificial models, in this implementation 'Neuron' objects communicate with each other through synapses. As show in figure 5.5 each 'Neuron' object contains a list of outgoing synapses (as addresses of 'Synapse' objects). At the same time, each 'Synapse' object contains the address of a presynaptic and a postsynaptic neuron.

In figure 5.5 neuron '*N1*' depicted as a blue rectangle, has 3 outgoing synapses: *S21*, *S32* and *S38* represented with dashed lines leaving neuron '*N1*' from its right hand side. Inside '*N1*' the list 'outgoing_synapses' contains the addresses of the outgoing synapses. The address of synapse *S21* is shown enclosed within a green ellipse inside the "outgoing_synapses' list in '*N1*'. At the same time the synapse *S21* inside the 'synapses' list contains the addresses of its corresponding presynaptic and postsynaptic neurons (N1 and N2) shown at the top surrounded by a blue and red ellipse.

In this way, each neuron can access each one of its outgoing synapses whilst each outgoing synapse can access its corresponding neurons.

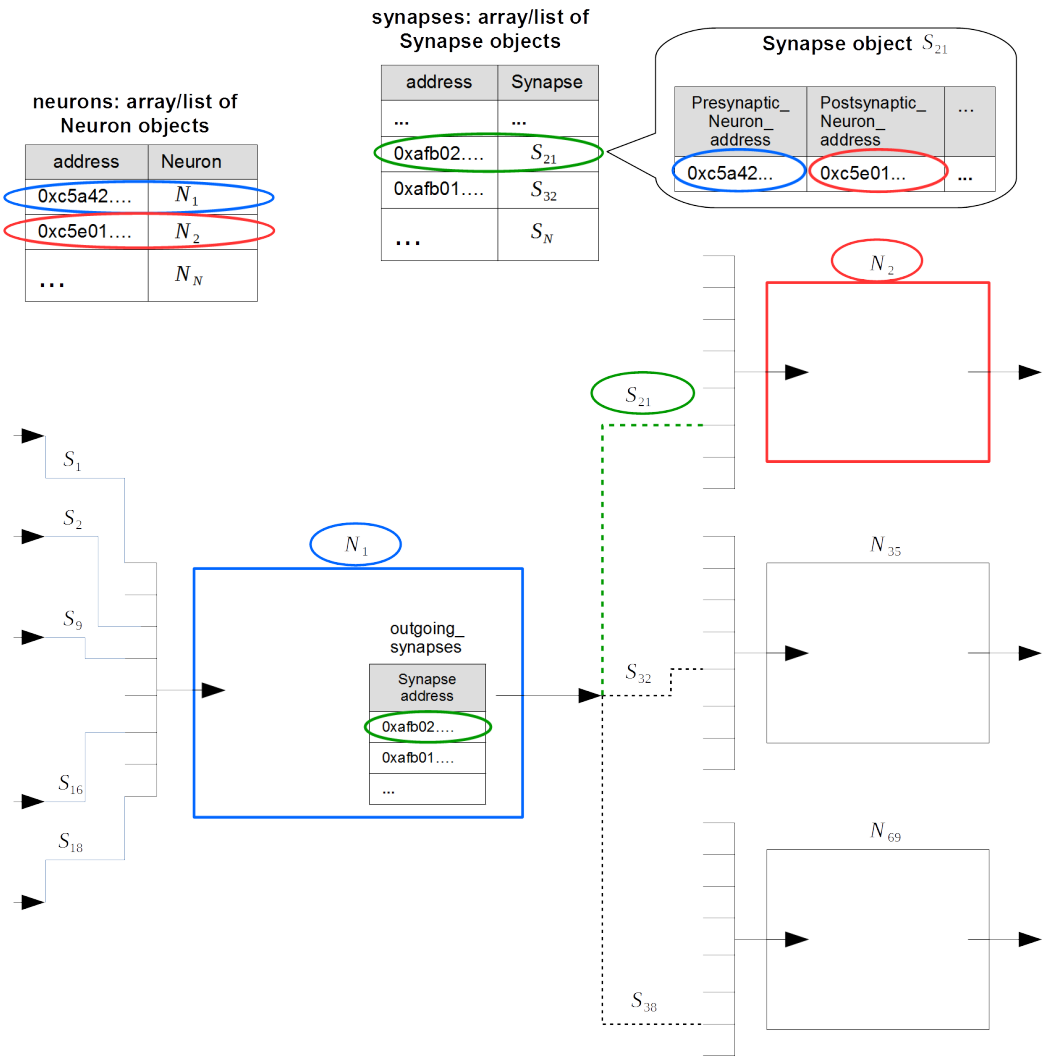


Figure 5.5: Memory organization in the implemented SNN.

5.1.7 The Neuron Components Architecture

Having the data structures, the next step is to implement the different dynamics of the artificial neuron, this includes the processing of presynaptic inputs including both pulses and modulatory signals, the evolution of the membrane potential, neuronal plasticity, firing events and the homeostatic processes described in section 4.4. This Characteristics of the neuron are implemented as different process components that interact within the neuron. Figure 5.6 shows these component and the way they communicate with each other:

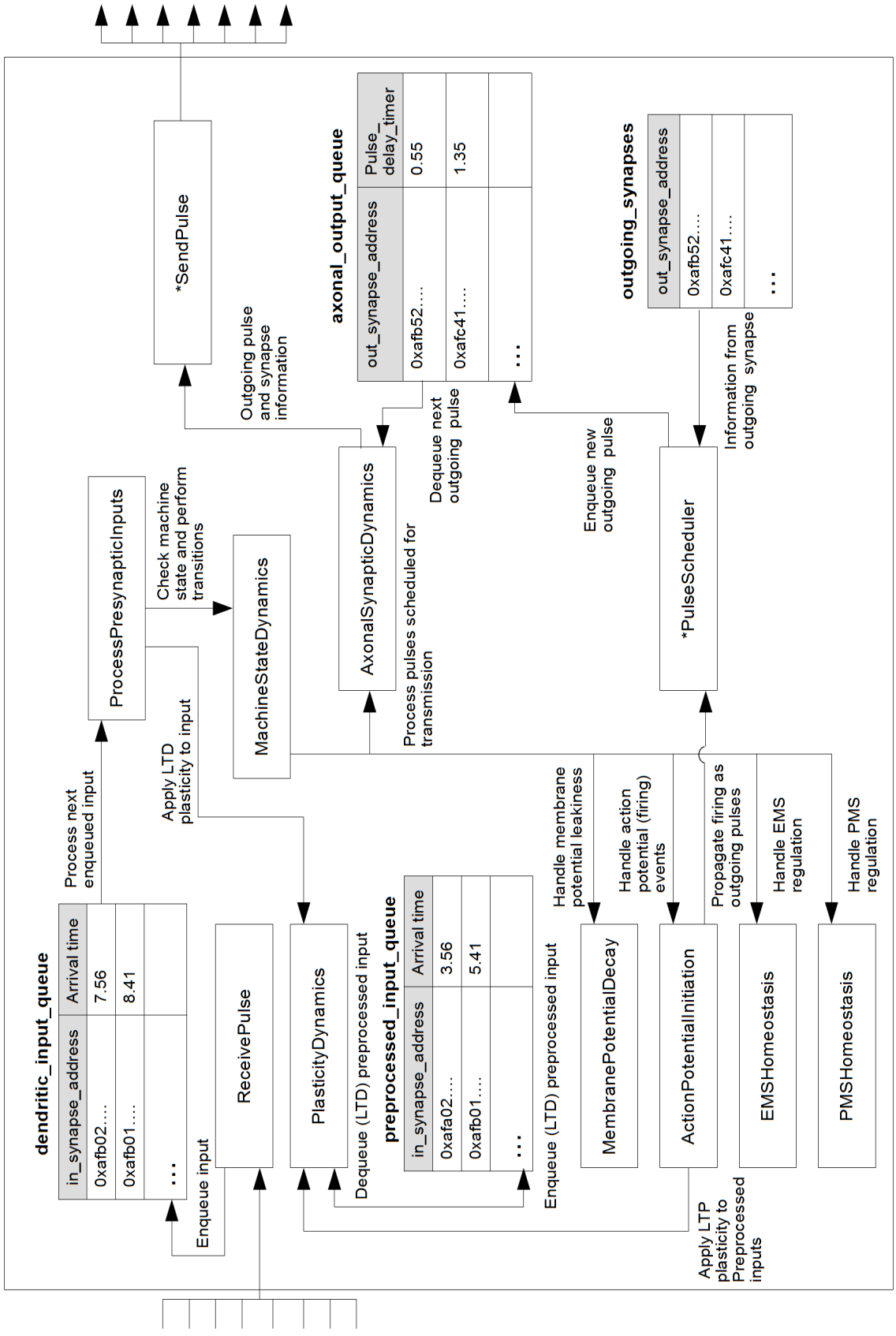


Figure 5.6: Internal Neuron components and their relationships.

Each one of the components shown in figure 5.6 is implemented as a function or methode encapsulated in the class Neuron. In the following sections the pseudocode implementation of each one of this components is described.

Run

The methode Run implemented below in figure 5.7 contains the main implementation of the Neuron object. From here all the different components are executed. The methode receives as parameters the current time (given in simulation iteration cycles) and the time increment 'time_step' at which the time of the simulation increases at each iteration. From here 5 process components are executed sequentially.

```
begin Neuron.Run(time, time_step)
      //… Call procedure that handles the inputs from
      //…  incoming synapses in 'Neuron'
      ProcessPresynapticInputs()
      //… Call procedure that handles the state machine dynamics
      //… in 'Neuron'
      MachineStateDynamics(time, time_step)
      //… Call procedure that handles the propagation of outgoing pulses
      //…  in 'Neuron'
      AxonalSynapticDynamics()
      //… Call procedure that handles EMS regulation in 'Neuron'
      EMSHomeostasis()
      //… Call procedure that handles PMS regulation in 'Neuron'
      PMSHomeostasis()
end
```

Figure 5.7: Pseudocode of Neuron component: Run

ProcessPresynapticInputs

The first component 'ProcessPresynapticInputs' implemented below in figure 5.8 is responsible for the processing of the incoming inputs from presynaptic neurons. The inputs are processed in their order of arrival until there are no more entries in the 'dendritic_input_queue'. Each input is processed according to the type of their carrier synapse indicated by 'synapse_type'. Inputs originated from PD neurons are processed as pulses. As mentioned before, if the neuron state is open, a pulse will elicit a change in 'membrane_potential'. Otherwise, if the neuron state is absolute-refractory any incoming pulse is discarded. On the other hand, inputs originated from NM neurons are processed as modulatory signals according to their type.

```
begin Neuron.ProcessPresynapticInputs(time)
     //… Iterate through 'dendritic_input_queue' processing each element as 'input
     foreach input in dendritic_input_queue do
          //… Use the address 'in_synapse_address' in 'input' to access the
          //… corresponding carrier synapse through the pointer 'p_current_synapse
          p_current_synapse ← input.in_synapse_address
          //… Store the synapse type (pulse driver or modulatory) in  synapse_type
          synapse_type ←  p_current_synapse.type
          //… if the carrier synapse is of pulse driver type then:
          if synapse_type = pulse_driver_type then
               //… Apply the STDP rule for long term depression (LTD) using the
               //… information in 'input'
               PlasticityDynamics(input, LTD)
               //… if the neuron machine state is open:
               if state = open then
                    //… Store 'input' in the list of inputs where the STDP rule
                    //… for LTD has been applied:
                    AppendTo(preprocessed_input_queue, input)
                    //… Get strength of the elicited postsynaptic potential (PSP)
                    psp ←  p_current_synapse.efficacy
                    //… If carrier synapse has affinity to EMS modulation
                    if p_current_synapse.ems_affinity = true then
                         //… Apply the EMS modulatory factor to 'psp'
                         psp ←  psp * EMS_parameters.current_concentration
                    endif
                    //… Perturbate the neuron's membrane potential with
                    //… the computed postsynaptic potential in 'psp'
                    membrane_potential ← membrane_potential + psp
                    //… Once processed, eliminate 'input' from dendritic_input_que
                    DeleteFrom(dendritic_input_queue, input)
               else
                    //… Neuron machine state is absolute_refractory. 'input' will
                    //… not elicit any change in neuron membrane potential and
                    //… should be ignored (eliminated) from  dendritic_input_queue
                    DeleteFrom(dendritic_input_queue, input)
               endif
          else
               //… The carrier synapse is of modulatory type:
               //… Store the synaptic efficacy (weight) in modulating_weight:
               modulating_weight ←  p_current_synapse.efficacy
               //… If synapse modulation type is of 'excitability modulating' type
               if synapse_type.subtype = excitability_modulating_type then
                    //… change the concentration of excitability modulating
                    //… substance (EMS) according to the modulating weight
                    EMS_parameters.current_concentration ←
                    EMS_parameters.current_concentration + modulating_weight
                    //… Store the time of the last incoming EMS modulatory signal
                    EMS_parameters.last_incoming_signal ← time
               else
                    //… If synapse type is of 'plasticity modulating' type then
                    //… change the concentration of plasticity modulating substanc
                    //… (PMS) according to the modulating weight
                    PMS_parameters.current_concentration ←
                    PMS_parameters.current_concentration + modulating_weight
                    //… Store the time of the last incoming PMS modulatory signal
                    PMS_parameters.last_incoming_signal ← time
               endif
               //… Eliminate 'input' from  dendritic_input_queue:
               DeleteFrom(dendritic_input_queue, input)
          endif
     endfor
end
```

Figure 5.8: Pseudocode of Neuron component: ProcessPresynapticInput

MachineStateDynamics

The next component in the sequence of execution is 'MachineStateDynamics' implemented below in figure 5.9 which is responsible for the handling and state transitions of the 2 machine states (open and absolute refractory) in the artificial neuron. In the 'open' state the procedure verifies if the neuron is prepared to fire, in which case the component procedure 'InitiateActionPotential' is called. Otherwise, if the membrane potential of the neuron has not reached its firing threshold then the leakiness or decay of the membrane is computed by the component procedure 'MembranePotentialDecay'.

```
begin Neuron.MachineStateDynamics(time, time_step)
     //… If the machine state of Neuron is open then
     if state = open then
          //… If the membrane potential of Neuron is not at resting then
          if membrane_potential != resting_potential then
               //… If the membrane potential of Neuron reached the firing
               //… threshold then:
               if membrane_potential >= firing_threshold then
                    //… Call procedure that handles the action potential
                    //…  process in Neuron:
                    InitiateActionPotential(time)
                    //… Set the machine state of Neuron to
                    //… absolute_refractory
                    state ← absolute_refractory
                    //… Initialize the refractory timer with the duration
                    //… of the refractory period
                    refractory_counter ← REFRACTORY_PERIOD_DURATION
               else
                    //… If membrane potential of Neuron is not at resting
                    //… and is not firing then call procedure that handles
                    //… the decay of the membrane potential in Neuron
                    MembranePotentialDecay()
               endif
          endif
     else
          //… If machine state of Neuron is not open then it is
          //… absolute_refractory.
          //…  Decrease the refractory counter(timer)  of Neuron
          //…  by 'time_step' units
          refractory_counter ← refractory_counter - time_step
          //… Keep the membrane potential of 'neuron' equal to its absolute
          //…  refractory potential
          membrane_potential ← absolute_refractory_potential
          //… If the Neuron refractory_counter reaches the end of the
          //…  refractory period then
          if refractory_counter <= 0 then
               //… Set machine state of Neuron to open
               state ← open
          endif
     endif
end
```

Figure 5.9: Pseudocode of Neuron component: MachineStateDynamics

InitiateActionPotential

The component 'InitiateActionPotential' implemented below in figure 5.10 starts by calling the component 'PlasticityDynamics' (see figure 5.11 for implementation) which applies the STDP plasticity rule to the presynaptic input pulses that arrived at the neuron before the triggering of the firing event. After that, the method proceeds with the scheduling of outgoing pulses (or modulating signals) in the 'axonal_output_queue'. The scheduling of outputs is done for each outgoing synapse stored in the list 'outgoing_synapses'. Each scheduled output is added to the 'axonal_output_queue' providing information about their corresponding output synapse and the transmission (synaptic) delay (given as time units).

```
begin Neuron.InitiateActionPotential(time)
      //… Store the current time as last firing time of Neuron
      last_firing_time ← time
      //… Apply the STDP rule for long term potentiation (LTP) to the
      //… preceding incoming spikes stored in 'preprocessed_input_queue'.
      //… NULL is given as first parameter since function is not
      //… processing a single input (LTD case) but the whole queue instead
      PlasticityDynamics(NULL, LTP)
      //… Empty 'preprocessed_input_queue' Since all its entries have been
      //… processed by the LTP plasticity function
      Empty(preprocessed_input_queue)
      //… for each outgoing synapse schedule a new pulse to be sent:
      //… Iterate through 'outgoing_synapses' processing each element
      //… as 'out_synapse'
      foreach out_synapse in outgoing_synapses do
            //… Use the address 'out_synapse_address' in 'out_synapse' to access
            //… the carrier synapse through pointer variable 'p_current_synapse'
            p_current_synapse ← out_synapse.out_synapse_address
            //… Instantiate a new outgoing pulse to be added (scheduled)
            //… in the 'AxonalOutputQueue'
            AxonalOutputQueue outgoing_pulse
            //… Provide information about the carrier synapse and synaptic delay
            //… to the new pulse
            outgoing_pulse.out_synapse_address ← p_current_synapse
            outgoing_pulse.pulse_delay_timer ← current_synapse.delay
            //… Add 'outgoing_pulse' to the queue of scheduled outgoing pulses
            //… named 'axonal_output_queue'
            AppendTo(axonal_output_queue, outgoing_pulse)
      endfor
end
```

Figure 5.10: Pseudocode of Neuron component: ActionPotentialInitiation

PlasticityDynamics

The component 'PlasticityDynamics' implemented below in figure 5.11 is responsible for applying the STDP rule (see section 3.2 and 4.2.1 for reference) to the incoming pulses sent by presynaptic neurons.

The provided parameter 'stdp_window_interval' indicates whether STDP will be applied to an input with pre-after-post timing (LTD window interval) or to inputs with pre-before-post timing (LTP window interval) as described previously in section 3.2 and illustrated in figure 3.1.

```
begin Neuron.PlasticityDynamics(input, stdp_window_interval)
     //… If timing of 'input' is on the LTD interval of the STDP window then
     if stdp_window_interval = LTD then
          //… get the elapsed time between the neuron's last firing event
          //… and the arrival time of the incoming spike stored in 'input'
          delta_time ← last_firing_time - input.pulse_arrival_time
          //… If the elapsed time is in a valid range for processing then
          if LTD_MIN_RANGE < delta_time and delta_time <= LTD_MAX_RANGE then
               //…Use the address 'in_synapse_address' in 'input' to access the
               //… carrier synapse through pointer variable 'p_current_synapse'
               p_current_synapse ← input.in_synapse_address
               //… If the carrier synapse is excitatory then
               if p_current_synapse.efficacy >= 0 then
                    //… Compute STDP rule according to 2-factor formula 4.2 LTD
                    //… store the efficacy change into 'nw'
                    nw ← -a_plst * exp((last_firing_time -
                                        input.pulse_arrival_time) / Tau_pms_neg)
                    //… If carrier synapse has affinity to PMS modulation:
                    if p_current_synapse.pms_affinity = true then
                         //… Apply the modulatory factor to the new efficacy
                         //… according to formula 4.3
                         nw ← nw * PMS_parameters.current_concentration
                    endif
                    //… Update the synaptic efficacy of the carrier synapse with
                    //… the computed efficacy change in 'nw'
                    p_current_synapse.efficacy ← p_current_synapse.efficacy + nw
               endif
          endif
     else
          //… If timing of 'input' is on the LTP interval of the STDP window then
          //… Iterate through 'preprocessed_input_queue' processing each element
          //… as 'prep_input'
          foreach prep_input in preprocessed_input_queue do
               //… get the elapsed time between the neuron's last firing event
               //… and the arrival time of the incoming spike in 'prep_input'
               delta_time = last_firing_time - prep_input.pulse_arrival_time
               //… If the elapsed time is still in a valid range for processing the
               if LTP_MIN_RANGE < delta_time and delta_time <= LTP_MAX_RANGE then
                    //… Use address 'in_synapse_address' in 'prep_input' to access
                    //… carrier synapse through pointer variable 'p_current_synapse
                    p_current_synapse ← prep_input.in_synapse_address
                    //… If the carrier synapse is excitatory then
                    if p_current_synapse.efficacy >= 0 then
                         //… Compute STDP rule according to 2-factor formula 4.2 LT
                         //… store the efficacy change into 'nw'
                         nw ← a_plst * exp(- (last_firing_time -
                                             input.pulse_arrival_time) / Tau_pms_pos)
                         //… If carrier synapse has affinity to PMS modulation
                         if p_current_synapse.pms_affinity = true then
                              //… Apply the modulatory factor to the calculated
                              //… efficacy change 'nw' according to formula 4.3
                              nw ← nw * PMS_parameters.current_concentration
                         endif
                         //… Update the synaptic efficacy of the carrier synapse w
                         //… the computed efficacy change in 'nw'
                         p_current_synapse.efficacy ← p_current_synapse.efficacy + nw
                    endif
               endif
          endfor
     endif
end
```

Figure 5.11: Pseudocode of Neuron component: PlasticityDynamics

MembranePotentialDecay

The component 'MembranePotentialDecay' implemented below in figure 5.12 handles the leakiness of the membrane potential towards its resting potential value. When the current value of the membrane potential represented by 'membrane_potential' is higher or lower than the value indicated by 'resting_potential' then the change towards the neuron resting potential is computed in 'delta_membrane' and added or subtracted to the current membrane potential value in 'membrane_potential'.

```
begin Neuron.MembranePotentialDecay(time_step)
      //… Calculate the leakiness of the neuron membrane potential
      //… according to the difference between the resting potential
      //… and the current membrane potential
      delta_membrane ← (abs(resting_potential - membrane_potential) /
      membrane_time_constant) *  time_step
      //… If membrane potential is higher than the resting potential
      if membrane_potential > resting_potential then
           //… Decrease membrane_potential by delta_membrane
           membrane_potential ←  membrane_potential -  delta_membrane
      else if membrane_potential < resting_potential then
           //… Else if membrane potential is lower than resting potential
           //… Increase membrane_potential by delta_membrane
           membrane_potential ←  membrane_potential -  delta_membrane
      endif
end
```

Figure 5.12: Pseudocode of Neuron component: MembranePotentialDecay

AxonalSynapticDynamics

The component 'AxonalSynapticDynamics' implemented below in figure 5.13 is responsible for the transmission of the pulses (or modulating signals) scheduled for transmission in 'axonal_output_queue'. For each scheduled output, this procedure decreases the delay timer which was previously initialized by the component 'InitiateActionPotential' according to the delay of the output carrier synapse. When the timer of the scheduled output reaches its end, the component AxonalSynapticDynamics calls the receiving postsynaptic neuron through its provided component interface 'ReceivePulse'.

```
begin Neuron.AxonalSynapticDynamics(time, time_step)
     //… Iterate through 'axonal_output_queue' processing each element
     //… as 'outgoing_pulse'
     foreach outgoing_pulse in axonal_output_queue do
          //… If it is time to send the pulse indicated by 'outgoing_pulse' then
          if outgoing_pulse.pulse_delay_timer <= 0 then
               //… Use the address 'out_synapse_address' in 'outgoing_pulse' to
               //… access the carrier synapse through pointer 'p_current_synapse'
               p_current_synapse ← outgoing_pulse.out_synapse_address
               //… Use the address 'postsynaptic_neuron_address' in
               //… 'p_current_synapse' to access the postsynaptic (receiving)
               //… neuron through the pointer 'p_postsynaptic_neuron'
               p_postsynaptic_neuron ← p_current_synapse.postsynaptic_neuron_address
               //… Deliver pulse to the postsynaptic target neuron providing
               //… the address of the carrier synapse and pulse arrival time
               p_postsynaptic_neuron.ReceivePulse(p_current_synapse, time)
               //… Once the pulse has been sent, eliminate 'outgoing_pulse' from
               //… 'axonal_output_queue'
               DeleteFrom(axonal_output_queue, outgoing_pulse)
          else
               //… If it is not the time to send the pulse indicated by
               //… 'outgoing_pulse' then continue decreasing the pulse delay timer
               outgoing_pulse.pulse_delay_timer ← outgoing_pulse.pulse_delay_timer
                                                  -  time_step
          endif
     endfor
end
```

Figure 5.13: Pseudocode of Neuron component: AxonalSynapticDynamics

ReceivePulse

The component 'ReceivePulse' implemented below in figure 5.14 is a methode that the neuron provides as an interface to receive incoming pulses and modulating signals from presynaptic neurons. The information of the incoming input (synapse address and input time) is passed as parameters by the presynaptic (sending) neuron to the component. The received input is then stored in 'dendritic_input_queue' for further processing by the receiving neuron.

```
begin Neuron.ReceivePulse(input_synapse_address, pulse_arrival_time)
     //… Instantiate a new input pulse to be added in the
     //… dendritic_input_queue of the receiving neuron
     DendriticInputQueue queue_input
     //… Provide information about the carrier synapse and pulse arrival time
     queue_input.in_synapse_address ←  input_synapse_address
     queue_input.pulse_arrival_time ←  pulse_arrival_time
     //… Add 'queue_input' to the queue of input pulses named
     //… 'dendritic_input_queue' in the receiving neuron
     AppendTo(dendritic_input_queue, queue_input)
end
```

Figure 5.14: Pseudocode of Neuron component: ReceivePulse

EMSHomeostasis

The component 'EMSHomeostasis' implemented below in figure 5.15 handles the regulation of the excitabiliy modulating substance (EMS) according to the dynamics described in section 4.3. When the current concentration of EMS represented by 'ems_parameters.current_concentration' is higher or lower than the EMS equilibrium concentration in 'ems_parameters.equilibrium_concentration' then the change towards the equilibrium concentration is computed in 'delta_ems' and added or substracted to the current EMS value.

```
begin Neuron.EMSHomeostasis(time)
      //… If the current concentration of the neuron excitability
      //… modulatory substance (EMS) is different to the excitability
      //… equilibrium concentration
      if ems_parameters.current_concentration !=
      ems_parameters.equilibrium_concentration then
            //… Calculate the change of the EMS towards its equilibrium
            //… concentration according to formula 4.10
            delta_ems ← (time - ems_parameters.last_incoming_signal) /
            ems_time_constant *  ems_parameters.a_mod
            //… If EMS is lower than its equilibrium concentration
            if ems_parameters.current_concentration <
            ems_parameters.equilibrium_concentration then
                  //… Increase EMS by delta_ems
                  ems_parameters.current_concentration ←
                  ems_parameters.current_concentration +  delta_ems
            else
                 //… Else if EMS is higher than its equilibrium concentration
                 //… Decrease EMS by delta_ems
                 ems_parameters.current_concentration ←
                 ems_parameters.current_concentration -  delta_ems
            endif
      endif
end
```

Figure 5.15: Pseudocode of Neuron component: EMSHomeostasis

PMSHomeostasis

The component 'PMSHomeostasis' implemented below in figure 5.16 handles the regulation of the plasticity modulating substance (PMS) according to the dynamics described in section 4.4. When the current concentration of PMS represented by 'pms_parameters.current_concentration' is higher or lower than the PMS equilibrium concentration in 'pms_parameters.equilibrium_concentration' then the change towards the equilibrium concentration is computed in 'delta_pms' and added or substracted to the current PMS value.

```
begin Neuron.PMSHomeostasis(time)
      //… If the current concentration of the neuron plasticity
      //… modulatory substance (PMS) is different to plasticity
      //… equilibrium concentration
      if pms_parameters.current_concentration !=
      pms_parameters.equilibrium_concentration then
            //… Calculate the change of the PMS towards its equilibrium
            //… concentration according to formula 4.08
            delta_pms ← (time – pms_parameters.last_incoming_signal) /
            pms_time_constant *  pms_parameters.a_mod
            //… If PMS is lower than its equilibrium concentration
            if pms_parameters.current_concentration <
            pms_parameters.equilibrium_concentration then
                  //… Increase PMS by delta_ems
                  pms_parameters.current_concentration ←
                  pms_parameters.current_concentration +  delta_pms
            else
                 //… Else if PMS is higher than its equilibrium concentration
                 //… Decrease PMS by delta_pms
                 pms_parameters.current_concentration ←
                 pms_parameters.current_concentration -  delta_pms
            endif
      endif
end
```

Figure 5.16: Pseudocode of Neuron component: PMSHomeostasis

5.1.8 Summary

This chapter suggested a software implementation approach for the heterosynaptic SNN system proposed in this thesis. A set of data structures to represent the different entities and components of the system were described and illustrated through UML diagrams. The architecture of the system was illustrated through components schematics with the corresponding pseudo-code implementation of each system component.

Chapter 6

Traditional non Modulated Neural Dynamics vs Plasticity and Efficacy Modulated Dynamics

This chapter explores the theoretical and practical applications of the propossed system by comparing two neural circuits: traditional non modulated neural dynamics vs Plasticity and Efficacy modulated dynamics. The experimental setup of both neural circuits applied to a virtual insect brain is described and the results are summarised.

6.1 A Bio-inspired experimental setup

Despite their lower neural complexity compared to vertebrates, the capabilities shown by insects to interact and cope with the environment are being considered as key features for implementation in the design of autonomous systems [42]. These capabilities include: exploration, reliable navigation, pattern recognition and interactions with each other. Based on the fact that SNN are able to reproduce to some extent the computational characteristics of biological neural systems, they are a potential computational instrument to achieve the above mentioned capabilities in artificial systems. There is increasing research (e.g., [41], [42], [86], [44]) on the use of SNN to control autonomous systems which exhibit insect-like intelligent behaviour in terms of learning and adaptation to the environment.
Given the aforementioned, this chapter will explore the application of the proposed heterosynaptic system as an artificial neural controller for an autonomous 'insect-like' agent moving in a simulated environment. This will be performed with the purpose to test the research hypothesis (see section 1.3) specifically the first statement which states that the conventional homosynaptic architecture can be extended to heterosynaptic models that are able to regulate their electrical spiking dynamics.
Section 3.3 described some of the existing underlying architectures for the design of artificial neural circuits using associative-learning and classical conditioning as a learning mechanism in autonomous systems. This chapter will use these architectures to setup an experiment where two neural circuits are implemented and compared as follows:
A simplified insect-like neural controller will be design based on the associative topologies described in chapter 3 and illustrated in Figure 3.4 . This neural controller (virtual brain) will be implemented in two different ways in order to test and compare the following two experimental conditions:

1. Using a traditional threshold fire SNN model.

2. Using the hereby proposed mechanism of heterosynaptic-modulation of plasticity and excitability.

In both cases the simulated insect should be able to learn to identify and avoid noxious stimuli while moving towards perceived rewarding stimuli.

At the beginning of the training phase, the insect will not be aware of which stimuli are to be avoided or pursued. Learning occurs through reward-and-punishment classical conditioning [47]. The experiment will compare the learning behaviour of the virtual brain in both systems and its ability to adapt to the changing environment.

6.2 Designing the associative neural circuit

The neural circuit presented in Figure 3.4 can be used to implement a simple neural circuit to control the movement of the artificial insect, in such a way that it would learn that whenever a neutral stimulus in neurons A or B is presented, a given action associated to M will be performed. For instance: after learning the association between the stimulus perceived from A and the conditioned action performed by M, if A was connected to a proximity sensor and M was connected to a motor (actuator), then the activation of the proximity sensor would produce a response in the actuator connected to M. Thus, a simple system to avoid obstacles could be implemented using the pair wise association between the two neurons. Although, on its own, this circuit only allows a limited margin of actions (trigger reflex or not) in response to input stimuli, it can be taken as a single building block which implemented within a larger neural topology can produce more sophisticated behaviours.

Connecting A and B from the circuit in Figure 3.4 with a second motoneuron R will allow the initially neutral stimuli perceived by neurons A and B, to be associated (or dissociated) with the corresponding actions elicited by R and M. The new neural circuit with two motoneurons is illustrated below in Figure 6.1.

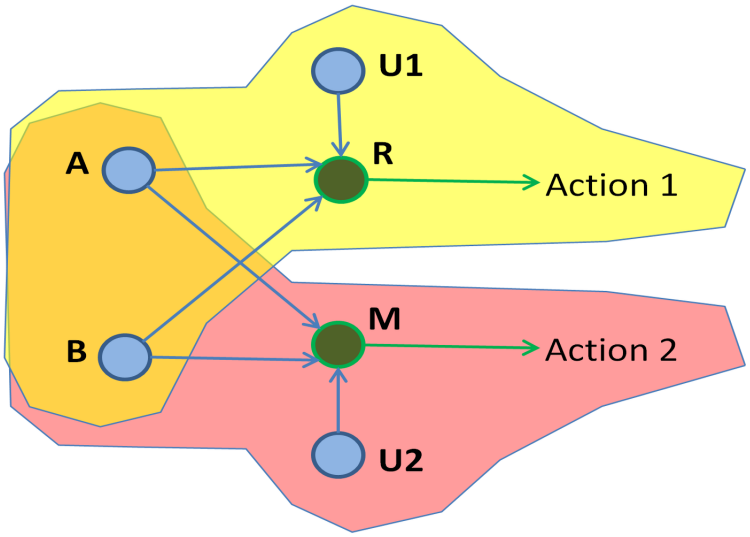


Figure 6.1: Neural circuit having two input neurons A and B both connected to motoneurons R and M. $U1$ and $U2$ are input neurons for the unconditioned stimuli eliciting reflex responses in R and M respectively.

As seen in Figure 6.1, the new neural topology is built using two instances of the associative neural circuit shown in Figure 3.4 The top part a) contains the sub-circuit which creates the association between the input stimuli received in A, B and the action elicited by R (Action 1). The bottom part b) contains the sub-circuit which creates the association between the input stimuli received in A, B and the action elicited by M (Action 2). Although both sub-circuits share the same input neurons A and B, the elicited behaviour in R and M will depend on the firing-times correlation between the neutral (conditioned) inputs A, B and the unconditioned neurons $U1$ and $U2$, *e.g.*: input in A could be highly excitatory for M resulting in the execution of Action 2 but have little or no effect in R, hence with no effect on Action 1.

In Figure 6.1 both sub-circuits a) and b) are concurrent meaning that both Action 1 and Action 2 can be performed at the same time if the same inputs in a) and b) are reinforced as excitatory in both sub-circuits. This behaviour however can be inconvenient if the system is expected to perform one action at the time. Inhibitory synapses between sub-circuits provide a control mechanism in cases where actions are mutually exclusive. Figure 6.2 below illustrates how the inhibition mechanism can be applied in the neural circuit described so far.

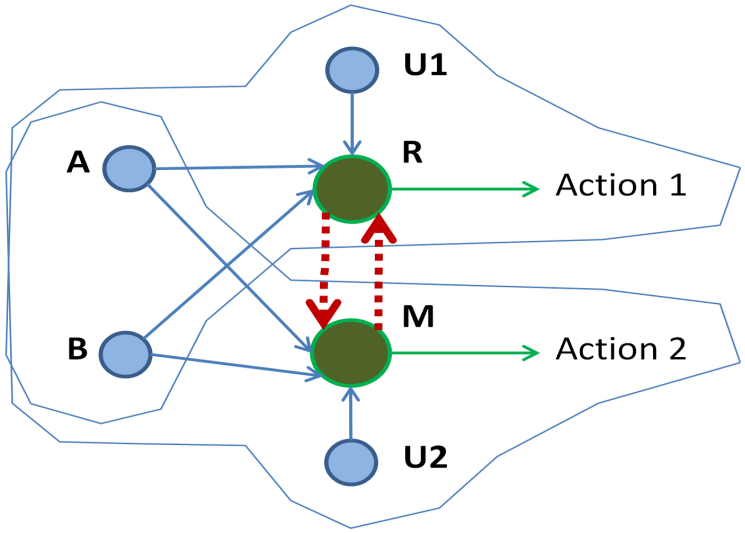


Figure 6.2: Neural circuit with two mutually inhibitory sub-circuits.

The mutually inhibitory synapses in motoneurons R an M work as a winner-take-all mechanism where the first firing neuron elicits its corresponding action while avoiding the concurrent activation of the other sub-circuit(s).

6.3 From a neural-circuit model to a virtual insect brain

In this Section the neural circuit in Figure 6.2 will be used as a model to implement a simple micro-brain to control the motion of the virtual insect in a simulated two-dimensional environment. The micro-brain of the virtual insect is able to process three types of sensorial information: (1) visual, (2) pain and (3) pleasant or rewarding sensation. The visual information is acquired through three receptors (see Figure 6.3) where each one of them is sensitive to one specific colour (red, black or green). Each visual receptor is connected with one afferent neuron which propagates the input pulses towards the motoneurons. Pain is elicited by a nociceptor whenever the insect collides with a wall (black patches) or a predator (red patches). Finally, a rewarding or pleasant sensation is elicited when the insect gets in direct contact with a food source (green patches).

	Neurons	
	R	M
Membrane Resting Potential	-65	-65
Membrane Firing Threshold	-55	-55
Equilibrium Concentration of EMS	1.0	1.0
Equilibrium Concentration of PMS	0.001	0.001
Initial Synaptic weight from Neuron A	5.0	5.0
Initial Synaptic weight from Neuron B	5.0	5.0
Initial Synaptic weight from Neuron C	5.0	5.0
Synaptic weight from Modulatory Neuron EM	-1.0	-1.0
Synaptic weight from Modulatory Neuron PM	1.0	1.0
Minimum synaptic weight from PD Neurons	1.0	1.0
Maximum synaptic weight from PD Neurons	9.0	9.0

Table 6.1: Experimental parameters for motoneurons R and M

Neuronal Parameters

Table 6.1 shows the parameters for membrane potential, plasticity (learning) and heterosynaptic modulation in both motoneurons R and M. The first two parameters: 'Membrane Resting Potential' and 'Membrane Firing Threshold' are set according to what is considered the typical voltages in biological neurons [82].

The equilibrium concentration of EMS is set to 1 as the default factor of excitability. The equilibrium concentration of PMS (default plasticity factor) is set to 0.001 in both neurons in order to reduce the plasticity (learning) in the absence of modulatory signals. The initial synaptic weights from neurons A, B and C are initialized with 5 as an intermediate value between the minimum (1.0) and maximum (9.0) allowed synaptic weights. These minimum and maximum values are determined considering the following: (1) the number of incoming synapses arriving from different neurons at each motoneuron and (2) the required change in the membrane potential to make the neuron fire. This is given by $FiringThreshold - RestingPotential$ (which is 10.0 in neurons M and R). Given that, a synapse with a weight of 1.0 will have a very low probability of activating a postsynaptic neuron on its own even when firing multiple spikes. On the other hand, a synapse with a weight of 9.0 will have a high probability of activating the postsynaptic neuron with just a few pulses.

The experiments described in the next sections of this chapter use these parameters for the implementation of the neural circuits.

6.3.1 Non-Heterosynaptic / Non-modulated case

Sensory - Motor Architecture

Figure 6.3 illustrates the complete neural architecture of the virtual insect without using the heterosynaptic mechanisms of plasticity and excitability modulation. There are three photoreceptors, one for each colour (red, black and green) synapsing with neurons A, B and C, respectively. A B and C work as afferent neurons carrying the visual sensory information towards the layer of motoneurons. The nociceptor (P) and reward-related sensor (F) elicit the immediate activation of motoneurons R and M respectively. Both neurons R and M execute the reflex behaviour through the activation of Actuator_1 and Actuator_2 for the actions of rotation and moving-forward respectively.

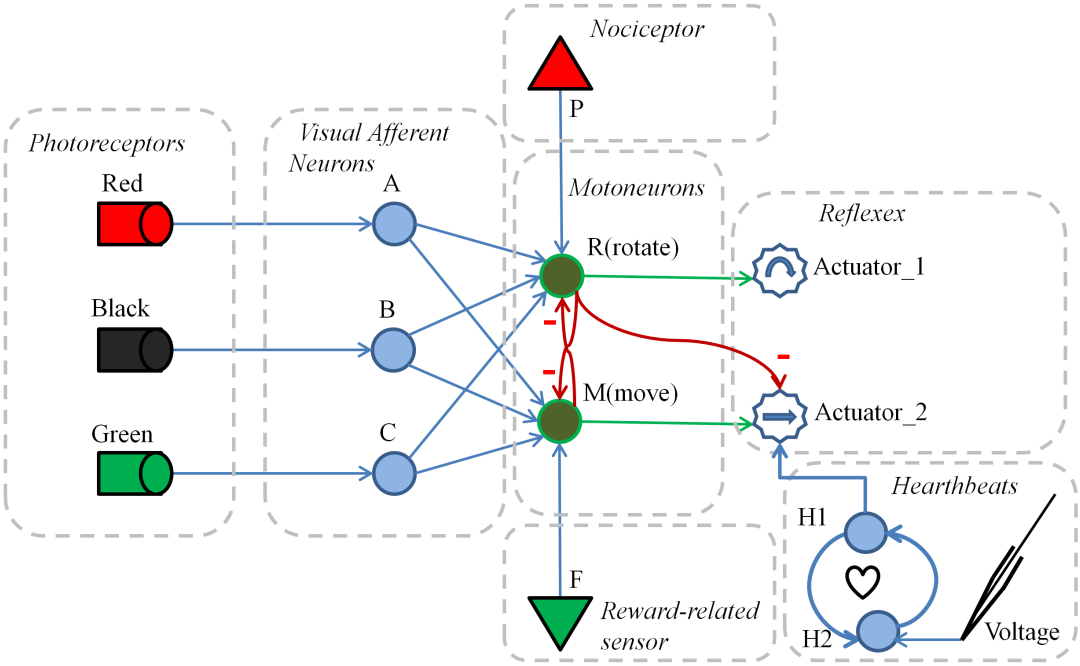


Figure 6.3: Neuro-inspired controller of the virtual insect with the traditional homosynaptic implementation.

The number of rotation degrees as well as the number of movement units are set in the simulation to $5°$ and 1 patch respectively. In order to keep the insect moving even in the absence of external stimuli, the Actuator_2 is connected to a neural oscillator sub-circuit composed of two neurons H1 and H2 performing the function of a pacemaker which sends a periodic pulse to the actuator neuron. The pacemaker is initiated by a pulse from an input neuron which represents an external input current (*i.e.* intracellular electrode). An inhibitory synapse from motoneuron R to Actuator_2 prevents the activation of the move actuator when the counteraction of rotation has been triggered shortly before.

6.3.2 Heterosynaptic / Modulated case

Sensory - Motor Architecture

Figure 6.4 below illustrates the neural architecture of the virtual insect using heterosynaptic plasticity and excitability modulation:

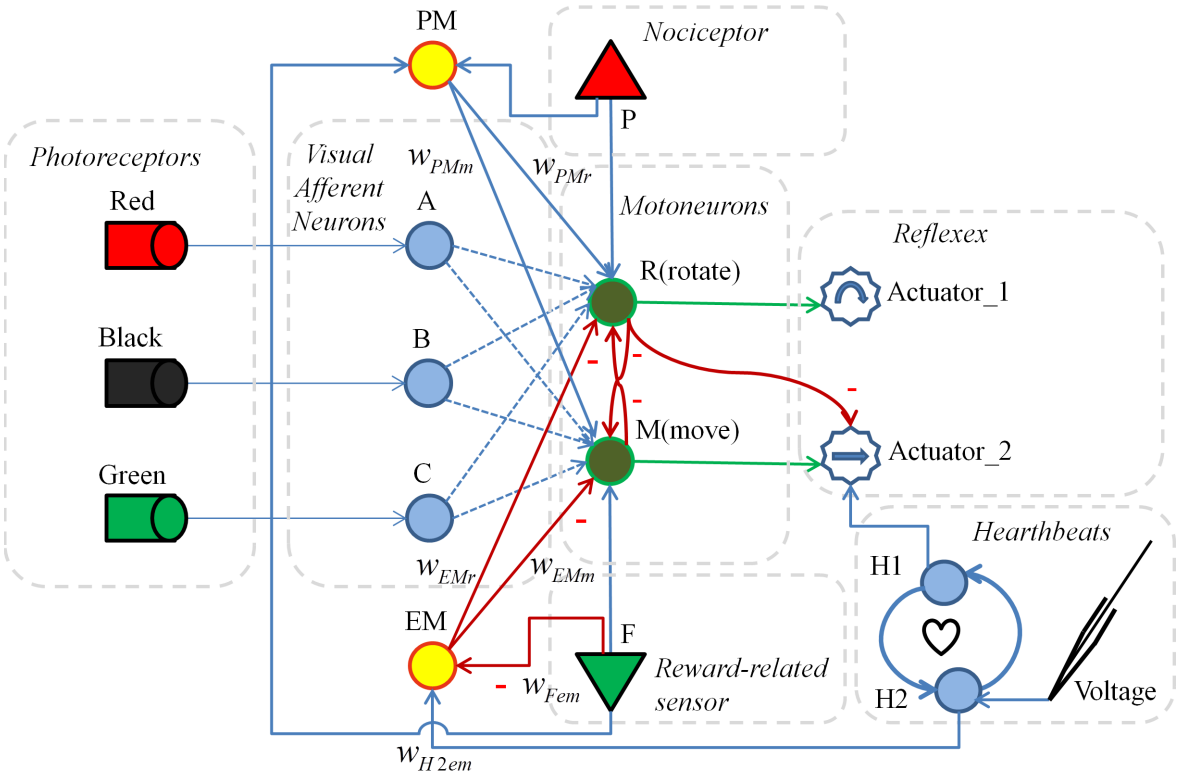


Figure 6.4: Neuro-inspired controller of the virtual insect with the proposed heterosynaptic implementation.

Excitability Modulatory Subcircuit

Compared to the previous circuit in Figure 6.3 the topology of the sub-network made up from PD neurons and receptors is still the same in Figure 6.4. The change in the dynamics of the system is mostly introduced by the modulatory effect of the neurons EM and PM. The neuron EM is an excitability modulatory neuron which is connected to motoneurons R and M through inhibitory synapses. The activation of EM leads to the transmission of a modulatory signal that decreases the concentration of the EMS in the postsynaptic neurons R and M. The amplitude of the excitability-inhibition effect is given by the weights W_{EMr}

and W_{EMm} for neurons R and M respectively. During the strong inhibition effect the motoneurons become significantly unresponsive to the pulses coming (from PD neurons) through synapses with affinity to the EMS. Given that the synapses with EMS affinity (illustrated with dashed lines in Figure 6.4) originate from the afferent neurons A, B and C, it implies that the perceived visual stimuli will have a weakened effect (or not effect at all) on the motor behaviour during a certain period of time after the activation of the modulatory neuron EM.

The rationale for the unresponsive motor interval is to allow the artificial insect to explore its surrounding environment for a short period of time without the bias of already learned conditioned stimulus-response behaviour. This should allow the insect to unlearn already acquired behaviour while learning to associate new stimulus-reflex behavioural pairs. As shown in Figure 6.4, neuron EM has an incoming synapse from PD neuron H1 which delivers periodic pulses acting as a hearthbeat. Because of the low synaptic weight W_{H1em} from H1 to EM, the activation of EM requires the arrival of several pulses during a long period of time in order to occur. In this way, the activation of EM is triggered periodically by H1 unless EM receives inputs from other incoming synapses. Neuron EM has an incoming inhibitory synapse from receptor F. This means that any encounter with a rewarding stimulus will disrupt or possibly reset the otherwise periodic activation behaviour of neuron EM.

The function of neuron EM including its incoming and outgoing synapses is to allow the system to change its behaviour in the absence of rewarding stimuli during a given period of time. Thus, for this experimental setup, the modulatory subsystem works as a "take the risk or starve" mechanism that temporarily allows the artificial insect to explore its surrounding world ignoring the previously learnt associations. This mechanism is inspired by the behaviour observed in nature where animals under extreme survival conditions (for instance, starvation, escaping from a predator or competing for mating opportunities) will try taking again certain routes that have already been associated with danger or other aversive responses. In other words, they give a second chance to an already failed or dangerous solution.

Plasticity Modulatory Subcircuit

The neuron PM is a plasticity modulatory neuron which is connected to motoneurons R and M through excitatory synapses. The activation of PM leads to the transmission of a modulatory signal that increases the concentration of the PMS in the postsynaptic neurons R and M. As shown in Table 6.1 the equilibrium concentration of PMS in motoneurons R and M is set at a very low value in order to ensure that plasticity in both neurons is almost neglected in the absence of a plasticity modulatory signal. As shown in Figure 6.4, neuron PM has incoming excitatory synapses from receptors P and F. Hence, any noxious or rewarding stimulus will trigger the activation of the PM neuron and consequently allow plasticity in the motoneurons R and M.

From the point of view of the behaviour of the artificial insect, the reason for the low equilibrium concentration of PMS in R and M is to allow the insect to only learn the association between noxious and rewarding stimuli with their corresponding elicited reflexes when the reflex action is triggered by a receptor associated with an unconditioned stimulus input (*i.e.* nociceptive or rewarding) and not by previously conditioned neurons (for instance, visual afferent neurons). This is to prevent the positive plasticity feedback loop that emerges when the activation of the afferent neurons (A, B or C) reinforces their own synapses producing runaway dynamics in the long term.

In order for neuron PM to elicit plasticity in the postsynaptic motoneurons R and M, it is necessary that the modulatory signal arrives at R and M before they become activated (initiate an action potential). This is because in a PD neuron the STDP processing of spikes with pre-before-post timing occurs during the initiation of the action potential (see InitiateActionPotential component in figure 5.10 of chapter 5). Otherwise, if the modulatory signal arrives just after the activation of the postsynaptic neuron, then all the incoming spikes from synapses with PMS affinity that preceded the postsynaptic action potential are processed with a lower PMS concentration. In contrast, the spikes that arrive after the modulatory signal will be processed with a significantly higher plasticity factor. This would result in an asymmetric processing of the STDP learning window, leading to a continuous LTD in the affected synapses.

To satisfy the timing constraints the synaptic (axonal) delays are adjusted in such a way that the modulatory signal reaches the target neurons before the spike potentials (transmitted by the activated receptors). Figure 6.5 illustrates the required timings for the synapses between neuron PM, nociceptor P and motoneuron R.

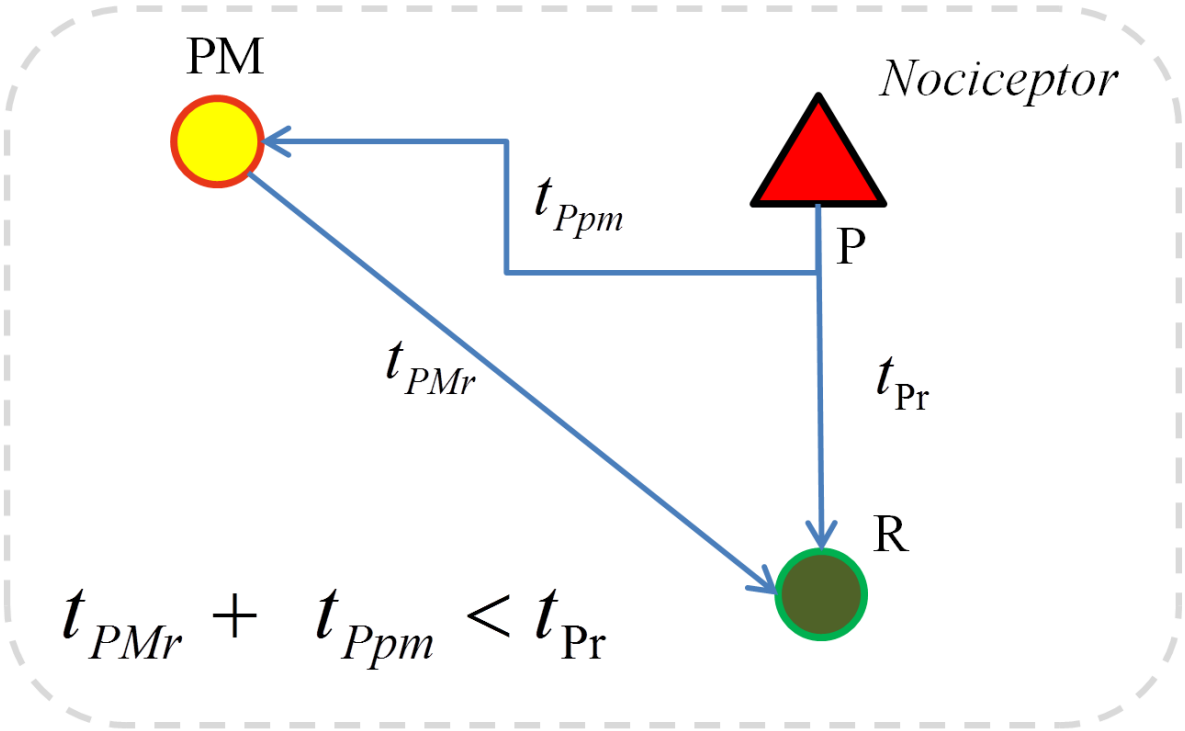


Figure 6.5: Timing constraints for the subcircuit formed by motoneuron R, modulatory neuron PM and receptor P.

Figure 6.5 shows that the synaptic delay t_{Pr} from Nociceptor P to motoneuron R is larger than the synaptic delay t_{Ppm} from Nociceptor P to neuron PM added to the synaptic delay t_{PMr} from PM to motoneuron R. This topology assumes that there is no delay inside neuron PM (during the activation and firing process). Otherwise the synaptic delay t_{Pr} would have to take neuronal delays into account.

The same timing constraints must be taken into account for the subcircuit involving the neuron PM, receptor F and motoneuron M. This is illustrated below in Figure 6.6.

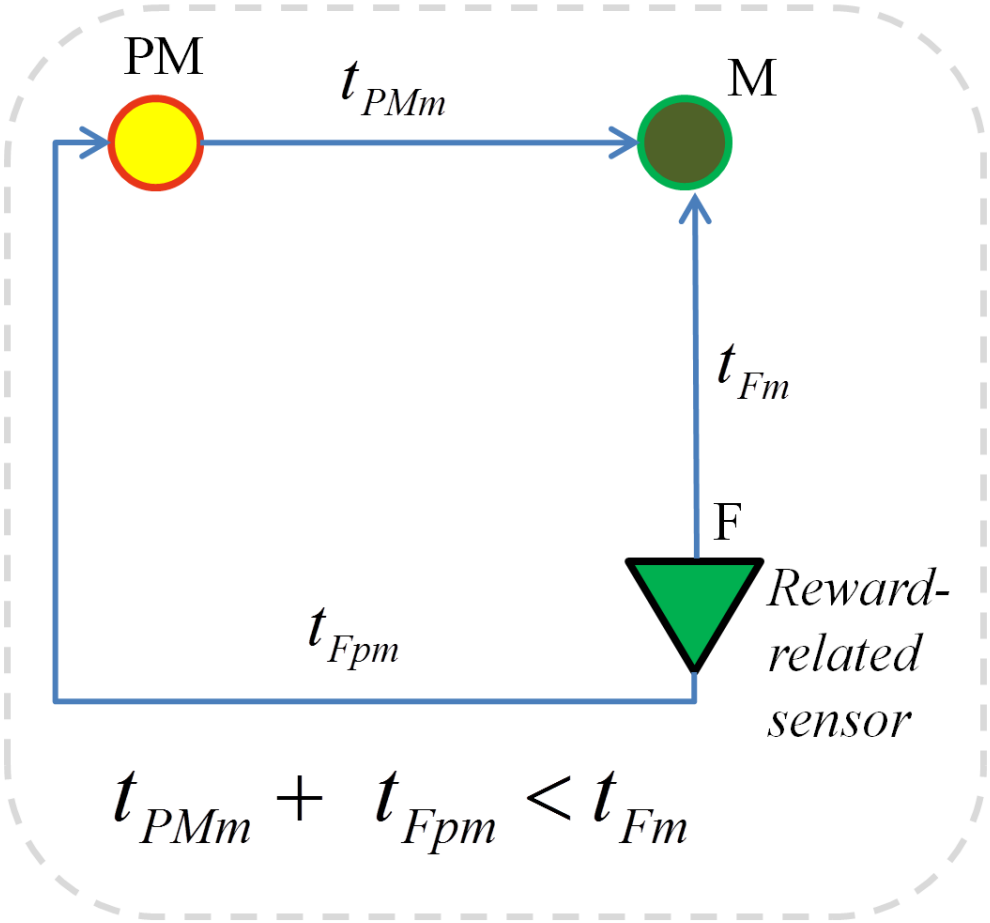


Figure 6.6: Timing constraints for the subcircuit formed by motoneuron M, modulatory neuron PM and receptor F.

Figure 6.6 shows that the synaptic delay t_{Fm} from the reward-related sensor F to motoneuron M is larger than the synaptic delay t_{Fpm} from receptor F to neuron PM added to the synaptic delay t_{PMm} from PM to motoneuron M. Again, as in Figure 6.5 this topology neglects any processing delay inside neuron PM.

6.4 Design of the Experiment

In order to test the proposed artificial insect controlled by a heterosynaptic SNN system, the following characteristics will be measured and compared with a non-heterosynaptic (non-modulated) implementation:

1. The association of a conditioned (neutral) input with a reflex response will be measured in terms of the evolving synaptic efficacy between afferent and motoneurons and the resulting motion behaviour (number of observed collisions as a measurement of error).

2. The unlearning of previous acquired input-response associations, for instance stimuli that were noxious become rewarding and vice versa. This will also be measured in terms of synaptic efficacy between afferent and motoneurons and the resulting motion behaviour.

In order to measure the characteristics mentioned in (1) and (2) the following two experimental conditions have been employed:

1. Associating visual neutral inputs with reflex responses in both heterosynaptic and non-heterosynaptic SNN System. Patches with the colours black and red will be associated with a noxious effect following a contact or collision with one of them. Patches with the colour green will be associated with a reward (or pleasant) effect after contact. The reflex behaviour for the activated nociceptor is the activation of actuator R (rotation). The reflex behaviour for the activated reward sensor is the activation of actuator M (move forward).

2. Unlearning of previous acquired visual-input with reinforced reflex-response associations in both heterosynaptic and non-heterosynaptic SNN System. In the first phase, patches with the colours black and red will be associated with a noxious effect following a contact or collision with one of them. Patches with the colour green will be associated with a reward (or pleasant) effect after contact. After training (first phase completed), in the second phase, patches with the colour green will be associated with a noxious effect following contact or collision. The reflex behaviours are the same as in experimental condition 1.

3. Unlearning of previous acquired visual-input with aversive response associations in both heterosynaptic and non-heterosynaptic SNN System. In the first phase, patches with the colours black and red will be attached with a noxious effect following a

contact or collision with one of them. Patches with the colour green will be attached with a reward (or pleasant) effect after contact. After training (first phase completed), in the second phase, patches with the colour red will be attached with a rewarding effect following contact or collision. The reflex behaviours are the same as in the experimental condition 1.

6.5 Implementation in Netlogo

A SNN engine was created in order to implement the SNN model described in this chapter as well as other experiments described in this thesis. The coding of the engine was done entirely in Netlogo language as a Netlogo model (See sample code in the appendix). Netlogo is a software application that provides an integrated environment for the simulation and programming of multi-agent models and the study of emergent behavior in complex systems [87]. The netlogo programming language provides a set of primitives which allows the agents to perceive and modify their virtual world and also to communicate and interact with other agents. Apart from its simplicity, one of the main advantages of using Netlogo in this work, is that it allows to monitor and manipulate on each single simulation iteration the state of each element of the neural circuit including: (1) neurons and their internal variables, (2) synapses and their parameters (efficacy and delay) and (3) ongoing pulses. Manipulation of the neural circuit can be done with commands given through the observer prompt or by using the agent monitoring tool provided by the Netlogo GUI.

6.5.1 Modelling with Agents and Patches

In Netlogo (version 5.3 at the time) there are four main types of agents. These agents are: Turtles, patches, links and the observer [87]. The turtles are motile individual agents from where other user-defined agents can be derived. Turtles can be created or destroyed at any time during the simulation. Different 'breeds' (user defined types) of turtles can be defined, each one having its own attributes and behaviours. Patches are non motile agents. Each patch has fix coordinates corresponding to a unique position in the virtual

2-dimensional plane of the simulated world. Patches can have their own user-defined attributes and behaviours in a similar way to turtles, however, with the difference that they can not move. A link is a type of agent that represents an association between two turtles-agents. Thus, a link has two main attributes each one representing a turtle-agent at each one of its ends. Similarly to turtles and patches a link may have its own user defined attributes and behaviours. The link is non motile and is shown as a line connecting the two turtles. The observer, is a single and unique agent (no agents can be derived from it) that can create, observe and command other type of agents.

In this experimental setup the simulated insect-like agent is represented by a turtle agent. Neurons are implemented as turtle agents using a defined 'breed' that implement the attributes and dynamics of the artificial neuron. Synapses on the other hand are implemented as agents of type link. All simulated entities including the artificial insect, neurons and synapses have their own variables and functions that can be manipulated using standard Netlogo commands. The Netlogo virtual world consists of a two dimensional grid of patches where each patch corresponds to a point (x; y) in the plane. In a similar way to the turtles, the patches own a set of primitives which allow the manipulation of their characteristics and also the programming of new functionalities and their interaction with other agents. The visualization of the insect and its environment is done through the Netlogo's world-view interface. The virtual world of the insect is an ensemble of patches of four different colours, where each one of them is associated with a different type of stimulus. As described before, black and red patches are both used to represent harmful stimulus. Thus, if the insect is positioned on a black or red patch, this will trigger a reaction in the insect's nociceptor (pain sensor) and its corresponding neural pathway (see figure 6.3). On the other hand, green patches trigger a reaction in the reward sensor of the insect whenever it is positioned on one of them. White patches represent empty spaces and do not trigger any sensory information in the insect.

6.5.2 The Simulated Insect's World

Timing and Movement

The simulation follows a clock-driven approach where the elapsed time is given by a counter whose value is increased at a discrete time step (Netlogo 'tick') in every cycle or iteration when the simulation is executed. The movement of the insect is not only determined by its neural controller but it also depends on the rotation degrees and number of patches at which the insect moves on each iteration or tick. For the experiments described in this chapter, in one iteration or tick the number of patches that the insect can move in response to the activation of motoneuron 'M' is 1-patch. On the other hand the amount of rotation in response to the activation of motoneuron 'R' is 5 degrees.

Space and Stimuli

As mentioned before the Netlogo virtual world consists of a two dimensional grid of patches where each patch corresponds to a point (with coordinates x; y) in the plane [87]. The virtual world of the insect is an ensemble of patches of four different colours, where each one of them is associated with a different type of stimulus.

During the first experiment black and red patches are both used to represent harmful stimulus (*e.g.* walls and predators). Thus, if the insect is positioned on a black or red patch, the simulation will activate the insect's nociceptor (activating pain sensor) and its corresponding neural pathway. On the other hand, green patches trigger the activation of the reward sensor whenever the insect is positioned on one of these patches. White patches represent empty spaces and do not trigger any sensory information in the insect. The type of stimulus associated to each colour can be changed any time during the simulation allowing to carry out the different experimental conditions. This is done using the implemented GUI interface for switching on and off the association between colours and the type of stimulus elicited on the insect. (See Figure 6.19).

The visualisation of the simulation is divided in two areas inside the Netlogo's world-view interface: (1) The Neural circuit topology which is shown on the left half of the screen. And (2) the insect and its environment which are shown on the right half side of the screen. This is shown in Figure 6.7 as follows:

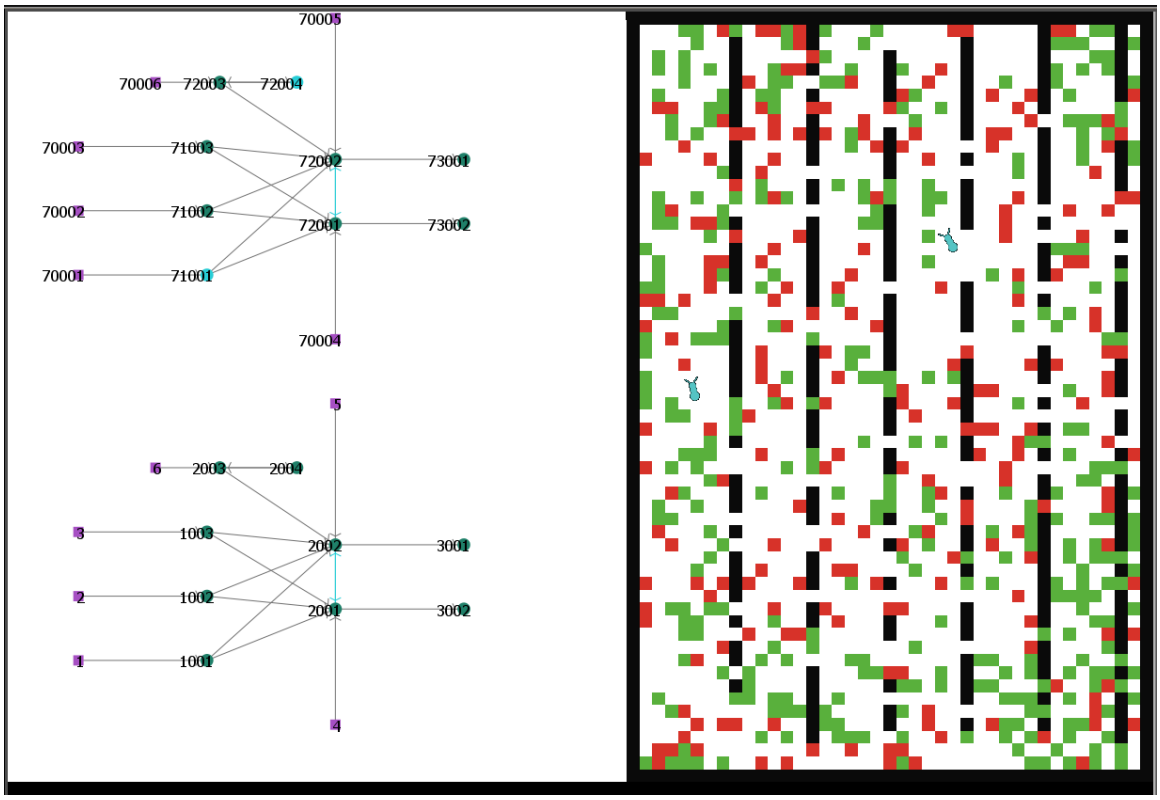


Figure 6.7: Visualization area in the simulation environment

The topology screen reflects any change (adding or removing components) done to the neural circuit in each iteration. The world screen on the right side, shows the simulated virtual world including patches of four different colours; white, black, red and green representing empty spaces, walls, harmful and rewarding stimuli respectively. The virtual insect is represented with an ant shaped agent that starts moving once the simulation is initiated. In addition to the simulated world, Netlogo provides several interface objects for plotting and monitoring agents behaviour. In the presented simulation, two plots have been implemented in order to visualize the change over time of the membrane potential of any two neurons selected by the experimenter.

6.6 Results

This Section describes the observed behaviour of both experimental heterosynaptic and non-heterosynaptic implementations in the previously presented experimental conditions:

6.6.1 Experiment 1: Associating visual neutral inputs with reflex responses in both heterosynaptic and non-heterosynaptic SNN system.

As illustrated below in Figure 6.8, for this experiment, in the simulation environment black and red colours patches were associated with harmful or noxious stimuli while green colour patches were associated with reward.

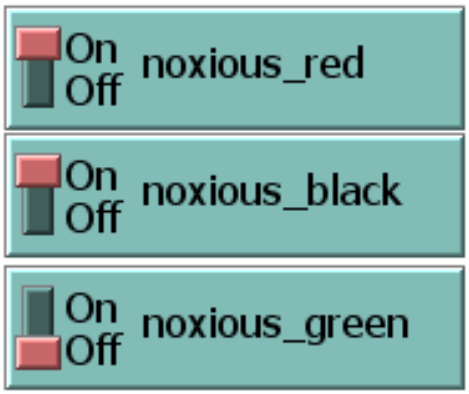
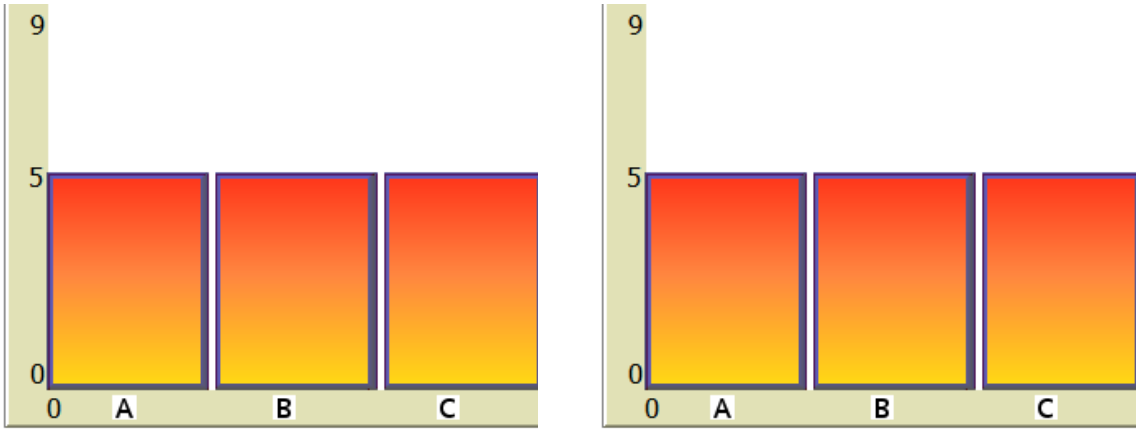


Figure 6.8: Switches in the control area of the simulation environment: The green switch set to Off indicates that no pain stimuli will be triggered in case of collision with a patch of this colour.

The efficacy (weight) of each one of the synapses from the afferent visual neurons A, B and C with the motoneurons R and M was initialized with a value of 5.0 as illustrated in Figure 6.9 for both neurons R (on the left) and M (on the right).

At the beginning of the training phase (see 6.10) the insect moves along the virtual-world colliding indiscriminately with all types of patches. The insect is repositioned in its initial coordinates every time it reaches the virtual-world boundaries. As the training phase progresses it can be seen that the trajectories lengthen as the insect learns to associate the red and black patches with harmful stimuli and consequently to avoid them (See 6.11).



(a) Synaptic weights from neurons A,B and C with postsynaptic neuron R

(b) Synaptic weights from neurons A,B and C with postsynaptic neuron M

Figure 6.9

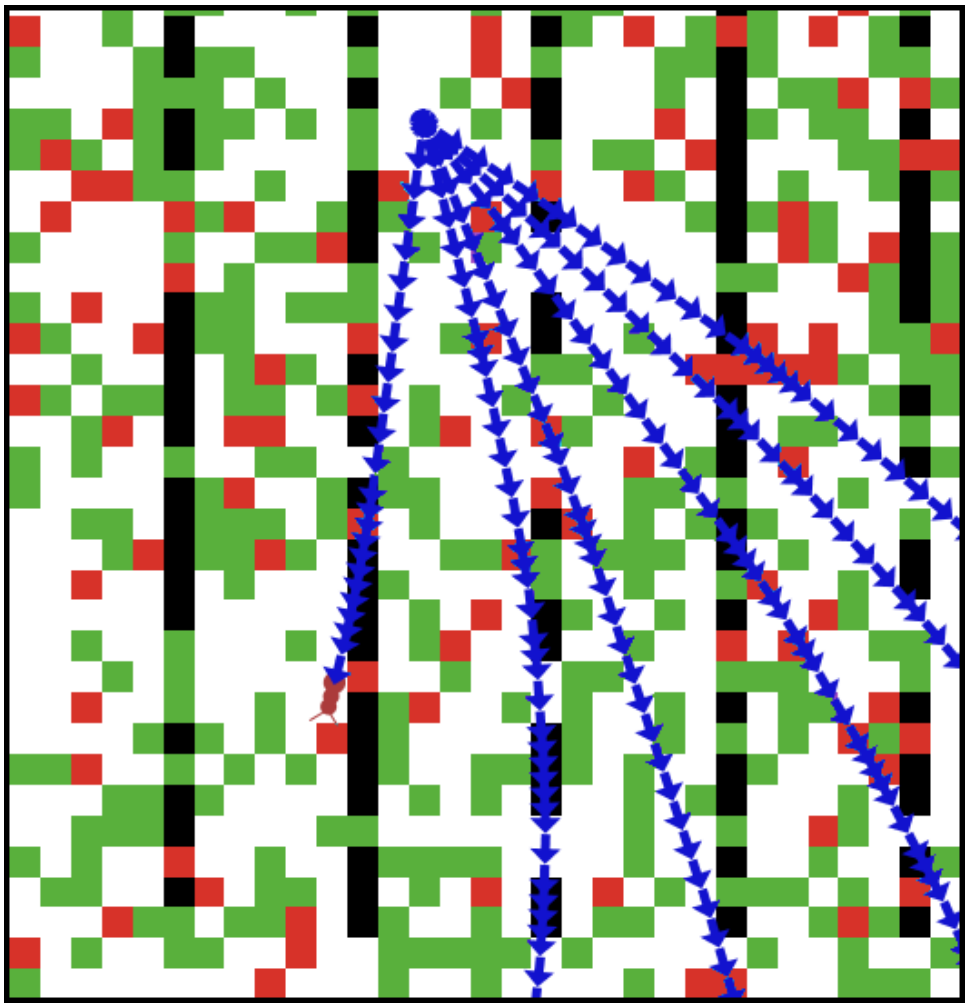


Figure 6.10: Short trajectories at the beginning of the training phase. The insect collides and escapes the world repeatedly. Each blue arrow indicates the movement of the insect on a patch at one tick.

After approximately 25000 iterations in both non-heterosynaptic and heterosynaptic cases, the insect moves collision free most of the time, avoiding red and black patches while accelerating when a green patch is in front of its sight line. Figure 6.11 illustrates the motion behavior after training:

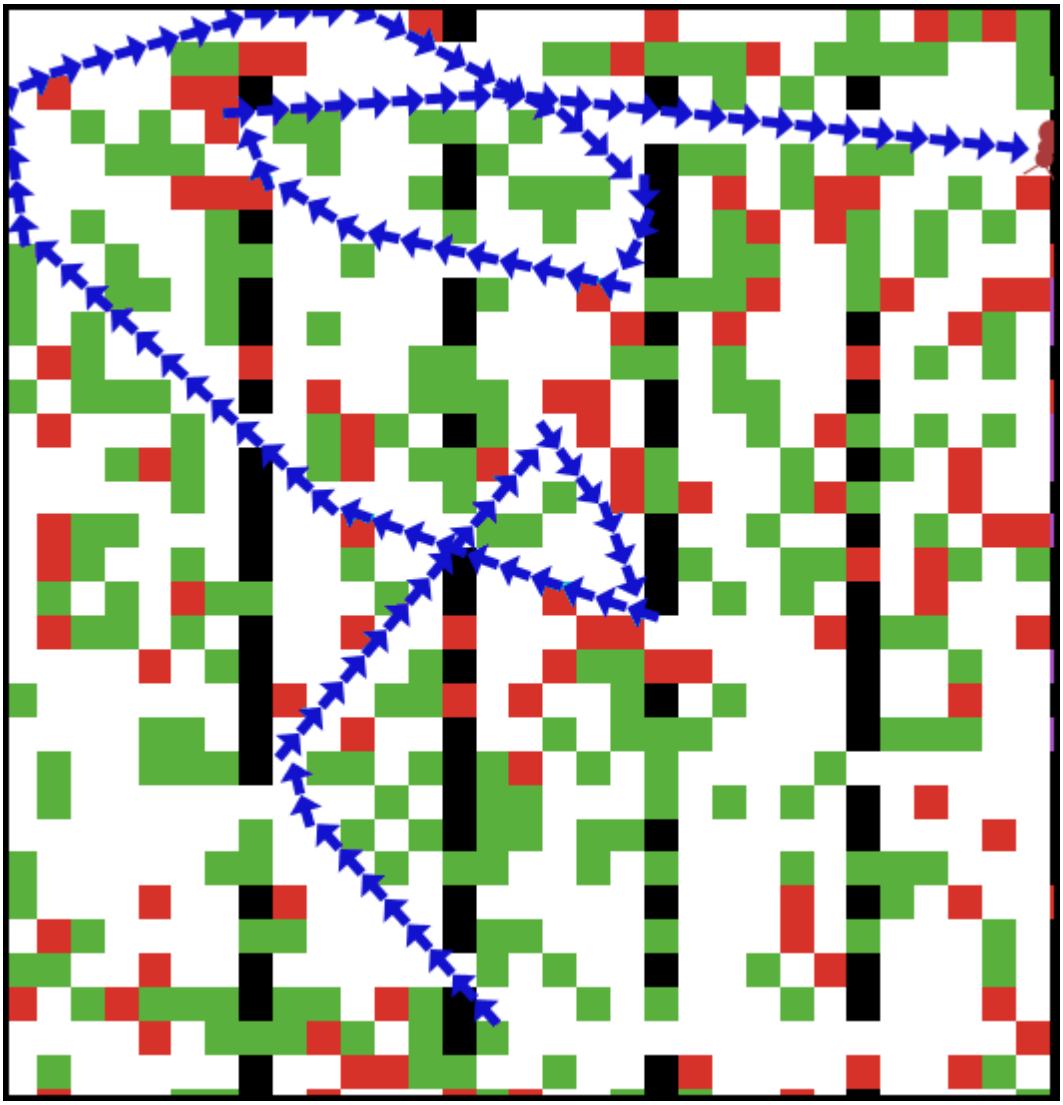


Figure 6.11: Long trajectory shows the insect avoiding red and black patches. Each blue arrow indicates the movement of the insect on a patch at one tick.

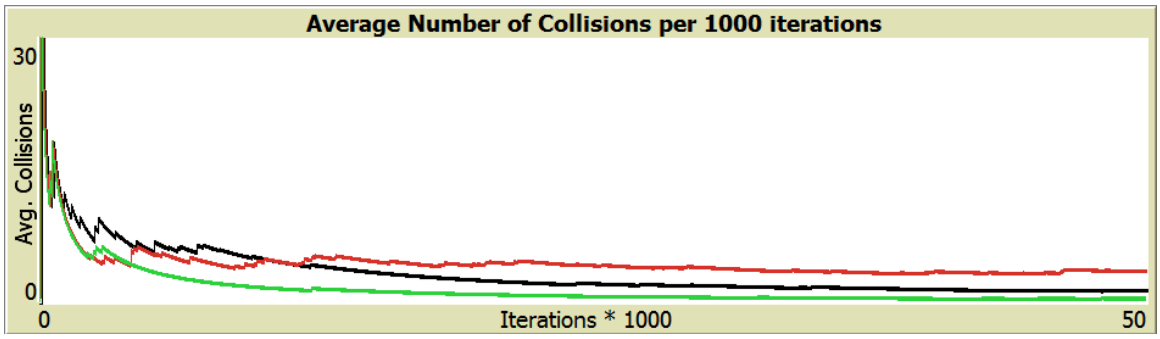


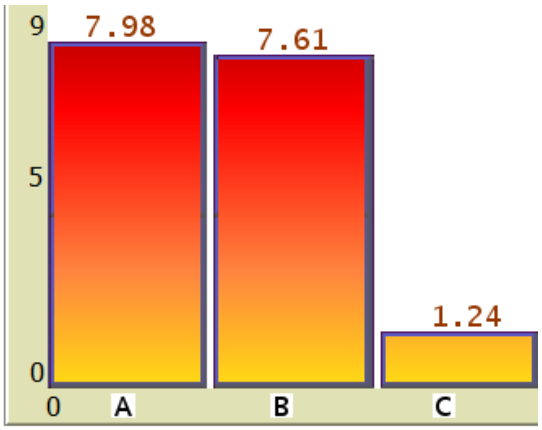
Figure 6.12: Average number of collisions during the simulation.

Figure 6.12 above illustrates the average number of collisions over time with red and black patches. The three plot-lines represent the information as follow:

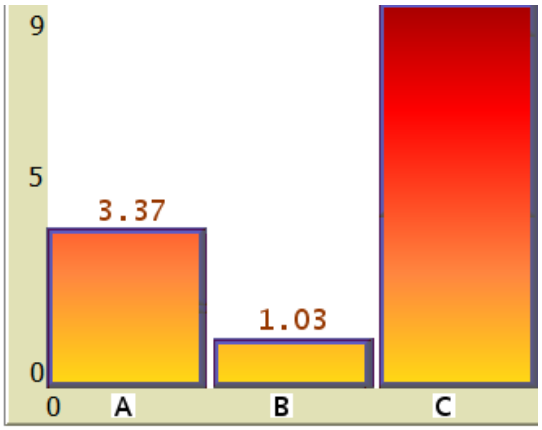
- The black line represents the non-heterosynaptic system.
- the red line represents the experimental heterosynaptic system with excitability and plasticity modulation.
- The green line represents the experimental heterosynaptic system with plasticity modulation but no excitability modulation.

As seen in Figure 6.12, the average number of collisions with noxious stimuli reaches its minimum steady value at about 25000 simulation-iterations in both non-heterosynaptic implementation (1.8 average collisions in 1000 iterations) and heterosynaptic implementation with excitability and plasticity modulation (3.6 average collisions in 1000 iterations). The green line shows that at the same number of iterations the average number of collisions is about 0.5 for the implementation with heterosynaptic plasticity but no excitability modulation. At first glance it seems that the implementation represented by the black line is performing better in terms of incurring in fewer collisions when compared to the red line. However, the evolution of the green line, which shows the lowest number of collisions, suggests that the increased error in the experimental heterosynaptic system (red line) is related to the fact that the underlying system is characterized by a risk prone behaviour. As explained before at the beginning of Section 5.2.2 the risk prone behaviour of the insect is originated during the (down) modulation of the excitability in the motoneurons M and R. It means that the insect will attempt to approach already known noxious (red and black)

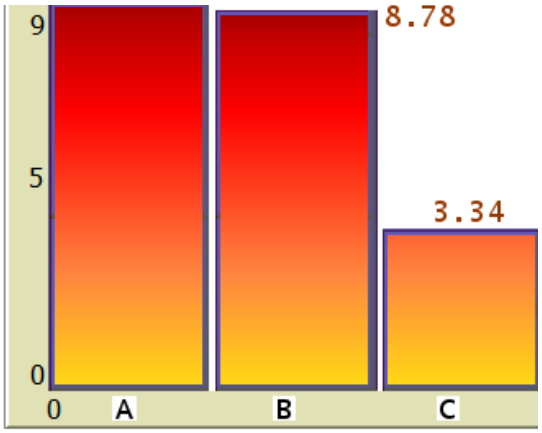
patches in its search for food when the conditions are met. For instance, after a given period of time without finding food (green) patches. Thus, the occasional trial and error behaviour explains the increased number of collisions. In order to gain a better picture of the learning mechanism inside both neural circuits implementations, Figure 6.13 shows the resulting synaptic weights between the visual afferent neurons A,B and C and motoneurons R and M after 25000 iterations:



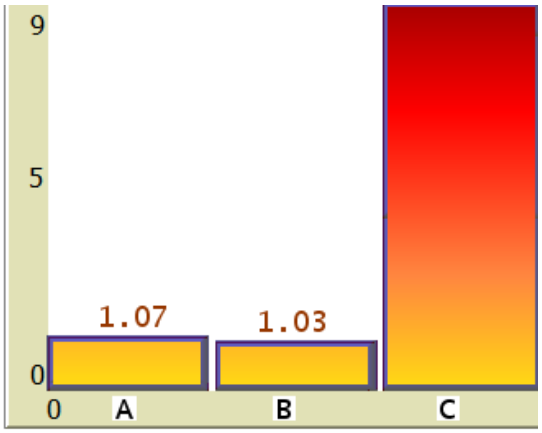
(a) Resulting synaptic weights from neurons A, B and C to neuron R after 25000 iterations in the non-heterosynaptic system.



(b) Resulting synaptic weights from neurons A, B and C to neuron M after 25000 iterations in the non-heterosynaptic system.



(c) Resulting synaptic weights from neurons A, B and C to neuron R after 25000 iterations in the heterosynaptic system.



(d) Resulting synaptic weights from neurons A, B and C to neuron M after 25000 iterations in the heterosynaptic system.

Figure 6.13

Figures 6.13(a) and (b) show the resulting synaptic weights in the motoneurons R an M after running the simulation for 25000 iterations in the non-heterosynaptic implementation. Figure 6.13(a) shows that the weights of the incoming synapses from the visual afferent neurons A and B have been significantly potentiated in the motoneuron R. Since neurons A

and B are activated by black and red photoreceptors respectively, This indicates that visual stimuli associated with these two colours may result in the activation of motoneuron R, this by taking into account that the potentiated weights are above 80% of the maximum value set for the incoming synapses in neuron R (see Table 6.1). On the other hand, the weight of the incoming synapse from the visual afferent neuron C has been strongly depressed almost reaching the minimum value for synaptic weights in neuron R. This means that the activation of neuron R through incoming EPSPs from presynaptic neuron C (only) is very unlikely. Thus, given that neuron C is activated by the green photoreceptor, this indicates that visual stimuli associated with green will have little effect on the activation of R.

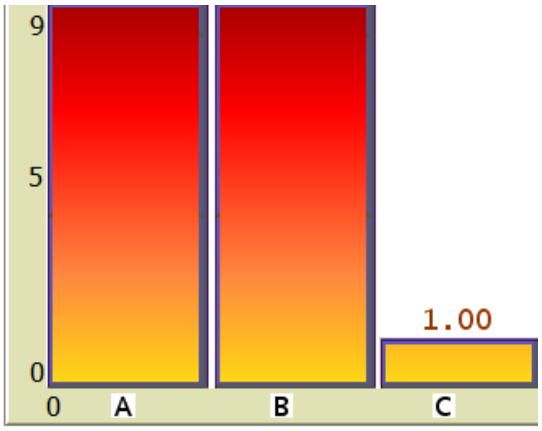
Figure 6.13(b) shows that the weights of the incoming synapses from the visual afferent neurons A and B have been strongly depressed in the motoneuron M. Thus, visual stimuli associated with black and red colours will have little effect in the activation of M. In contrast, the weight of the incoming synapse from the visual afferent neuron C has been potentiated to its maximum value. Thus, green visual stimuli will likely result in the activation of motoneuron M.

Figures 6.13(c) and (d) show the resulting synaptic weights in the motoneurons R an M after running the simulation for 25000 iterations in the experimental heterosynaptic implementation. Figures 6.13(c) and (d) show a synaptic weight distribution similar to the ones shown in figure 6.13(a) and (b) respectively. It can be inferred from these similarities that the resulting behaviour between the visual afferent neurons A,B and C and the motoneurons R and M is consistent in both experimental implementations.

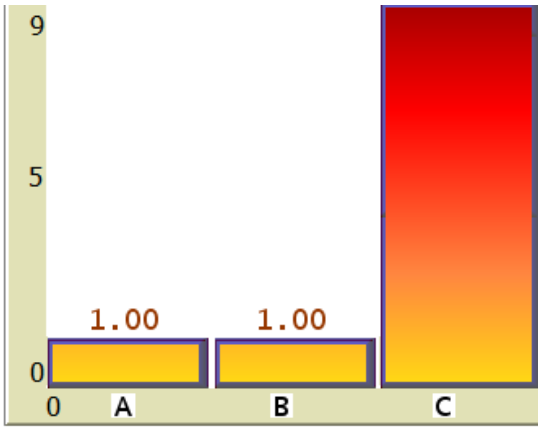
The similarities in the synaptic weight distribution also indicate that there are no significant differences between the learning rates in both experimental systems. This confirms that the increased number of collisions in the heterosynaptic implementation shown in Figure 6.12 is not the result of slower plasticity dynamics but is due to the reasons exposed before (*i.e.* risk prone behaviour).

6.6.2 Experiment 2: Unlearning of previous acquired visual-input with reinforced reflex-response associations in both heterosynaptic and non-heterosynaptic SNN Systems.

Both implementations of the artificial insect were trained in the same way as indicated in the previous experiment associating black and red patches with noxious stimuli and green patches with a reward related stimulus. However, for this (second) experiment the simulation was run for 500 thousand iterations in both experimental cases to ensure that there was no further changes in the synaptic weights of the incoming synapses in the motoneurons R and M. The reason for this is to allow both implementations to converge to the same solution (*i.e.* the same synaptic weights) and thereby ensure that both systems have the same initial conditions before starting the second phase of the experiment. Figure 6.14 below shows the resulting synaptic weights in the heterosynaptic and non-heterosynaptic implementations:



(a) Resulting synaptic weights from neurons A, B and C to neuron R in both experimental implementations.



(b) Resulting synaptic weights from neurons A, B and C to neuron M in both experimental implementations.

Figure 6.14

The number of iterations required to reach the above illustrated synaptic weights (without further significant change) was about 42000 and 50000 iterations for the heterosynaptic and non-heterosynaptic implementations respectively.

In the second phase of the experiment, the green patches were associated with noxious stimuli in the simulation environment as illustrated next in Figure 6.15:

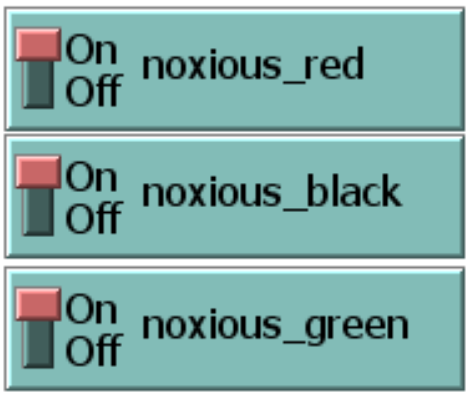
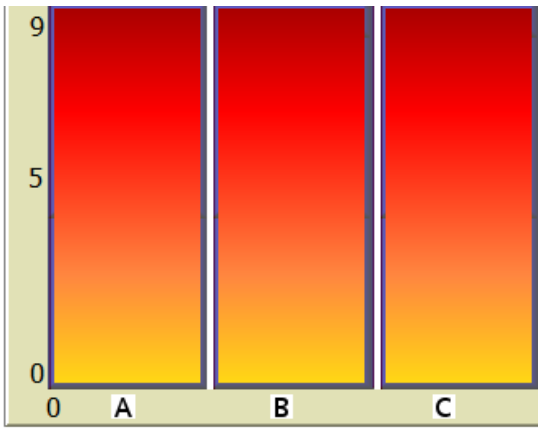


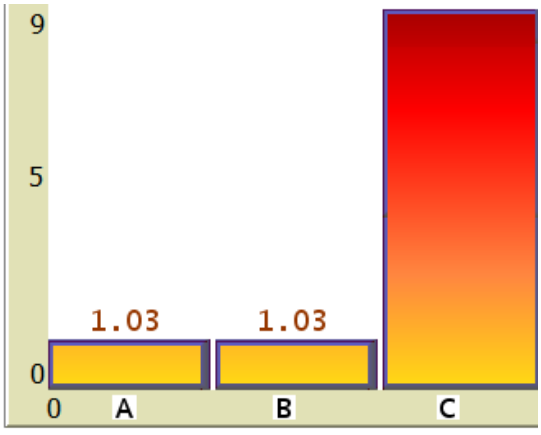
Figure 6.15: Switches in the control area of the simulation environment: Red, black and green switches set on 'On' indicate that pain stimuli will be triggered in case of collision with these colours.

After setting the parameters as indicated above, the simulation was resumed and run for another 1 million iterations in both experimental cases. This in order to observe the behaviour of the artificial insects with the new environment condition (green patches attached to noxious stimuli).

In the non-heterosynaptic implementation the artificial insect was able to associate the green patches with the nociceptive reflex response after about 200000 iterations. The resulting synaptic weights are illustrated in Figure 6.16 below:



(a) Resulting synaptic weights from neurons A, B and C to neuron R in the non-heterosynaptic implementation.

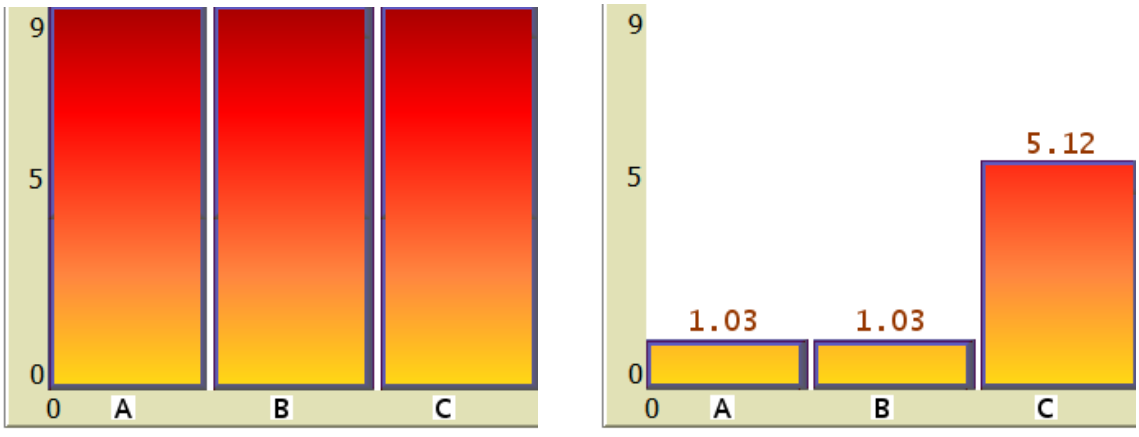


(b) Resulting synaptic weights from neurons A, B and C to neuron M in the non-heterosynaptic implementation.

Figure 6.16

Figure 6.16(a) shows that the incoming synapses from the visual afferent neurons A,B and C have been potentiated in motoneuron R reaching their maximum synaptic weight allowed by the system. This indicates that red, black and green visual stimuli will elicit a strong response in the action associated to motoneuron R.On the other hand, in the motoneuron M the incoming synapse from the afferent neuron C (Figure 6.16(b)) did not change with respect to the first phase of the experiment. Thus, the incoming synapse from neuron C is still strongly potentiated in neuron M, meaning that green visual stimuli will elicit strong responses in neuron M.The synapses from the (green) visual afferent neuron C have been strongly potentiated in both motoneurons R and M, having almost the same synaptic weights. Therefore, the resulting behaviour when green visual stimuli are presented will be more stochastic than deterministic in nature since the competition for activation between R and M will no longer depend on the synaptic efficacy but on the timing of the pulses arriving first at the target motoneurons.

In summary, the artificial insect in the non-heterosynaptic implementation has learnt to associate green visual stimuli with an aversive reflex response. However, it has failed to unlearn the previous association between green visual stimulus and the reward related response. In the heterosynaptic implementation the artificial insect was able to associate the green patches with the nociceptive reflex response after about 80000 iterations. The resulting synaptic weights are illustrated in Figure 6.17 below:



(a) Resulting synaptic weights from neurons A, B and C to neuron R in the heterosynaptic implementations.

(b) Resulting synaptic weights from neurons A, B and C to neuron M in the heterosynaptic implementations.

Figure 6.17

Figure 6.17(a) shows that in the same way as occurred in the previous experimental implementation, the incoming synapses from the visual afferent neurons A,B and C have been potentiated in motoneuron R, also reaching their maximum synaptic weights. Again, this indicates that red, black and green visual stimuli will elicit a strong response in the motoneuron R.

For the case of motoneuron M, the incoming synapse from the afferent neuron C (Figure 6.17(b)) has been significantly depressed (from 9.0 to 5.12) with respect to the first phase of the experiment. The resulting synaptic weight of 5.12 indicates that green visual stimuli will still eliciting a relatively strong response in neuron M. However, given that the synaptic weight between C and motoneuron R is stronger than the synapse between C an motoneuron M (9.0 vs 5.12), the resulting behaviour tends to favour the activation of R when green visual stimuli is presented.

The resulting synaptic weights in both experimental conditions are reflected in the navigation behaviour of their corresponding implementation of the artificial insect. This is shown below in figure 6.18: In figure 6.18 the black line shows that after 200000 iterations

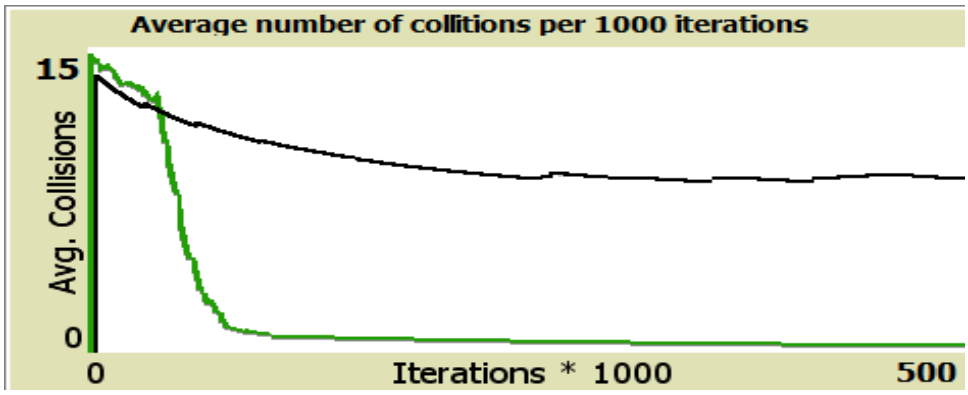


Figure 6.18: Average number of collisions after making green patches noxious.

the non-heterosynaptic system is still colliding permanently with noxious stimuli. On the other hand, the heterosynaptic system represented by the green line stops colliding with noxious stimuli after about 80000 iterations.

In summary, the artificial insect in the heterosynaptic implementation has learnt to associate green visual stimuli with an aversive reflex response. The insect also shows adaptation to the environment by avoiding the green patches most of the time. This behaviour reflects the synaptic changes in the neural circuit where the synapses controlling

the previous acquired association between green visual stimulus and the reward related response have been weakened.

6.6.3 Experiment 3: Unlearning of previous acquired visual-input with aversive response associations in both heterosynaptic and non-heterosynaptic SNN systems.

The first phase of this experiment was carried out in the same way to the first phase of experiment 2. Both implementations of the artificial insect were trained until they learnt to associate black and red patches with an aversive reflex response while green patches were associated with a reward related action. The resulting synaptic weights in the heterosynaptic and non-heterosynaptic experimental conditions were the same as shown before in Figure 6.14.

In the second phase of the experiment, the red patches were attached with reward-related stimuli in the simulation environment as illustrated below in Figure 6.19:

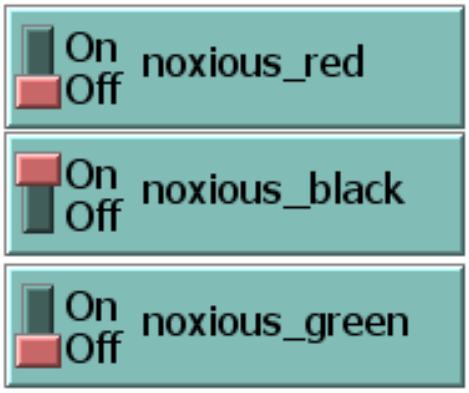


Figure 6.19: The red and green switches set on 'Off' indicate that no pain stimuli will be triggered in case of collision with these colours. On the other hand, the switch for black is set on 'On' indicating that pain stimulus will be triggered on contact with this colour.

After setting the parameters as indicated above, the simulation was resumed in order to observe the behaviour of the artificial insects with the new environment condition (red patches associated with reward-related stimuli).

In the non-heterosynaptic implementation the artificial insect did not approach the red patches during the second phase of the experiment. This behaviour was expected because

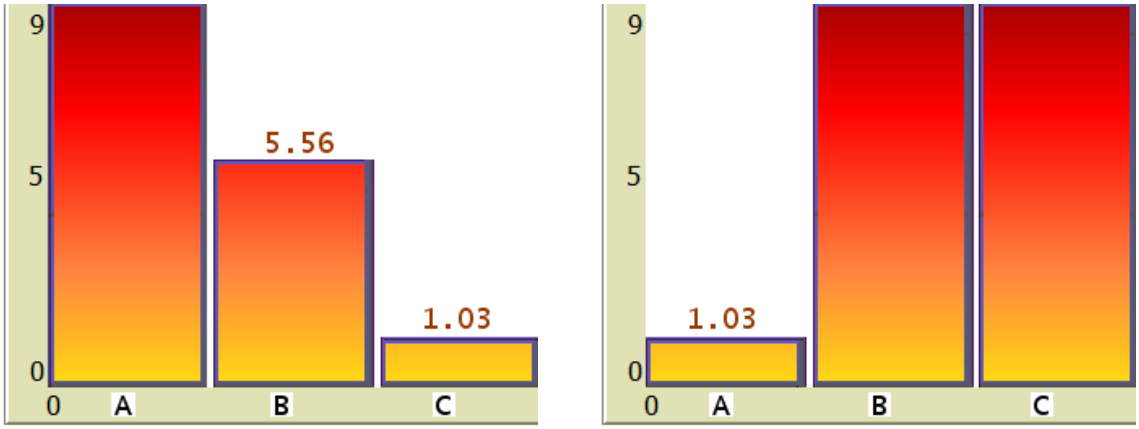
of the conditioning of the red patches with the aversive behaviour acquired during the first phase of the experiment. In order to allow the insect to try approaching the already known noxious patches, a programmatic risk prone behaviour was added in the neural simulation by temporarily neglecting the inputs from the afferent visual neurons in the motoneurons R and M. This allowed the artificial insect to try patches of different colours without eliciting their corresponding conditioned response. The simulation was executed for several thousand iterations without observing any significant change at the end in the synaptic weights of the motoneurons. However, during the simulation it could be observed that immediately after contact with a red patch the synapse from neuron B (visual afferent for red colour) to motoneuron M was slightly potentiated. Nonetheless, after just a few iterations the same synapse was rapidly depressed, reaching their minimum (original) weight as shown in Figure 6.14.

The rapid depression of the incoming synapse from B in motoneuron M can be explained by further incoming pulses (from neuron B) following the activation of motoneuron M yielding to the pre-after-post-timing dynamics of the STDP learning window.

In summary, despite the added risk prone artefact, the system did not show adaptation to the changed environment condition.

In the heterosynaptic implementation, it could be observed that the insect approached the red patches after a time interval without finding food (green patches). This behaviour was expected due to the neural circuit architecture described before in Section 5.2.2. At the beginning of the second phase of this experiment, the learning occurred very slowly. This is because the exploration of the red patches does not happen very frequently due to the initial higher amount of green patches available which keep the artificial insect fed while inhibiting the activation of its risk-prone mechanism. As the amount of green patches decreased (eaten by the insect) the exploration of red patches became more frequent due to the activation of the risk-prone behaviour. It was observed that the insect started looking for red patches on its own, *i.e.* without the activation of the risk-prone-mechanism (neuron EM in Figure 6.4) when the number of green patches fell under 25% of the initial quantity. This behaviour indicates that the insect was able to unlearn the previously associated aversive response to red patches while learning the opposite action (moving towards them). This

was reflected in the resulting synaptic weights of the motoneurons R and M as illustrated below in Figure 6.20:



(a) Resulting synaptic weights from neurons A, B and C to neuron R in the heterosynaptic implementation.

(b) Resulting synaptic weights from neurons A, B and C to neuron M in the heterosynaptic implementation.

Figure 6.20

Figure 6.20(a) shows that the weight of the incoming synapse from the visual afferent neuron B (red) was significantly depressed in motoneuron R (from 9.0 to 5.56). The resulting weight of 5.56 is still relatively high and could possibly evoke an action potential in R if a few pulses arrive shortly enough from each other. However, the activation of motoneuron R through EPSPs from neuron B is unlikely given that the incoming synapse from B in motoneuron M (Figure 6.20(b)) has been strongy potentiated, reaching its maximum value (9.0). Since motoneurons R and M are mutually inhibitory, the neuron with the stronger activation (in this case M) will fire first and prevent the opposite neuron to become activated.

6.7 Summary

This chapter presented three experiments that tested a simulated insect-like agent controlled by two different implementations of a neural controller. The first implementation used a homosynaptic architecture for associative learning while the second implementation used the heterosynaptic modulatory architecture described in this thesis for the same purpose. The first experiment tested the ability of the agent to learn associations between (initially

neutral) visual stimuli with innate reflex responses. Through a trial and error process the insect agent was able to successfully learn the association between visual inputs and reflex responses in both homosynaptic and heterosynaptic implementations. This demonstrated that the proposed system was able to perform associative learning as well as the original homosynaptic implementation.

The second and third experiments tested the ability of the simulated insect-agent to adapt to changing environment conditions which required the unlearn of previously acquired associations between visual stimuli and reflex responses. This time the homosynaptic implementation showed the following limitation: despite being able to recognize new pair-wise associations between stimulus and behaviour, this implementation of the agent failed to change the previously acquired memory (i.e. already reinforced synapses remained unchanged). By contrast, the insect-agent in the heterosynaptic implementation was able to adapt to the changes by avoiding the noxious stimuli that were initially related to rewards (this is indicated in figure 6.18 which shows the number of collisions with noxious stimuli in both implementations).

The results in the second and third experiments demonstrate that the modulation of neural dynamics in the heterosynaptic system allowed the simulated insect-agent to adapt successfully to the changing environment. This verifies the research hypothesis by showing that the proposed system is able to extend the homosynaptic neural model to overcome some of its weaknesses and also that it can be implemented in practical systems.

In conclusion, the proposed heterosynaptic system used as a controller for the artificial insect allowed it to:

1. Explore previously known noxious stimulus under a simulated starvation condition.
2. Unlearn visual to aversive-response conditioning behaviour.
3. Adapt its behaviour according to the environment conditions by learning that something that was noxious in the past becomes the opposite (rewarding) and vice-versa

Chapter 7

Implementing a Heterosynaptic Neural Circuit to Control the Navigation of a Robot.

This chapter validates the proposed system through the implementation of a neural circuit based on visual pattern recognition and associative learning that is able to control the navigation of a physical robot.

In a similar way to the previous experiment described in chapter 6, the system described in the following Chapter is able to associate visual stimuli with reflex responses. However, in contrast with the system shown in the previous Chapter, the resulting motor behaviour is associated with non-linear visual patterns (composed by multiple pixels) instead of single inputs (one pixel colour).

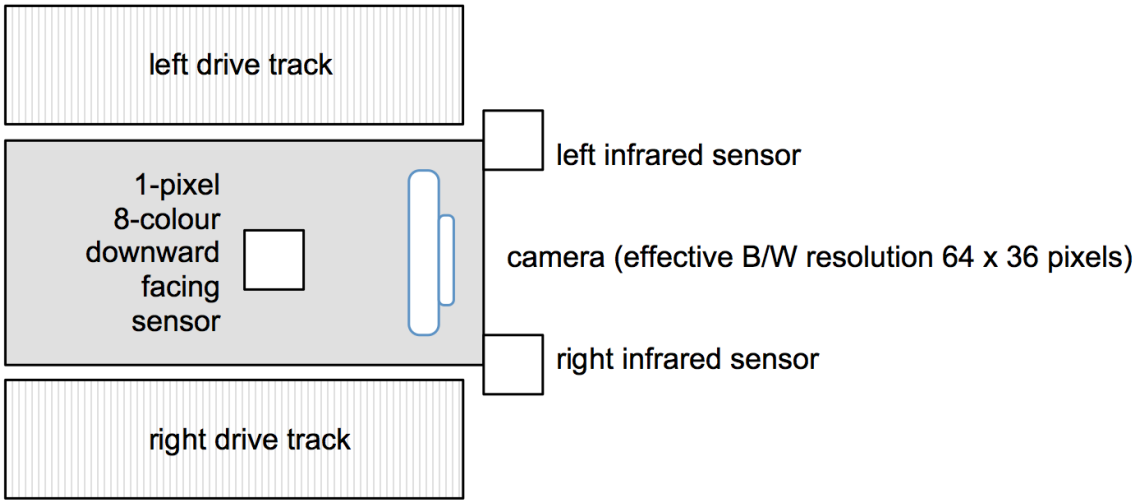


Figure 7.1: Schematic of the robot.

A simplified neural circuit was implemented to control the navigation of the robot in different types of environments allowing it to avoid obstacles while looking for rewarding stimuli. The robot detected collisions with objects (anything that was closer than 10 cms) using two infra-red sensors on the left and right sides at the front of the robot (see Figures 7.1 and 7.11).

The activation of the infra-red sensors fed the "pain" receptors of the neural circuit which consequently activated the entire nociceptive pathway eliciting its corresponding reflex response (rotating). As in the first experiment, the neural circuit also had a reward-related pathway which, when activated, triggered the movement of the robot towards the source of the stimulus.

The reward-related sensor was implemented in the hardware of the robot by using a one-pixel light/colour sensor-camera which, when positioned over a pre-defined colour, activated the reward-related sensor of the neural circuit (see Figure 7.12).

The objective of this experiment was to observe and compare the learning behaviour of a robot under two different experimental conditions, as follows:

1. Implementing the neural circuit that controls the robot based on a traditional threshold fire SNN model.

2. Implementing the neural circuit that controls the robot based on the mechanism of heterosynaptic-modulation for plasticity, as proposed in this thesis.

Based on the observation of the learning behaviour in both experimental conditions, the answers to the questions listed below will verify the research hypothesis mentioned in section 1.3 of chapter 1:

1. To what extent was the robot in both experimental conditions able to navigate autonomously based on (unsupervised) learnt visual information from the environment?

2. Were the runaway dynamics emerging from the embedded STDP learning mechanism overcome by the heterosynaptic modulatory architecture of the system?

In both experimental conditions the robot was expected to learn to identify and avoid visual patterns and landmarks associated with noxious stimuli. The robot was intended to move towards visual landmarks associated with rewarding stimuli. At the beginning of the training phase, the robot was not expected to be aware of which stimuli were to be avoided or pursued. Thus learning occurred through trial and error where the robot learnt to associate collisions with visual information acquired shortly before the collision sensor was triggered. The same learning mechanism was applied in the case of a rewarding stimuli where the triggering of the reward-related sensor was associated with the visual stimuli acquired shortly before.
In order to test the robot controlled by the heterosynaptic SNN system, the following characteristics were measured and compared with the non-heterosynaptic implementation:

1. the association of neutral visual information with a reflex response was measured in terms of the evolving synaptic efficacy between visual afferent neurons and motoneurons.

2. the resulting motion behaviour determined by the robot's ability to predict and avoid collisions with obstacles as well as to approach reward-related stimuli.

During the testing phase the infra-red (collision) sensors and the one-pixel colour camera (reward-related sensor) were disconnected in order to let the robot navigate based solely on visual information.

7.1 Methodology

7.1.1 Preprocessing of the Visual Input

Before feeding the neural circuit with visual information, this information was prepared to enable the neural system to work with it efficiently while avoiding unnecessary processing and greater computational overhead. In order to get a simplified but usable visual input the following steps were performed in the preprocessing phase:

1. Acquiring the image from the camera in a grey scale of 8-bits depth (see Figure 7.2).
2. Applying the Canny algorithm [88] for edge detection and to produce a binarized image (see Figure 7.3).
3. Enhancing the contours of the edges (see Figure 7.3).
4. Resizing the image according to the number of input receptors in the neural circuit (see Figure 7.4).

For the first step, the settings of the video-capture software were adjusted to produce a grey scale image of 256 shades (8 bits) with a resolution of 480 pixels width and 270 pixels height. For the rest of the preprocessing steps a capture interface application was implemented in Microsoft Visual C# using the OpenCV [89] API to perform the Canny filter, contours-enhancement and the downscaling of the target image. The details of the used API-functions with their corresponding parameters are shown in the appendix A. The visual-preprocessing application generated an ASCII file containing the pixel data of the resulting image. This file was then used by the neural engine (implemented in Netlogo) to extract the visual data in order to feed the neural circuit.

Figure 7.2: Unprocessed image from the camera frame-buffer with a resolution of 480 x 270 pixels and 8 bits colour depth..

Figure 7.3: Resulting image after applying the Canny and edge detection filters.

Figure 7.4: Downscaled image with 64 x 36 pixels, ready to be processed by the neural system.

7.1.2 The Neural Circuit

Quads: Photo-receptors, Input Neurons and Classifiers

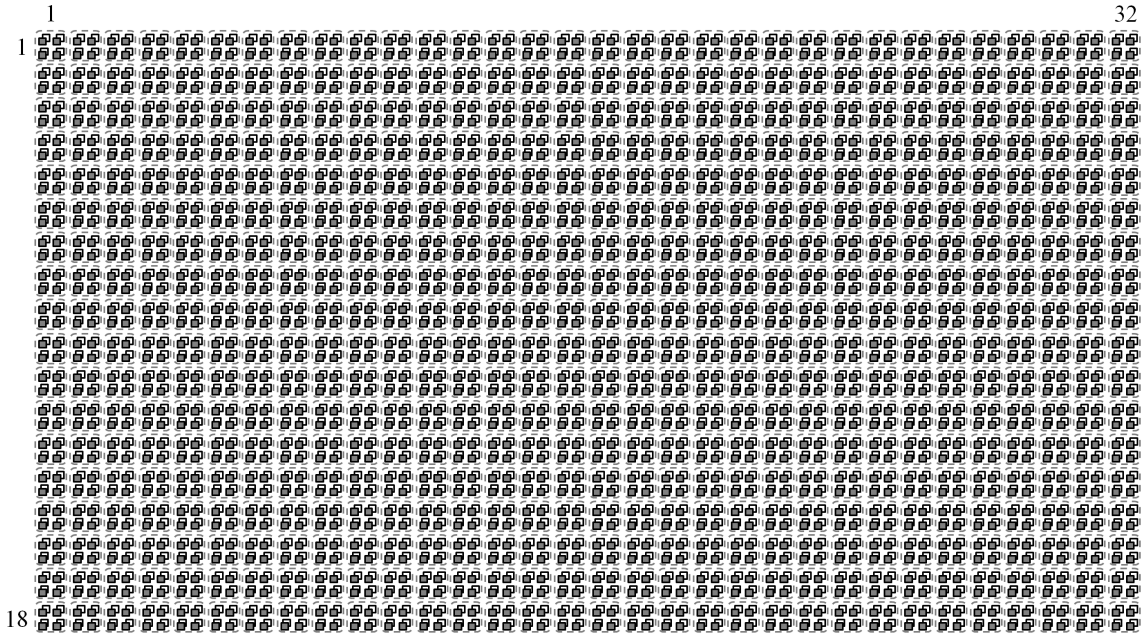


Figure 7.5: Visual input layer with 64 x 36 pairs of black and white receptors.

The visual input layer of the neural circuit is built up of several pairs of black and white (or ON/OFF) receptors grouped into quadruplets (quads) which together form an input matrix as shown in Figure 7.5. The input matrix has a resolution of 64 pairs width and 36 pairs height (or 32x18 quads).

Each black and white receptor-pair detects the absence or presence of light respectively. The light information is provided by the colour (black or white) of the pixel corresponding to that area in the preprocessed image.

Depending on the colour of the sensed pixel only one receptor can be activated within a receptors-pair at the same time. Thus, a receptor for colour black is only activated if the sensed pixel is black and a receptor for white is only activated if the sensed pixel is white. Receptors are used by the SNN engine as an interface to translate numerical information into spikes. For instance, if there is a black pixel in certain position the SNN engine feeds with the number 1 the receptor for black corresponding to the position of the pixel. In the implemented engine, numerical information can be only given as integer values. An integer value n is translated by the receptor into a spike with a firing delay of $n - 1$ where

$n > 0$. Thus, if a value of 1 is sent to a given receptor, this will generate and transmit a spike or pulse immediately (with delay 0).

Each receptor is connected with one afferent neuron which is responsible to forward the pixel information as a spike towards the next layer of the neural circuit. The architecture of the quad group of receptors with their afferent neurons is illustrated in Figure 7.6:

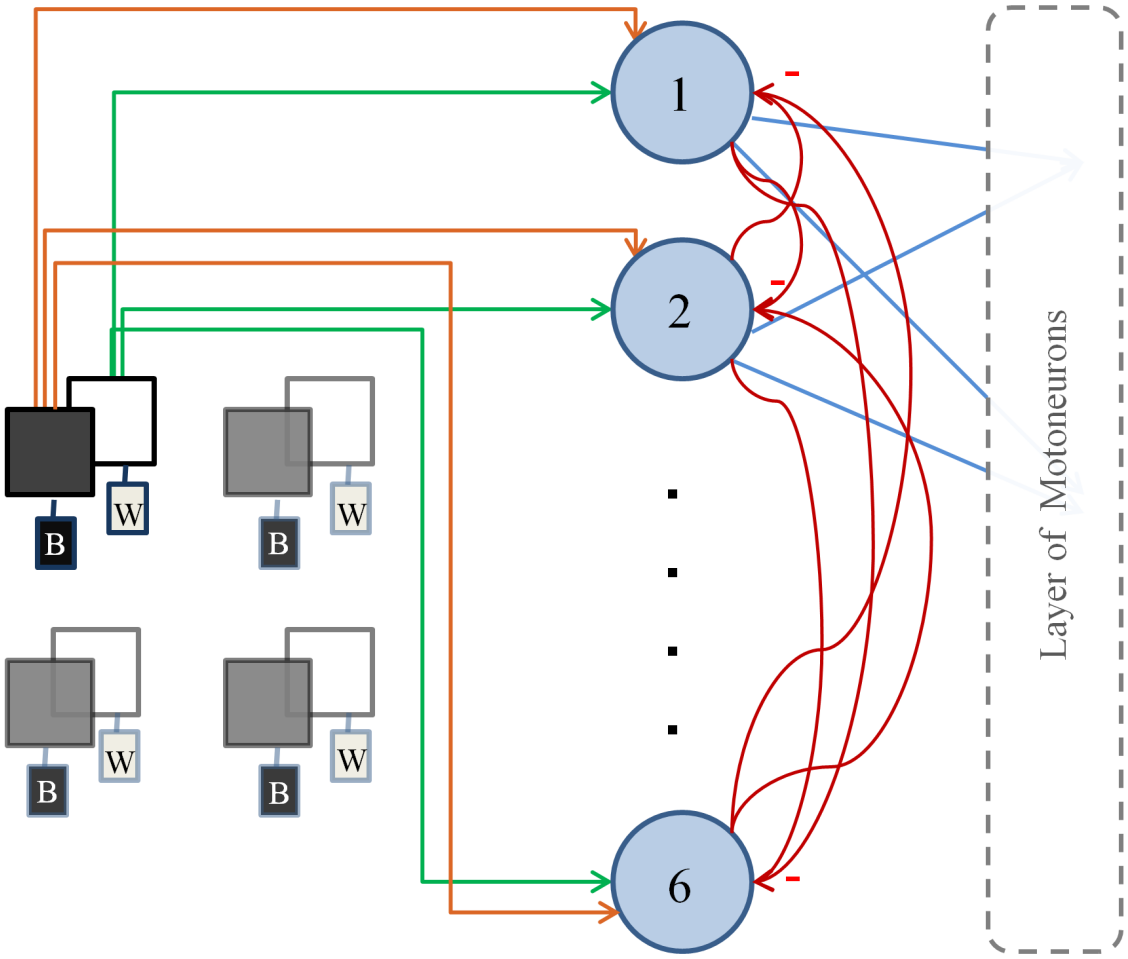


Figure 7.6: Architecture of a quad subcircuit.

In Figure 7.6 the black and white receptors are indicated with the letters B and W respectively. The afferent neurons are represented with black and white squares, each one with an incoming connection from its corresponding receptor.
Each afferent neuron is connected with the six neurons that comprise the next layer. These are represented in Figure 7.6 with blue round shapes enumerated from 1 to 6. For the sake of simplicity Figure 7.6 only shows the outgoing synapses from the top-left afferent pair with the neurons 1, 2 and 6 of the next layer.
The function of the six-neurons layer is to perform the classification of primitive shapes based on the four black and white pairs arrangement. As shown in Figure 7.6 there are mutually inhibitory synapses between the six classifier neurons. This serves the purpose of only one primitive shape being reported at a time by a group of quads to the following layer of motoneurons.
The type of shapes that each classifier detects can be set up in advance by either:

1. manually hard-wiring the synapses and weights between afferent and classifier neurons.

2. training the quad layers with the desired input-pattern to output-neuron mapping.

The first option is easier to implement and does not require the extra training phase. However, it is less flexible for experimental purposes since every change in the type and number of shapes would require a new adjustment of connections and their parameters.
The second option which is the one used in this work, requires the preparation of a short dataset and the training of the quad layers in order to get the appropriate synaptic weights between the afferent (black and white) neurons and the classifiers. This approach is not only more sophisticated but also offers more flexibility when trying different shapes and increasing or reducing the number of classifiers.

Figure 7.7 below illustrates the primitive shapes associated to each classifier neuron. These 6 shapes were selected from the 16 possible patterns in a quad layout. The reason for selecting this combination of 6 black and white patterns is because during the experiments this combination shown the best results in terms of classification and pattern recognition of the visual inputs.

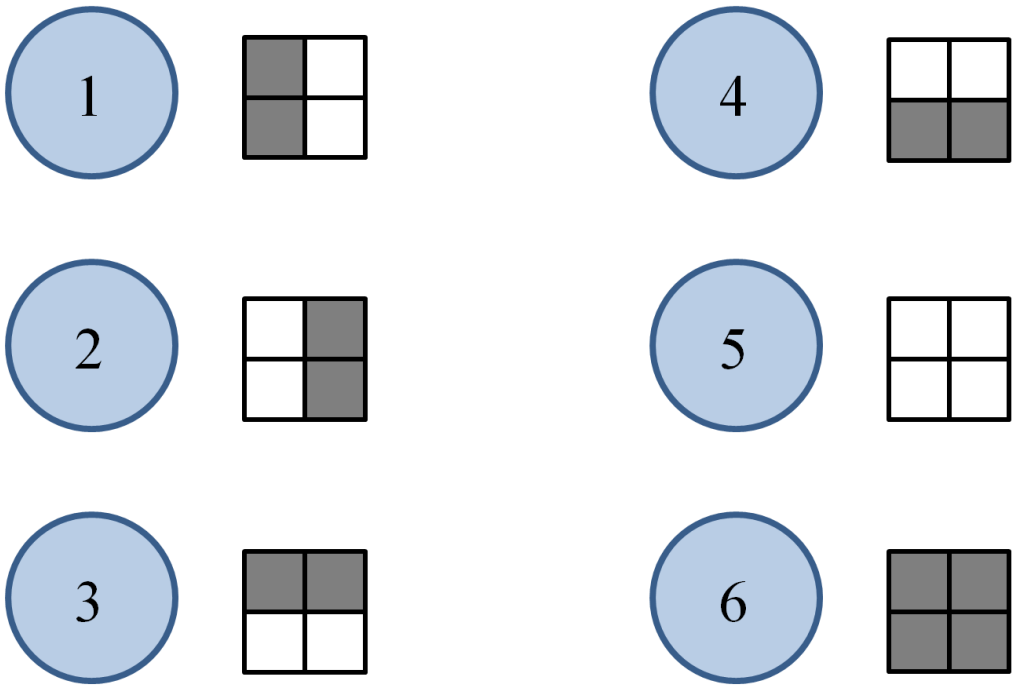
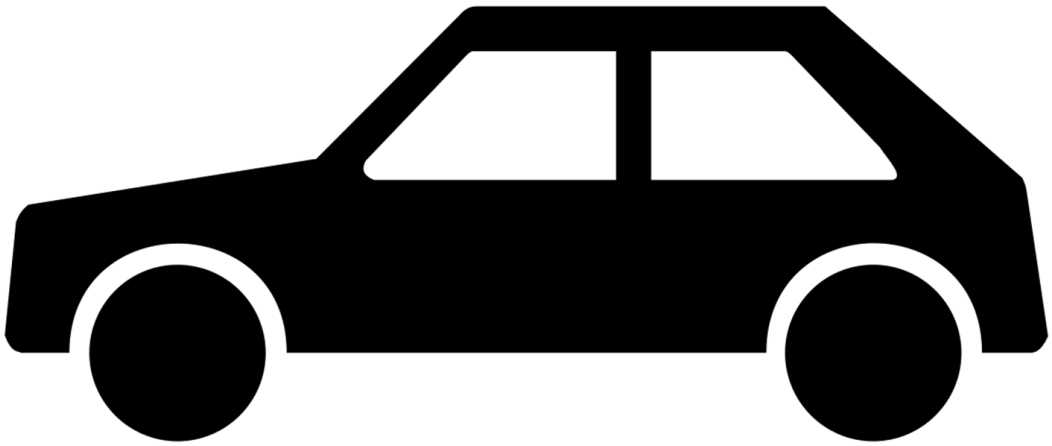


Figure 7.7: Classifier neurons with their corresponding primitive-shapes.

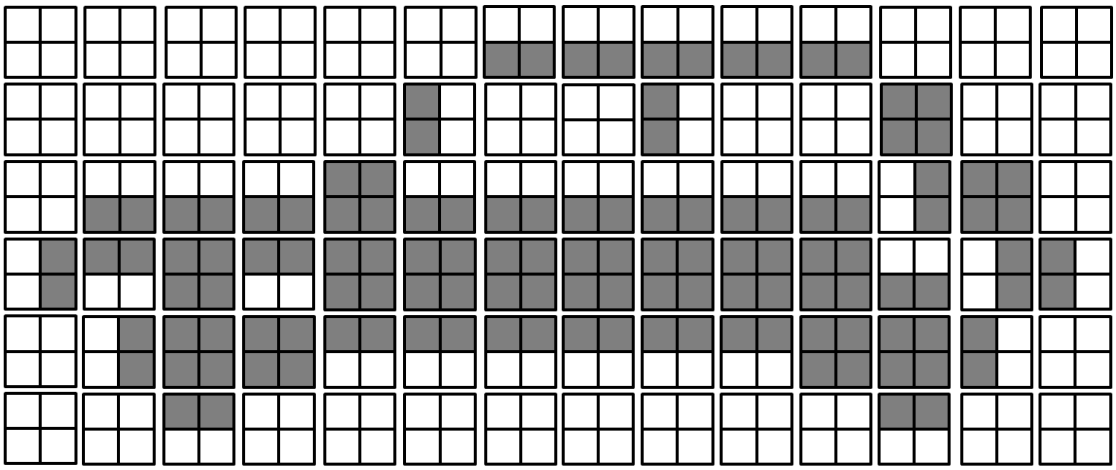
Given their combination of black and white pixels, the primitives classified by neurons 1 to 4 are more prone to detect the presence of edges in the image. On the other hand, the primitives classified by neurons 5 and 6 are mostly related to areas where the pixels colour is homogeneous, *i.e.* indicating the absence of detectable features. For this reason, in order to reduce the computational overhead in the neural circuit and to avoid the transmission of spikes representing empty spaces, the neurons 5 and 6 do not have any outgoing synapses. Nevertheless, despite having no connections with further layers, these neurons (5 and 6) work as filters avoiding the wrong activation of the other classifier-neurons (1 to 4) in homogeneous areas (all pixels black or white).

Figures 7.8(a),(b) and (c) illustrate with an example the visual processing within the layers of the quad groups:



(a)

(b)

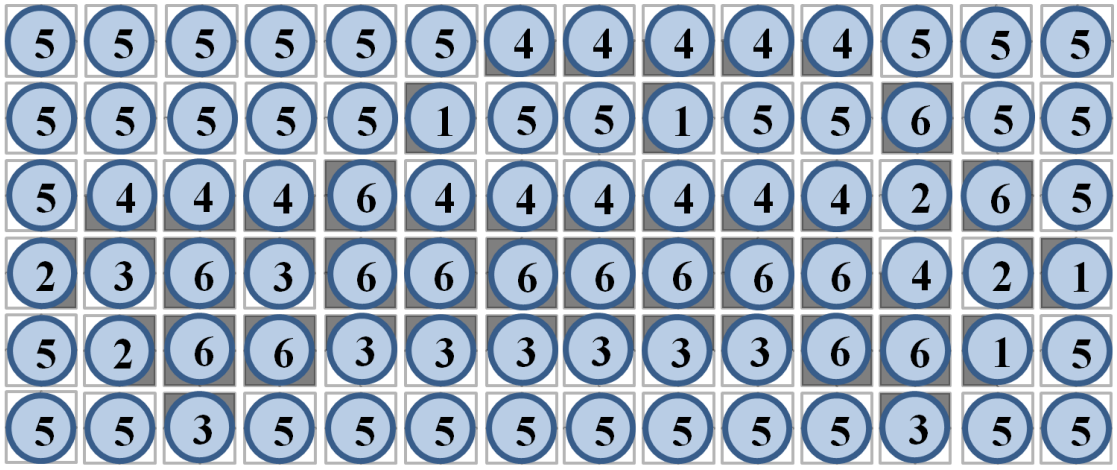


(c)

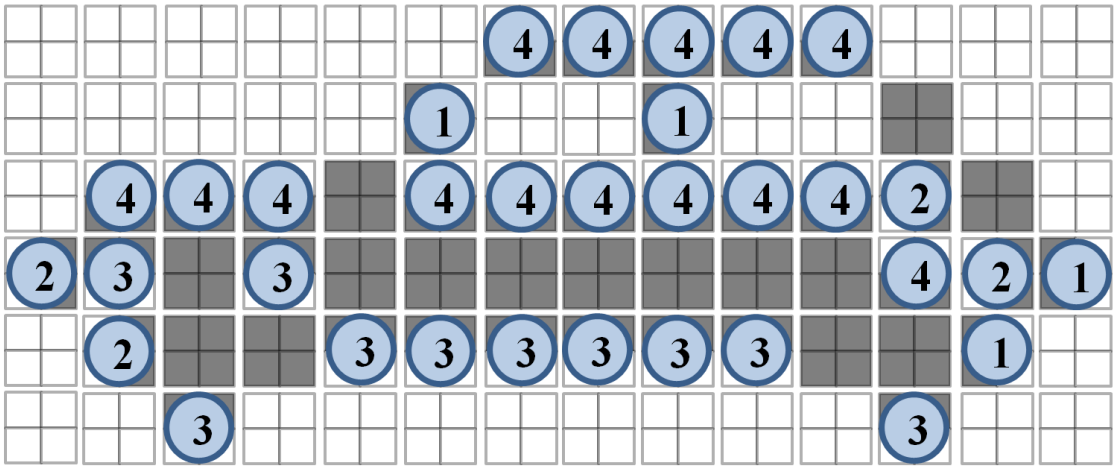
Figure 7.8

Figure 7.8(a) shows the binarized shape of a car. Figure 7.8(b) shows the activation of the black and white receptors in the area where the shape in (a) is captured. The activation of the black and white receptors is indicated with green and yellow squares respectively. The absence of activity or non-activation in the black and white receptors is illustrated with grey and white squares respectively. Figure 7.8(c) illustrates the detected primitives based on the patterns of pixels presented to each quad.

Below, Figure 7.9(a) shows the activated classifier-neurons resulting from the detected primitives in the group of quads. Figure 7.9(b) shows the classifier-neurons that will report their activation (*i.e.* transmit spikes) to the layer of motoneurons. Note that neurons 5 and 6 have been filtered out.



(a)



(b)

Figure 7.9

Sensory - Motor Architecture

The acquired visual information is propagated by each quad to the layer of motoneurons. The propagation occurs through the spikes transmitted by the activated primitive-classifiers (neurons labelled 1 to 4, see Figure 7.10). It is worth stressing, that because of the mutually inhibitory synapses in the classifiers layer, there can be only one classifier neuron activated at the time in each group of quads.

Given that there are 576 (32 x 18) quads and each quad group has 4 propagating classifier-neurons, the number of visual-input synapses reaching each motoneuron is 2304. However, as mentioned before, since each quad transmits one spike at the time, the maximum theoretical peak is 576 incoming spikes at the same time in a motoneuron. This number is an important indicator in order to establish the appropriate delimiter range (minimum and maximum values) of the synaptic weights between the quads and motoneuron layers. This is because the resulting conditioned behaviour on each motoneuron depends on the combinatorial patterns of activation of the classifier layers. Thus, if the maximum allowed synaptic-weight value is set too high it will make the motoneurons more prone to be activated by a small (dominant) subset of visual inputs and in a shorter period of time, reducing their combinatorial capabilities. Thus, it is more convenient to use small weights delimiters in order to give the motoneurons the time to integrate as much information as required before triggering their activation. The value of the synaptic weight delimiters as well as other neural parameters are described in Table 7.1.

Motoneurons with Plasticity Modulation

The weights (or synapse efficacy) of the synapses between the quads-classifiers and motoneurons R and M are equally initialized with the lowest delimiter synaptic value (See Table 7.1). This is to ensure that at the beginning of the experiment the visual inputs will not have any effect on the motor behaviour (since visual information is meant to be a neutral stimulus) and also, to guarantee that the initial conditions will be the same for each one of the synapses leaving the quads-classifiers. These synapses are depicted in Figure 7.10 below with blue lines going from the classifier neurons (1, 2...) to motoneurons R and M:

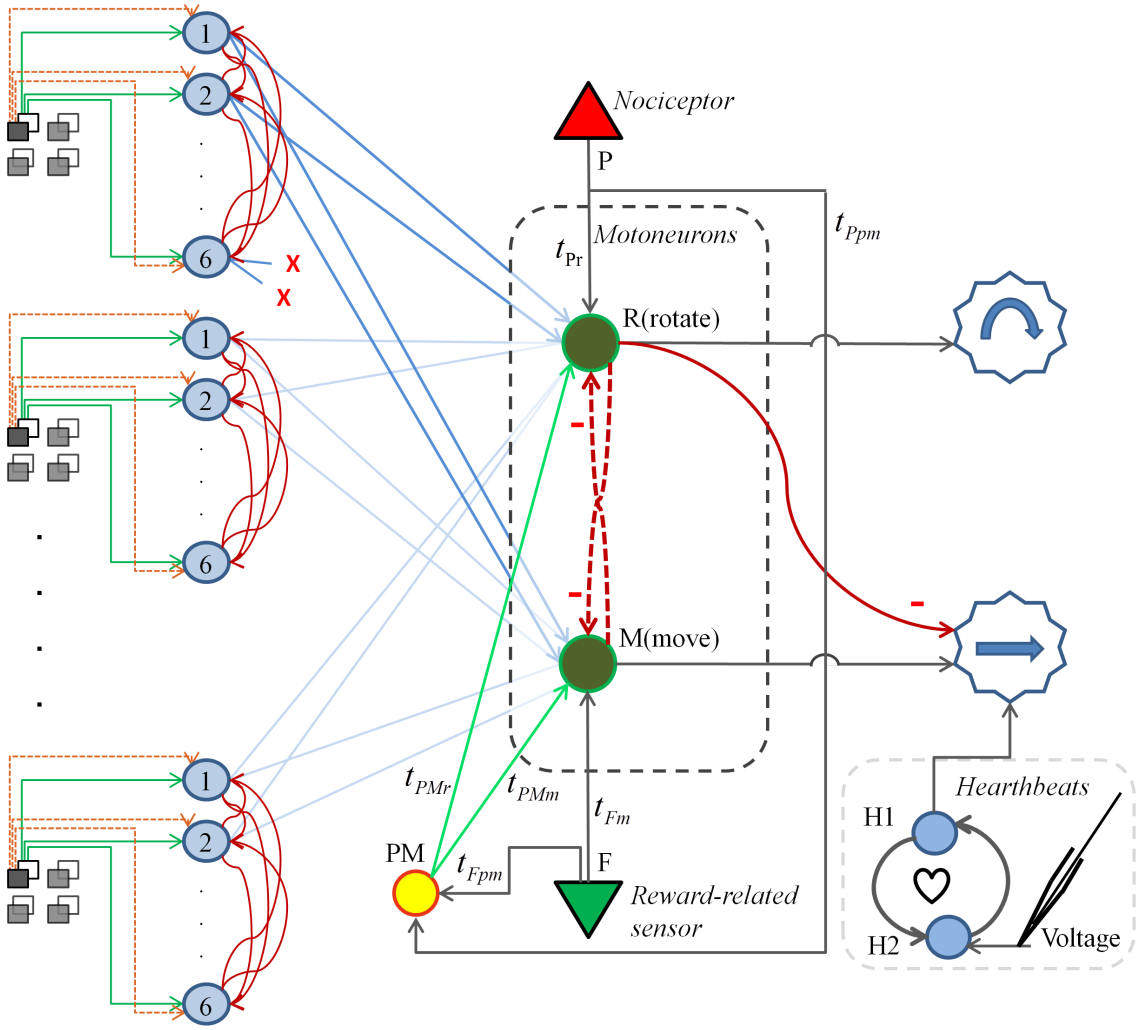


Figure 7.10: The architecture of the neural circuit shows the interaction between the different components of the sensory-motor system, including the quads-layer, motoneurons R and M, the nociceptive and reward-related pathway (receptors P and F respectively), the plasticity modulatory neuron PM, the heartbeats system and actuators to execute the motor behaviour.

The blue synapses from the quads to the motoneurons implement the modulated plasticity mechanism described in Chapter 4. Thus, the synaptic change in these synapses is highly dependent on the PMS concentration in the target neurons (R and M). Given that both motoneurons R and M are also innervated by the modulatory synapses originated in neuron PM, the synaptic change in the visual-motor (blue) synapses is highly dependent on the activity of the plasticity modulatory neuron PM. As seen in Table 7.1 the equilibrium concentration of the PMS in R and M is set with a very low plasticity factor (0.001) which means that in the absence of any modulating activity the plasticity in these synapses is almost neglected. As shown in Figure 7.10 the modulatory neuron PM receives two synapses from receptors P and F. Both synapses are strongly excitatory (as indicated in Table 7.1) this in order to trigger the firing of neuron PM (*i.e.* releasing its modulatory signal) whenever either one of the receptors is activated by its corresponding input stimulus. The outgoing modulatory synapses from neuron PM depicted with green lines in Figure 7.10, have a strongly increasing effect on the PMS of the target neurons (a factor of 1.0 as shown in Table 7.1). The increased plasticity is sustained for a certain interval until it begins to decay rapidly towards the PMS equilibrium concentration.

As described in the previous experiment in chapter 6, in order for the elicited plasticity to be effective in both sides of the STDP learning-window (pre-before-post and post-before-pre timings), it is necessary that the modulatory signal reaches the target neurons R and M before they become activated by either one of the receptors P or F. As explained before, this is because in a PD neuron (in this case R and M) the processing of the spikes with pre-before-post timing will occur during the initiation of the action potential in the motoneurons. Otherwise, if the modulatory signal arrives just after the activation of the postsynaptic neuron, then all the previous incoming spikes from synapses with PMS affinity preceding the modulatory signal and the postsynaptic action potential are processed with a lower PMS concentration compared to the spikes that arrive after the modulatory signal.

Again, to satisfy the timing constraints the synaptic (axonal) delays are adjusted in such a way that the modulatory signal from PM (delays t_{PMr} and t_{PMm}) reaches the target motoneurons before the spike potentials transmitted by the activated receptors P (delays t_{Pr} and t_{Ppm}) and F (delays t_{Fm} and t_{Fpm}) trigger the activation in R and M.

	Neurons		
	R	M	PM
Synaptic weight from Receptor P	20.0	0.0	20.0
Synaptic delay from Receptor P	20.0	0.0	20.0
Synaptic weight from Receptor F	0.0	20.0	20.0
Synaptic delay from Receptor F	20.0	0.0	20.0
Equilibrium Concentration of PMS	0.001	0.001	N.A
Initial Synaptic weight from Classifiers	0.01	0.01	N.A
Synaptic weight from Modulating Neuron PM	1.0	1.0	N.A
Minimum synaptic weight from PD Neurons	0.01	0.01	N.A
Maximum synaptic weight from PD Neurons	0.4	0.4	N.A

Table 7.1: Experimental parameters for motoneurons R and M, and the plasticty modulatory neuron PM

7.1.3 Implementation in Hardware

The simulation environment (Netlogo) running the SNN engine was interfaced to Lego Mindstorms EV3 robotic platform [90] which served as embodiment for the simulated neural circuit described above (see Figure 7.10). The robot visual system was implemented using the rear camera of a Samsung Galaxy Note 3 smartphone (see Figure 7.11). The Samsung video output was captured in real time using the IP-Camera video driver [91] and the OpenCV API [89]. As mentioned before in this chapter the preprocessing (edge detection, binarizing and downscaling) of the captured video frames was done using OpenCV in a separated piece of software written in Visual C#. Each preprocessed frame was sent to the SNN engine as an ASCII file with a frequency of about 15 frames per second.

Figure 7.11: Experimental robot built with Lego-Mindstorms (EV3) and smartphone camera as visual system.

The reward-related sensor was implemented using the EV3 colour sensor camera (Figure 7.13(b)). This sensor is able to distinguish 8 different colours and has one pixel resolution. The sensor was positioned in front of the robot looking downwards (Figure 7.12). The colour captured by the sensor was converted into a number and sent to the simulation environment. In the experimental setup the colour selected to represent a rewarding input stimulus was green, thus every time the colour-camera sensed a green surface on the floor, the reward-related receptor in the neural circuit was activated by the simulation environment.

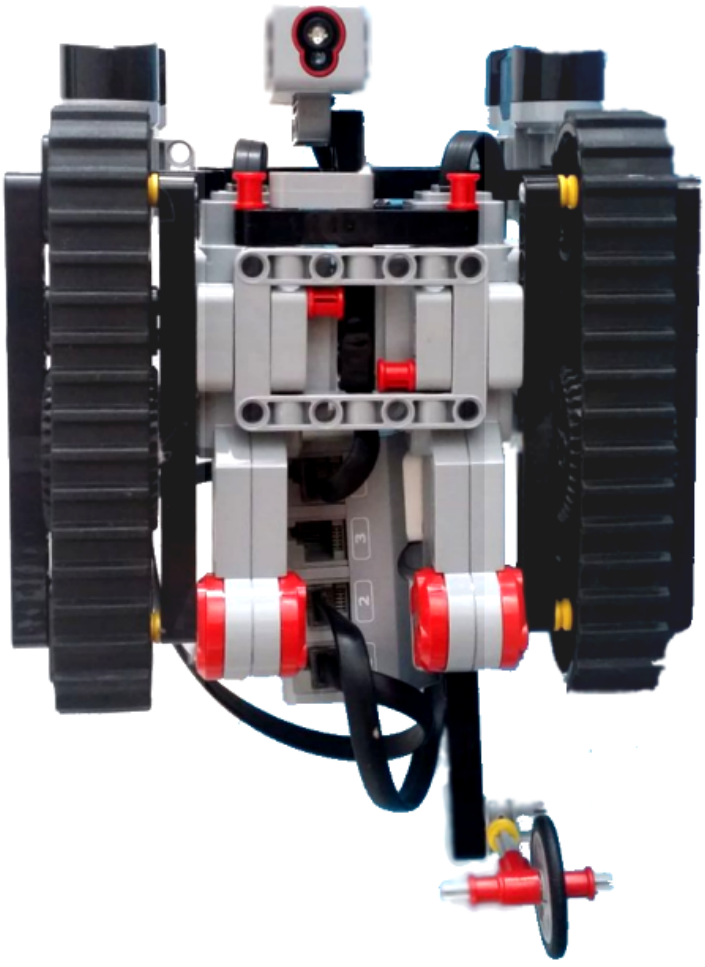
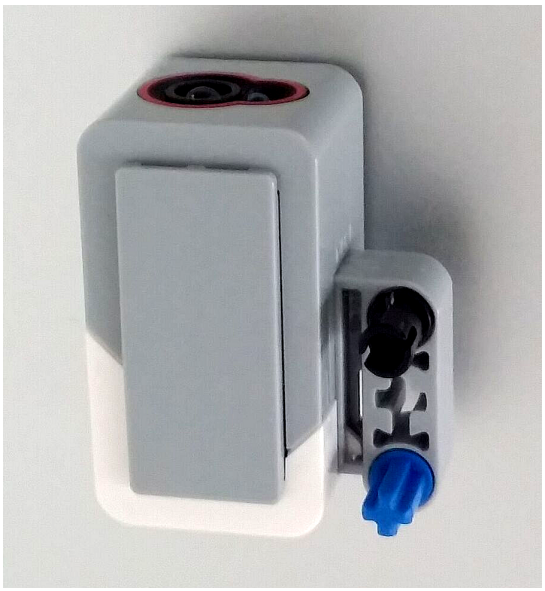


Figure 7.12: Bottom view of the experimental robot.

The nociceptive input was simulated using 2 EV3 Infrared sensors (Figure 7.13(a)) which were positioned on the right and left sides in front of the robot (Figure 7.11). Each sensor reported the distance (expressed as centimetres (cm) with 1 cm resolution) to objects located in front of it. The distance information was sent as a numerical value to the SNN engine. Any value lower than 10 cms was considered to be a collision by the SNN engine which consequently activated the nociceptor in the neural circuit.

(a) Infrared sensor (from the EV3 platform used to detect proximity and collisions with objects in front of it.



(b) Colour/light sensor (from EV3 platform used to detect reward related stimuli during the experiments.

Figure 7.13

The movement and rotation of the robot was controlled by 2 servo motors. When the motoneuron M (move forward) fired, the simulation environment sent a command to the EV3 platform requesting the activation of both servo motors using the same direction and power level for 500 milliseconds, hence moving forward. When the motoneuron R fired, the simulation sent a command to the EV3 platform requesting the activation of both servo motors rotating in opposite direction and same power level, resulting in a spinning of the robot of about 10 degrees.

The robot environment

The training of the robot was performed in an area of about 110×120 cms as shown in Figures 7.14. Rectangular boxes were used to create obstacles and landmarks in the experimental environment. Some areas of the floor were painted with green circles to provide the robot with reward stimuli as illustrated in Figure 7.14 with the green areas depicted with a smiley face.

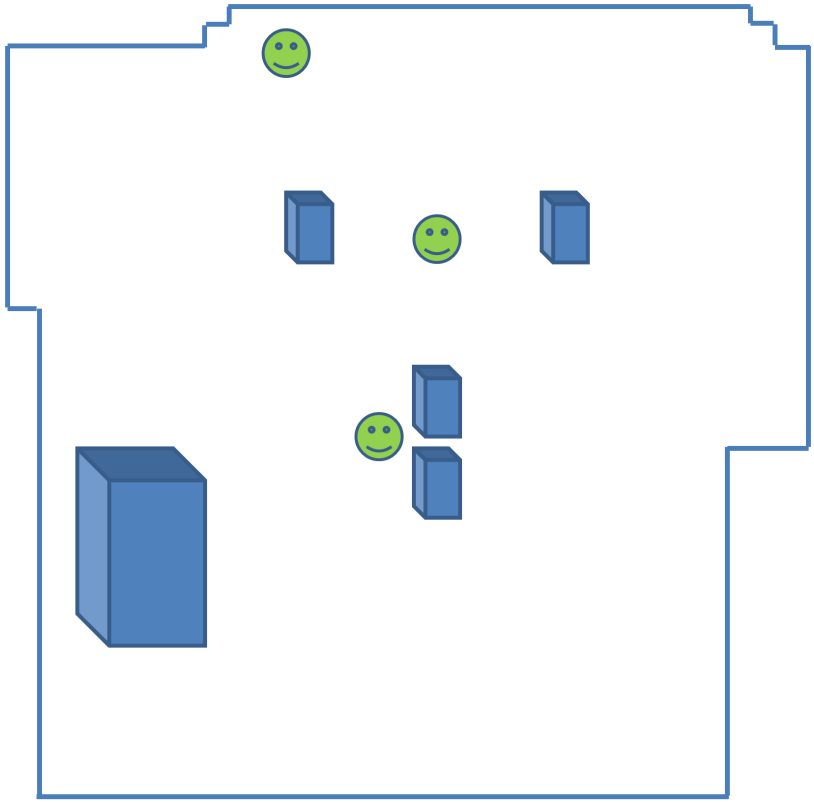


Figure 7.14: Robot test area with obstacles (blue blocks) and reward stimuli (green smiley faces) placed on the floor.

7.2 Results

This Section describes the observed behaviour of both experimental conditions: heterosynaptic and non-heterosynaptic implementations of the robot.

7.2.1 Observed behaviour during the learning phase in both experimental conditions.

At the beginning of the experiment the robot moved through the test environment driven by the pacemaker sub-circuit and the reflex responses triggered by the collision infra-red sensors and the one-pixel colour camera positioned at the bottom. Given that the SNN engine was triggering the collision event in the neural circuit when obstacles were detected at a proximity of 10 cms or less, real collisions with objects were avoided keeping the robot moving smoothly and without the need of human intervention to resolve situations where the robot was stuck.

As illustrated in Figure 7.15, in both experimental conditions, during the first 30 to 50 thousand iterations of the simulation the navigation of the robot was very close (on the edge of the collision range) to the delimitation areas of the test environment and the obstacles placed within the test area.

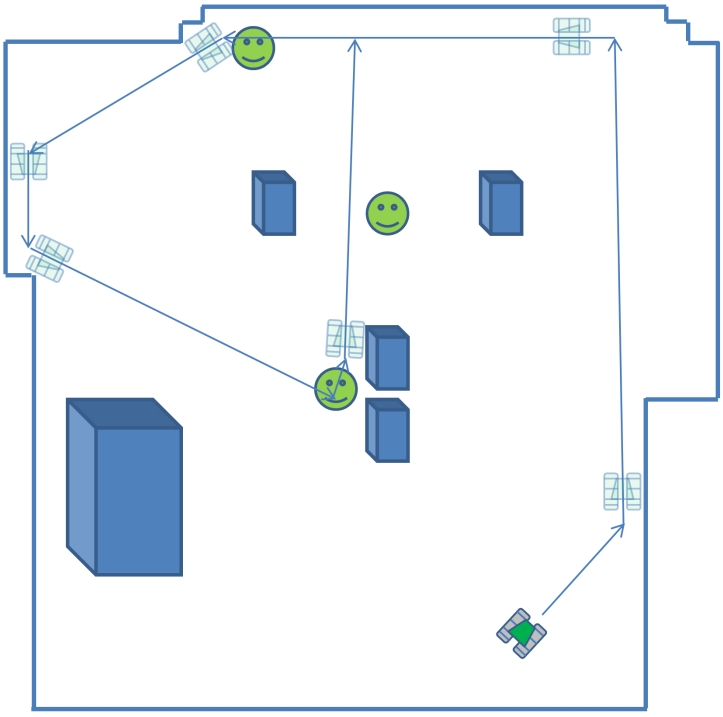


Figure 7.15: Trajectories of the robot during its learning phase.

After about 50 thousand iterations the robot in both experimental conditions started to show obstacle-avoidance behaviour keeping in some cases distances larger than 10 cms from the objects to be avoided. This indicates that the navigation was starting to be

driven by visual information instead of the collision sensors. Despite the increasingly larger distances at which obstacles were detected and avoided, the infra-red sensors were deactivated only after 100 thousand collision-free iterations. This was done in order to ensure that the robot could rely completely on the visual based navigation. At 100 thousand collision-free iterations, the pattern of navigation in both experimental conditions showed some differences mostly regarded to the distance of object avoidance and the ability to detect reward-related landmarks.

Observed navigation in the non-heterosynaptic implementation

In the non-modulated plasticity implementation at 100 thousand iterations, the avoidance distance was between 15 to 30 cms with a tendency to increase as the experiment continued. Movement towards the reward-related areas was also observed, with the robot showing a slight increase of speed when these areas were in the visual field. However, as the robot continued moving it was observed (see Figure 7.16) that while the aversive behaviour towards obstacles landmarks was potentiated the opposite happened with the pursue towards reward-related landmarks where, after approximately 130 thousand iterations, the robot was not showing any approach behaviour towards the sites where the reward inputs (green marks on the floor) were located. This indicated that the continuously potentiated aversive response increasingly overrode the association between visual stimuli and the reward-pursuing action in the robot.

In order to observe the long term behaviour of the robot, it was left running for 250 thousand iterations (ticks in the SNN simulation engine) taking approximately 2 hours to complete. Figure 7.17 illustrates the behaviour observed in the long term test. It shows that the avoidance distances increased to a point (between 30 to 50cms) where the robot was able to move only a few centimeters before turning around creating a loop pattern of movement.

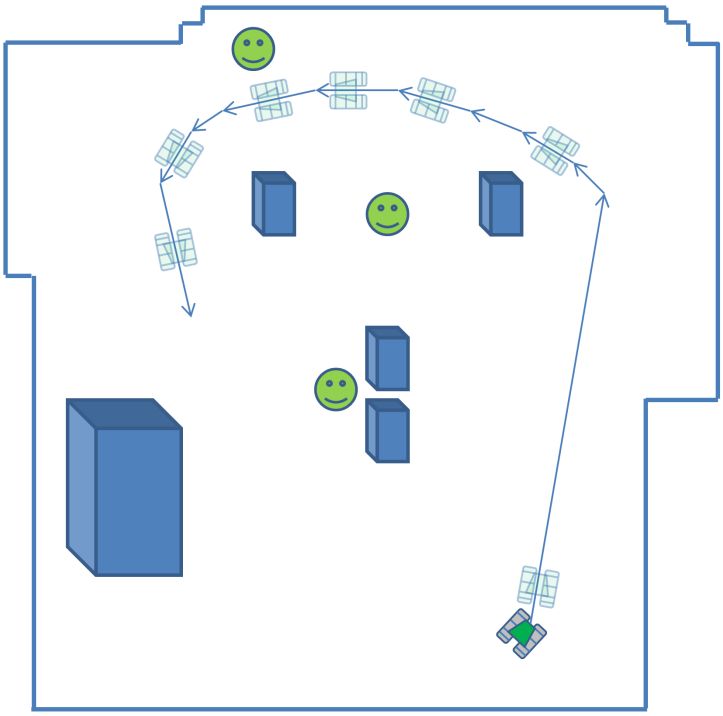


Figure 7.16: Trajectories of the robot after 100 thousand collision free iterations in the non-heterosynaptic experimental condition.

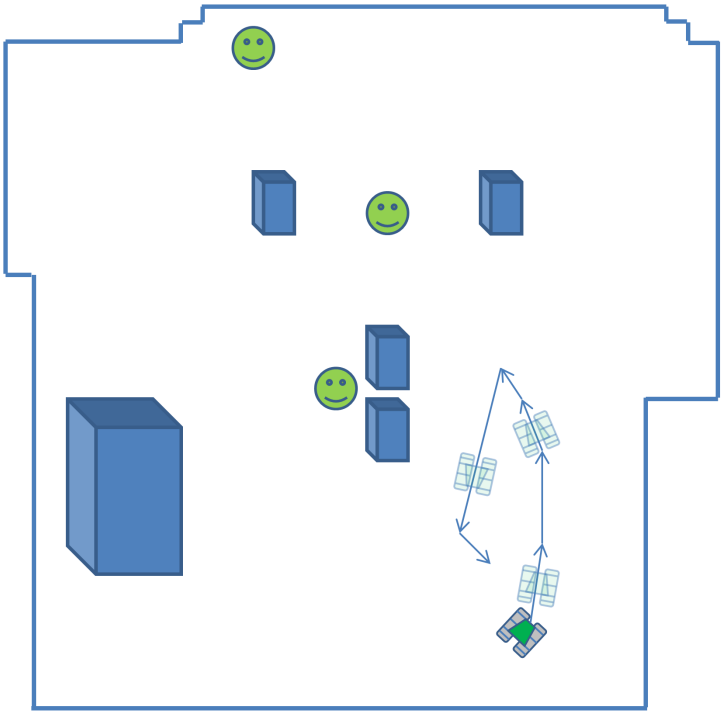


Figure 7.17: Trajectories of the robot after 250 thousand collision free iterations in the non-heterosynaptic experimental condition.

Resulting synaptic weights in the non-heterosynaptic implementation

The motion behaviour described above can be better understood by looking at the resulting synaptic weights between the classifier neurons in the quad-groups and the motoneurons R and M. Given that there are 2304 incoming synapses on each motoneuron, for the sake of simplicity only a section of the visual matrix will be described as illustrated in Figure 7.18.

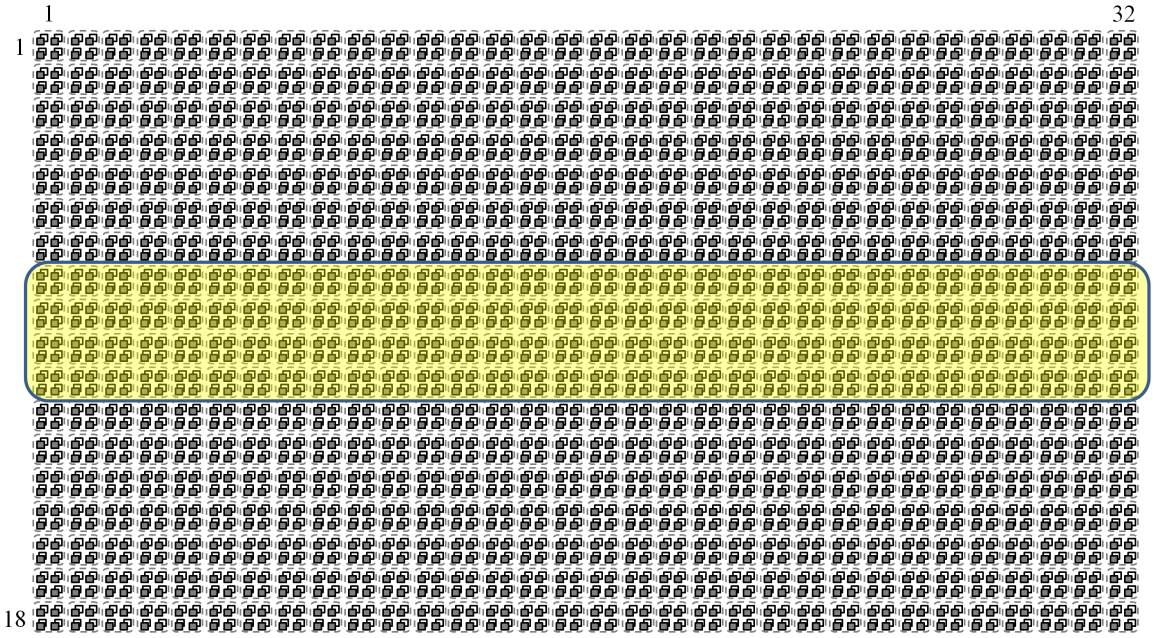
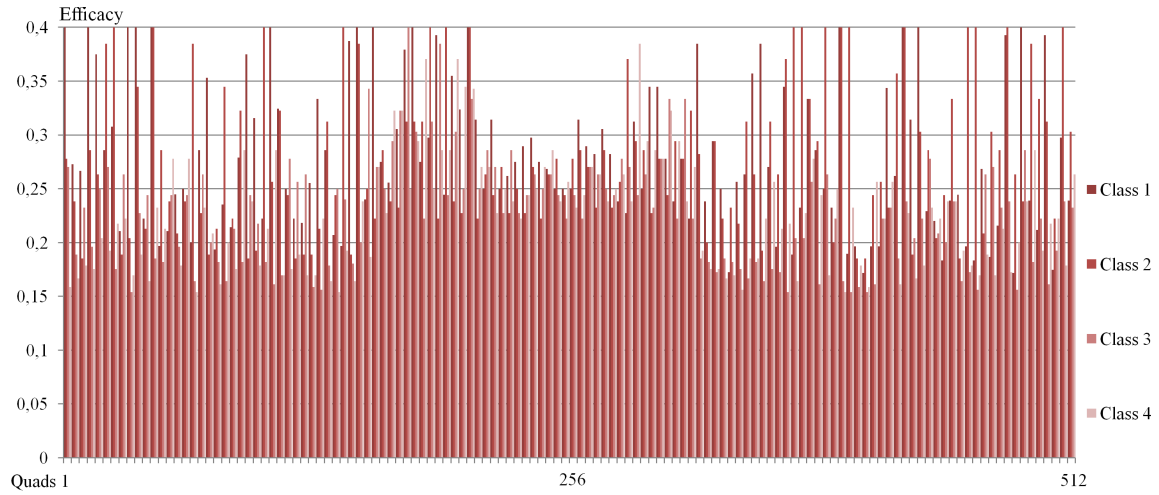


Figure 7.18: The yellow highlighted area in Figure 7.18 indicates the set of Quads with the analysed outgoing synapses.

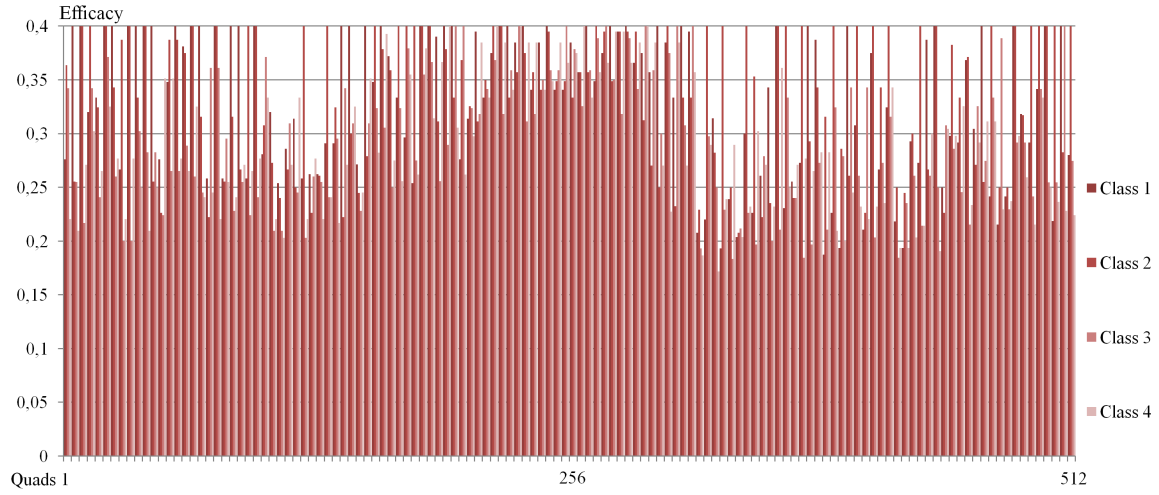
The 4 rows with 32 quads each, highlighted in Figure 7.18 have 1024 outgoing synapses (128 quads x 4 classifiers x 2 synapses) with 512 of these synapses ending in each of the motoneurons R and M. This area of the visual matrix has been chosen taking into account its high spiking activity (compared to the rest of the matrix) observed during the experiments.

Figure 7.19(a) show the weights of the 512 synapses with motoneuron R after running the experiment for 100 thousand collision-free iterations. Each bar shows the weight-value that represents the strength at which the corresponding primitive classifier is associated with the activation of motoneuron R. For instance, looking at the leftmost bar in Figure 7.19(a) it indicates that in the first quad (the top leftmost of the highlighted rows) the classifier neuron which have the strongest association with motoneuron R is the one corresponding

to the visual-primitive class 1 (see Figure 7.7).



(a)



(b)

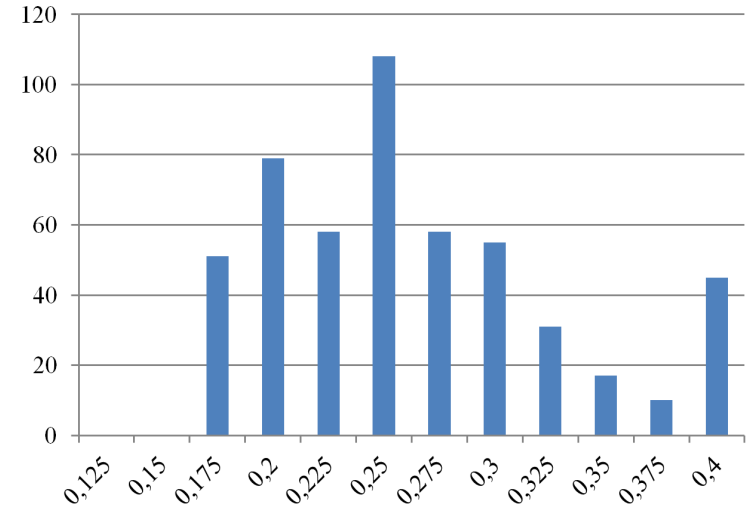
Figure 7.19

The synaptic weights shown in Figure 7.19(a) reached a minimum and maximum value of 0.15 and 0.4 respectively. The average weight value was 0.25 with a median of 0.24 and a standard deviation of 0.06.

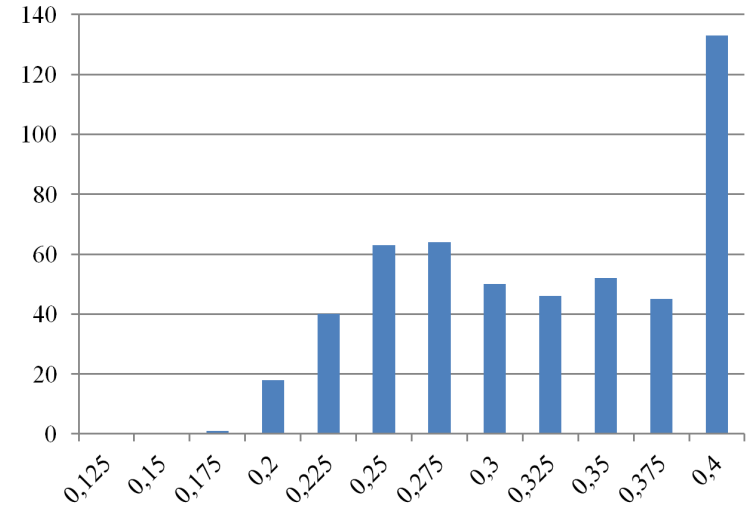
Figure 7.19(b) illustrates the same synaptic weights after 250 thousand iterations. It can be observed that most of the synaptic weights have increased with many of them reaching the ceiling of 0.4. The minimum measured value of a synaptic weight at this stage was 0.17 with an average value of 0.31, a median of 0.31 and a standard deviation of 0.06.

In order to more clearly visualize the change in behaviour of the synaptic values shown in

Figures 7.19 a and b The following two histograms show the number of synaptic weights grouped within various ranges of synaptic strength.



(a)



(b)

Figure 7.20

This way of plotting the data shows that once the SNN simulation has reached 250 thousand iterations, the distribution of synaptic strength is skewed towards higher values that are closer to the ceiling of 0.4. This skewed distribution of the synaptic weights in the motoneuron R explains to some extent the motion behaviour described previously in the non-heterosynaptic condition and illustrated in Figure 7.17. Having increasingly higher synaptic weights in the entire synaptic population reduces the combinatorial capacity of the neuron. For instance it can be seen that the number of neurons required to fire at the

same time in order to elicit an action potential decreases over time, as more iterations are run. The average synaptic weight after 100 thousand iterations is approximately 0.25. Therefore the number of neurons required to fire at the same time in order to elicit an action potential in the motoneuron R, assuming that its membrane potential is currently at rest (65mv) and the firing threshold is (50mv), can be calculated as $(65 - 50)/0.25 = 60$ neurons.

On the other hand, the average synaptic weight after 250 thousand iterations is approximately 0.31; therefore the number of neurons required to fire at the same time in order to elicit an action potential in the motoneuron R, assuming again that its membrane potential is currently at rest (65mv) and the firing threshold is (50mv), can be calculated as $(65 - 50)/0.31 = 48$ neurons.

As seen so far, the robot in the non-heterosynaptic implementation has been able to associate visual landmarks with motor reflex responses. However, the SNN plasticity-system is not able to stabilize or consolidate the already successfully established visual-motor associations. This affects the combinatorial capacity of the neural system over time due to the following:

- The learning dimensional space of the individual synapses is reduced since their weight range becomes smaller as their lower weight delimiter (floor) increases in all synapses.

- At population level, the existence of strongly potentiated synapses reduces the number of presynaptic neurons that are able to contribute to the activation of the postsynaptic neurons. The consequent premature activation, triggered by a small subset of strongly potentiated synapses, results in a reduction of the overall integrative capacity of the postsynaptic neurons.

Observed navigation in the heterosynaptic plasticity-modulated implementation

In the modulated plasticity implementation, after 100 thousand iterations, the observed avoidance distance was between 12 to 25 cms. Movement towards the reward-related areas was also noticed, with the robot showing a significant increase of speed (about twice the moving forward speed) when these areas were in the visual field.

As in the previous experimental condition, in order to observe the long term behaviour of the robot, it was left running for 250 thousand iterations taking again about 2 hours to complete. Figure 7.21 illustrates the behaviour observed in the long term test:

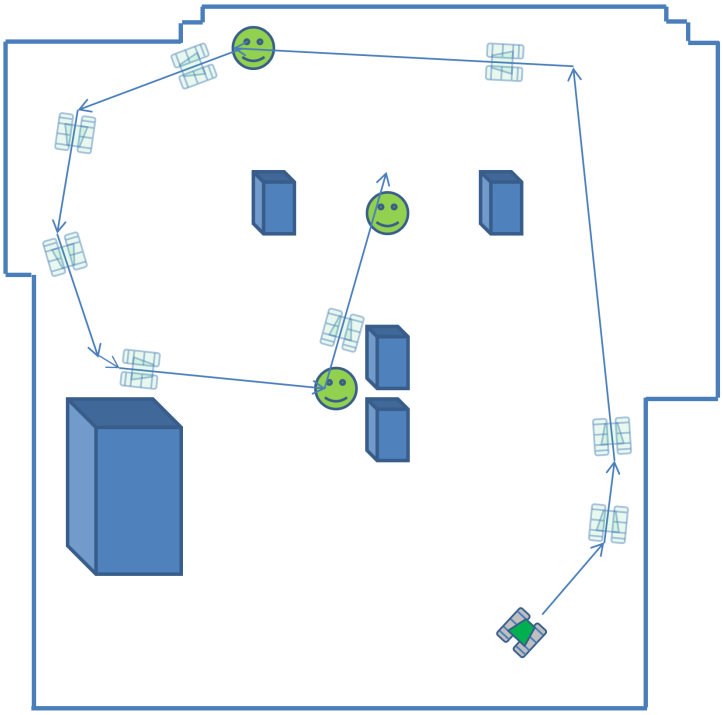
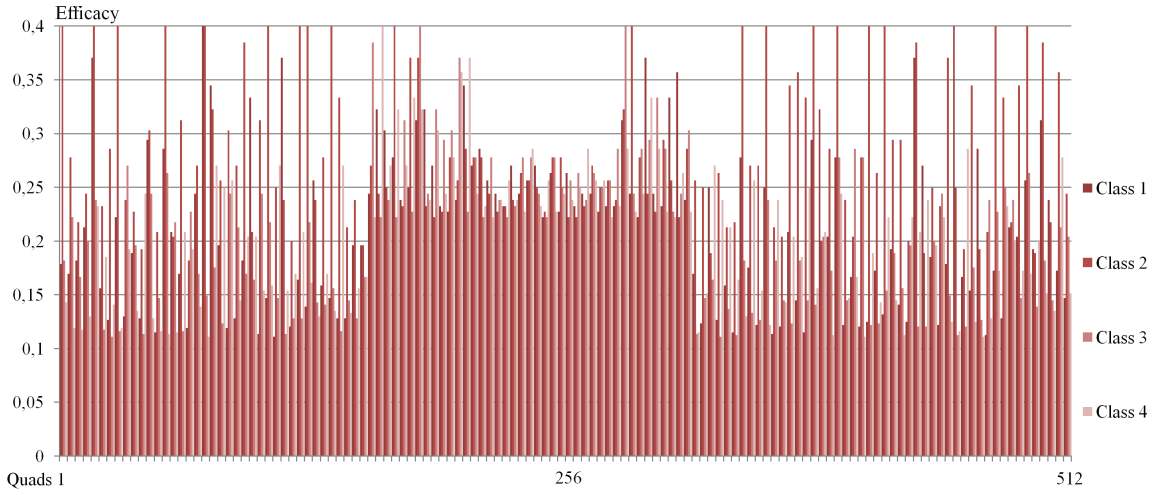


Figure 7.21: Trajectories of the robot after 250 thousand collision free iterations in the heterosynaptic experimental condition.

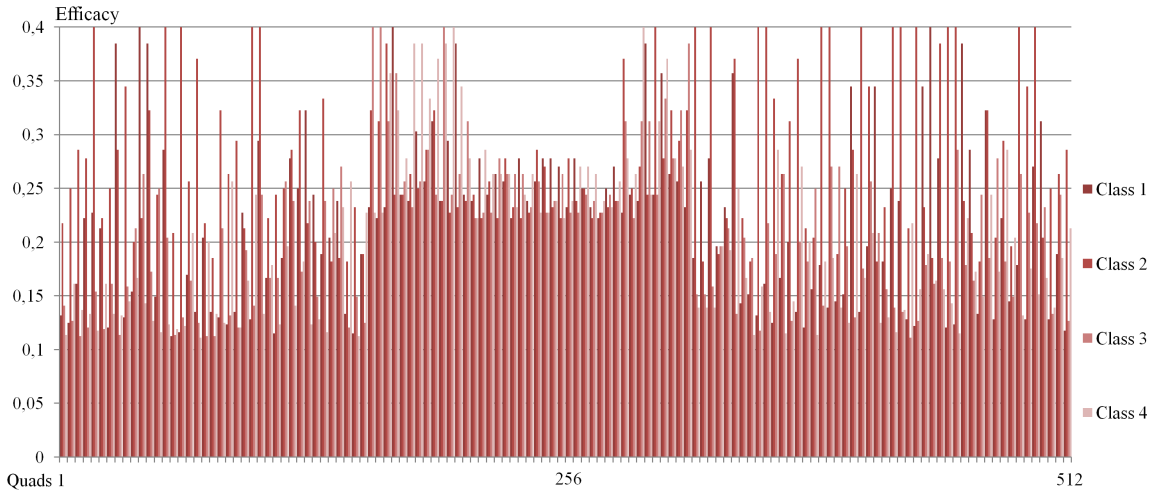
No significant change in behaviour was observed as the robot continued moving (between 100 to 250 thousand collision free iterations). Both aversive behaviour towards obstacles landmarks and pursue towards reward-related landmarks were still observed without changes in the response distance.

Resulting synaptic weights in the heterosynaptic implementation

The same section of the visual matrix described in the previous experimental condition (see Figure 7.18) was used in this experiment in order to analyse the resulting synaptic weights between the classifier neurons in the quad-groups and the motoneurons R and M. Figure 7.22(a) show the weights of the 512 synapses with motoneuron R after running the experiment for 100 thousand collision-free iterations.



(a)



(b)

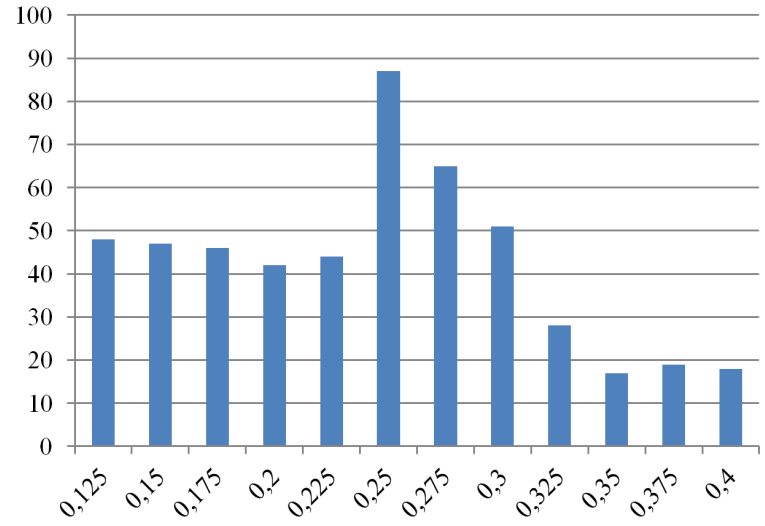
Figure 7.22

As in the non-heterosynaptic experimental condition, each vertical bar shows the weight-value that represents the strength at which the corresponding primitive classifier is associated with the activation of motoneuron R. The synaptic weights shown in Figure

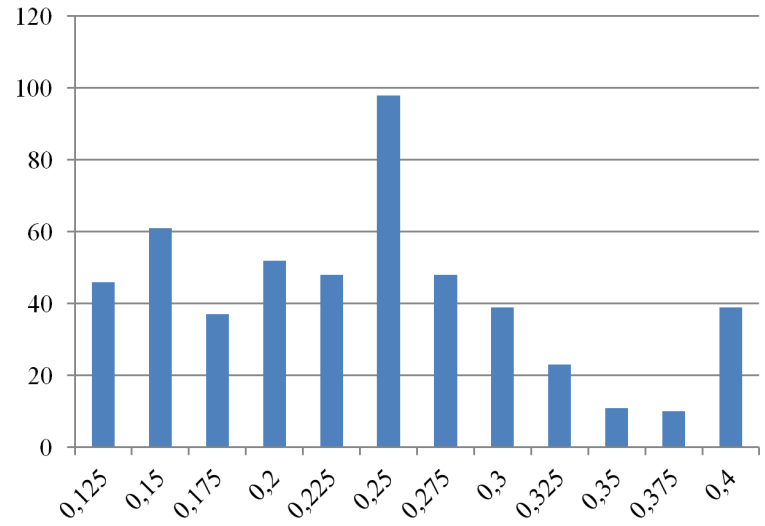
7.22(a) reached a minimum and maximum value of 0.11 and 0.4 respectively. The average weight value was 0.22 with a median of 0.22 and a standard deviation of 0.07.

Figure 7.22(b) illustrates the same synaptic weights after 250 thousand iterations. By looking at the graphic both synaptic weights in (a) and (b) seems to have a similar behaviour. The minimum measured value of a synaptic weight after 200 thousand iterations was 0.11 with an average value of 0.22, a median of 0.23 and a standard deviation of 0.07.

In order to more clearly visualize the change in behaviour of the synaptic values shown in Figures 7.22(a) and (b) The following two histograms show the number of synaptic weights grouped within various ranges of synaptic strength.



(a)



(b)

Figure 7.23

The diagrams in Figures 7.23 (a) and (b) show that after more iterations are run, the distribution of the synaptic weights changes as it is expected, since the system is continuously learning. However, the observed distribution of the synaptic weights does not show the same characteristics as in the first experiment described in this chapter. The distribution of values is not skewed towards higher values closer to the ceiling and in fact the distribution of values across the entire weight range is not affected by the number of iterations, with an unchanged mean and standard deviation.

7.3 Summary

As seen so far, the observed trend to general increase in the synaptic weights in the non-heterosynaptic implementation is a result of mutually synaptic reinforcement that occurs between the visual-classifier neurons. That is, when the stronger synapses from the visual classifiers elicit the activation or firing in the motoneurons, this activation consequently reinforce the weaker synapses that were trying to activate the target neuron shortly before the firing event. In other words, the stronger visual synapses reinforce or recruit the weaker synapses. Therefore, in the long term the synaptic population is entirely potentiated leading the robot to the behavior previously described in the non-heterosynaptic experimental condition. The evolution of the synaptic weights in this case coincides with other experimental observations in the literature [92, 93], where the memory capacity of traditional hebbian-based plasticity systems becomes saturated over time leading to the loss of previously formed memories and a reduction in computational capacity.

On the other hand, the robot in the heterosynaptic implementation has been able to associate visual landmarks with motor reflex responses. Moreover, the SNN plasticity-system demonstrated that already established visual-motor associations are not degraded over time by the continuous flow of information occurring whilst the robot is still exploring its environment. Therefore, the heterosynaptic plasticity modulation successfully regulates the formation of new associations while maintaining the consolidation and preservation of previously acquired learning.

With regard to the first question at the beginning of this chapter: To what extent was the robot in both experimental conditions able to navigate autonomously based on (unsupervised) learnt visual information from the environment?
whit respect to the non-heterosynaptic experimental condition the robot demonstrated that it was able to learnt and navigate on its own however this was only the case for a limited period of time. As the neural controller continued acquiring stimuli and consequently learning, runaway dynamics degraded the previously acquired memories eventually completely impairing the navigation of the robot as shown in figure 7.17. By contrast, the situation in the heterosynaptic experimental condition demonstrated that the robot was able to learnt and navigate on its own without showing any degradation of its memories and behaviour during the entire execution of the experiment. This, also answers the second question formulated at the beginning of this chapter: Were the runaway dynamics emerging from the embedded STDP learning mechanism overcome by the heterosynaptic modulatory architecture of the system? The behaviour of the synaptic weights at the beginning (figure 7.22(a)) and at the end (figure 7.22(b)) of the experiment demonstrates that there was no degradation (weights were not skewed towards its maximum value) in the distribution of the synaptic weights. Moreover, as mentioned before the robot behaviour was not impaired over time. Thus, for the experiments described in this chapter, the proposed heterosynaptic modulatory system was able to overcome hebbian related runaway dynamics.

Chapter 8

Conclusions and Future Work

This thesis briefly reviewed the history of neural network research and addressed the problem that in third generation neural networks the conventional homosynaptic neural model is unsatisfactory because it leads to chaotic behaviour which has, at best, been addressed by ad-hoc approaches that do not generalize into a coherent theory.

The response to this problem described in these was to develop a new theory and model of spiking neurons in which the homosynaptic approach is replaced by a completely new heterosynaptic architecture. Apart from its superior theoretical properties this thesis has shown that the approach works in practice by building a working simulation programs and implementing the theory on a real robot.

8.1 The Research Hypothesis

The Research Hypothesis of this thesis was given as:

The conventional homosynaptic neural model is unsatisfactory but

(i) can be extended to heterosynaptic models that explicitly represent chemical regulation of the electrical spiking dynamics that

(ii) overcome the weaknesses of the homosynaptic model, and

(iii) can be implemented in practical systems

This hypothesis has been demonstrated to be true by
(i) developing a new heterosynaptic theory and model that explicitly represents chemical regulation of the electrical spiking dynamics;
(ii) implementing this model and showing that it does not have the 'runaway' behaviour of the homosynaptic model; and
(iii) the implementation of models within a NetLogo simulation environment, and the implementation of a real Lego robot.

8.2 Discussion

This thesis presented the development and experimental implementation of a novel SNN system in which neuron excitability and synaptic plasticity are modulated by means of heterosynaptic transmission. The proposed system implemented the two mechanisms of excitability and plasticity modulation by embedding their dynamics into an extended artificial neuron model that not only implements the traditional spike-based synaptic transmission but also supports non-electrical signalling between modulatory neurons and pulse-driver (traditional) neurons.

This thesis described the implementation of the system through an algorithmic and systemic framework which explained the different components that intervene in the proposed heterosynaptic approach. *i.e.* pulse driver neurons, modulatory neurons, modulatory substances, synapses and their collective as a network.
Basic circuit topologies and spike timing constraints were explained and experimentally tested in order to serve as guidelines for further research and engineering applications.

In the first experiment described in Chapter 6 a simple SNN circuit was implemented to control a simulated insect in a two dimensional environment. Both non-heterosynaptic and heterosynaptic implementations were compared in terms of the learning and adaptation observed in the simulated insects.

The heterosynaptic implementation demonstrated the use of efficacy modulation as a mechanism to encourage exploration and risk prone behaviour in autonomous systems by temporarily inhibiting the conditioned motor responses elicited by associated (learned) visual stimuli. This experiment demonstrated that it is possible to achieve more complex and intelligent behaviour through the combination of efficacy and plasticity modulation without the necessity to significantly increase the number of neurons and synapses in a SNN circuit.

The second experiment described in chapter 7 demonstrated the implementation of the proposed heterosynaptic system in order to avoid the emergence of 'runaway' dynamics in a STDP-based learning system. Two experimental conditions were implemented and compared. In the first experimental condition, the non-heterosynaptic implementation showed that without any weight normalization or control mechanism the system tended

to degrade over time affecting the integrity of previous acquired memories. Moreover, the runaway dynamics were reflected in the motion behaviour of the robot where it was observed that after several iterations the system was not able to navigate successfully along the test field due to the over-reinforced association between visual inputs and the avoidance-reflex response. In contrast, the second experimental condition which implemented plasticity modulation through the proposed heterosynaptic neural architecture, was able to keep the integrity of the synaptic weights over time. The observed motion behaviour of the robot in the second experimental condition was consistent during the tests showing the robot ability to recognize landmarks associated to avoidance as well as landmarks related to reward that encouraged the robot to approach them.

In both chapters 6 and 7 the architecture used in the experiments for the modulation of synaptic plasticity required the modulatory signal to arrive before the triggering of the hebbian coactivation. In other words the behaviour of the modulated target neurons was changed a priori in relation to the input (stimuli) to be reinforced. This implementation contrasts with traditional reward based learning where (as explained before in Section 3.6) the modulatory signal arrives after the event that triggers the synaptic change. This shows that the proposed system is not constrained by the requirements of a reward-based design. However, the tuning of the timing between the modulatory signals and the spiking activity triggering the hebbian plasticity is still a challenging task. In the proposed architecture of the implemented experiments, the timing between the modulatory signals and the affected spiking activity was controlled through the (axonal) synaptic delays of the modulatory neurons, sensory neurons and receptors (see Figures 6.5, 6.6 and 7.10). Through the addition of different synaptic delays between layers of pulse driver neurons and modulatory neurons, it was possible to control the sequence in which the spike activity was propagated among the neural circuit and thus the precedence of modulatory signals to plasticity events. However, other control mechanisms for the timing between modulatory signals and spiking activity need to be explored.

8.3 Further research directions

After bringing the research reported in thesis to a conclusion there are many open questions and possible new research directions. Some of these are discussed below.

Mathematical Theory

The research in this thesis adopted a systems approach in which the new heterosynaptic mechanism was developed based on theoretical considerations that motivated the particular mathematical formulae used. There is scope for further research into the general behaviours of these mathematical formulae and variants of them.

This suggests the possibility of predicting the parameters for heterosynaptic systems rather than using the empirical approach adopted in this research.

Massive spiking neural systems as larger neural circuits using evolutionary algorithms

The research in this thesis has demonstrated the validity of the new heterosynaptic architecture for a relatively small number of neurons. It remains an open question as to how the approach scales up to thousands or millions of neurons.

An important research challenge is scaling up to very large systems using evolutionary algorithms to determine connectivities and parameters. Evolutionary algorithms provide an interesting research direction here.

More complicated environments

The simulations and real robot used to test the new theory developed in this thesis were relatively simple. A next step in the research is to experiment with more complicated environments with robots having few and many neurons.

Subsystem specialization and multilevel systems

In biological systems there is specialization at the macrolevel with neurons grouping, for example, into brain regions. It is an open question as to how the new heterosynaptic neurons may form specialized subsystems.

Implementation in hardware

In the longer term it is conceivable that standard heterosynaptic subsystems could be implemented in hardware chips, greatly increasing the efficiency of the systems that can

be built and increasing the scope of what they can learn.

8.4 Critical reflections on the research

Although the research reported in this thesis is a success in terms of identifying a weakness in the current theory and suggesting an operational new architecture that over comes it, the ideas given in the previous section show that there remain many research questions and directions that were outside the scope of the study.

Scalability is the major open question to this research. Does the architecture developed here only work on small example, or can it be extended to much larger systems and much more demanding tasks? As suggested in the previous sections there are ways to address this question, including new mathematical and empirical research.

Appendix A

Sample Code of Insect Model implemented in Netlogo

```
;; Spiking Neural Networks with STDP learning
;; Author: Cristian Jimenez Romero - The Open University - 2015

breed [neurontypes neurontype]
breed [inputneurons inputneuron]
breed [normalneurons normalneuron]
directed-link-breed [normal-synapses normal-synapse]
directed-link-breed [input-synapses input-synapse]

 neurontypes-own
 [
   neurontypeid
   ;;;;;;;;;;Neuron Dynamics;;;;;;;;;;;
   restingpotential
   firethreshold
   decayrate ;Decay rate constant
   relrefractorypotential ;Refractory potential
   intervrefractoryperiod ;Duration of absolute-refractory period
   minmembranepotential ;lowest boundary for membrane potential
   ;;;;;;;Learning Parameters;;;;;;;;;
   pos_hebb_weight ;;Weight to increase the efficacy of synapses
   pos_time_window ;; Positive learning window
   neg_hebb_weight ;;Weight to decrease the efficacy of synapses
   neg_time_window ;; negative learning window
   max_synaptic_weight ;;Maximum synaptic weight
   min_synaptic_weight ;;Minimum synaptic weight
   pos_syn_change_interval ;;ranges of pre-to-post synaptic interspike intervals
   ; over which synaptic change occur.
   neg_syn_change_interval
   neuronsystemtype ;type of neuron 0 - Spiking, 1 - Efficacy modulator,
   ;2 - Plasticity modulator
 ]

 ;;; Create a neuron type
 ;;; Parameters: Type-identifier, resting potential, firing threshold, decay
 ; rate, refractory potential, duration of refractory period, lowest boundary for
 ; membrane potential
 to setup-neurontype [#pneurontypeid #prestpot #pthreshold #pdecayr #prefractpot
  #pintrefrectp #minmembranepotential #pneuronsystemtype]
   create-neurontypes 1
   [
      set shape "square"
      set neurontypeid #pneurontypeid
      set restingpotential #prestpot
      set firethreshold #pthreshold
      set decayrate #pdecayr
```

```
47       set relrefractorypotential #prefractpot
48       set intervrefractoryperiod #pintrefrectp
49       set minmembranepotential #minmembranepotential
50       set neuronsystemtype #pneuronsystemtype
51    ]
52  end
53
54  ;;; Set learning parameters for neuron type pneurontypeid
55  ;;;
56  to set-neurontype-learning-params [ #pneurontypeid #ppos_hebb_weight
57   #ppos_time_window #pneg_hebb_weight #pneg_time_window #pmax_synaptic_weight
58                                      #pmin_synaptic_weight
59                                      #ppos_syn_change_interval
60                                      #pneg_syn_change_interval]
61
62    ask neurontypes with [neurontypeid = #pneurontypeid]
63    [
64      set pos_hebb_weight #ppos_hebb_weight
65      set pos_time_window #ppos_time_window
66      set neg_hebb_weight #pneg_hebb_weight
67      set neg_time_window #pneg_time_window
68      set max_synaptic_weight #pmax_synaptic_weight
69      set min_synaptic_weight #pmin_synaptic_weight
70      set pos_syn_change_interval #ppos_syn_change_interval
71      set neg_syn_change_interval #pneg_syn_change_interval
72    ]
73  end
74
75  ;;; Declare an existing neuron "pneuronid" as neuron-type "pneurontypeid"
76  ;;;
77  to set-neuron-to-neurontype [ #pneurontypeid #pneuronid ] ;Call by observer
78      ask neurontypes with [neurontypeid = #pneurontypeid]
79      [
80        ask normalneuron #pneuronid
81        [
82          set nrestingpotential [restingpotential] of myself
83          set nmembranepotential [restingpotential] of myself
84          set nfirethreshold [firethreshold] of myself
85          set ndecayrate [decayrate] of myself
86          set nrelrefractorypotential [relrefractorypotential] of myself
87          set nintervrefractoryperiod [intervrefractoryperiod] of myself
88          set nminmembranepotential [minmembranepotential] of myself
89          set nsystemtype [neuronsystemtype] of myself
90          ;;;;;;;;;;;;;;;;Learning Parameters;;;;;;;;;;;;;;;;
91          set npos_hebb_weight [pos_hebb_weight] of myself
92          set npos_time_window [pos_time_window] of myself
```

```
93          set nneg_hebb_weight [neg_hebb_weight] of myself
94          set nneg_time_window [neg_time_window] of myself
95          set nmax_synaptic_weight [max_synaptic_weight] of myself
96          set nmin_synaptic_weight [min_synaptic_weight] of myself
97          set npos_syn_change_interval [pos_syn_change_interval] of myself
98          set nneg_syn_change_interval [neg_syn_change_interval] of myself
99        ]
100       ]
101 end
102
103 normalneurons-own [
104   nlayernum
105   nneuronid
106   nneuronstate
107   nrestingpotential
108   nfirethreshold
109   nmembranepotential
110   ndecayrate
111   nrelrefractorypotential
112   nintervrefractoryperiod
113   nrefractorycounter
114   naxondelay
115   nsynapsesarray
116   nnumofsynapses
117   nlast-firing-time
118   nincomingspikes
119   nlastspikeinput
120   nneuronlabel
121   nminmembranepotential
122   nsystemtype
123   ;;;;;;;;;;;;;;;;Learning Parameters;;;;;;;;;;;;;;;;
124   npos_hebb_weight ;;Weight to increase the efficacy of synapses
125   npos_time_window ;; Positive learning window
126   nneg_hebb_weight ;;Weight to decrease the efficacy of synapses
127   nneg_time_window ;; negative learning window
128   nmax_synaptic_weight ;;Maximum synaptic weight
129   nmin_synaptic_weight ;;Minimum synaptic weight
130   npos_syn_change_interval ;;ranges of pre-to-post synaptic interspike
131   ; intervals over which synaptic change occur.
132   nneg_syn_change_interval
133 ]
134
135 normal-synapses-own [
136   presynneuronlabel
137   possynneuronlabel
138   presynneuronid
```

```
139    possynneuronid
140    synapseefficacy
141    exc_or_inh
142    synapsedelay
143    joblist
144    learningon?
145    efficacymodon?
146    modulated_learning?
147  ]
148
149  inputneurons-own [
150     layernum
151     neuronid
152     neuronstate
153     pulsecounter ;;Count the number of sent pulses
154     interspikecounter ;;Count the time between pulses
155     numberofspikes ;;Number of spikes to send
156     postsynneuron
157     encodedvalue
158     isynapseefficacy ;;In most cases should be large enough to activate
159     ; possynn with a single spike
160     neuronlabel
161  ]
162
163  to-report get-input-neuronid-from-label [#pneuronlabel]
164    let returned_id nobody
165    ask one-of inputneurons with [neuronlabel = #pneuronlabel][set returned_id neuronid]
166    report returned_id
167  end
168
169  to-report get-neuronid-from-label [#pneuronlabel]
170    let returned_id nobody
171    ask one-of normalneurons with [nneuronlabel = #pneuronlabel][set returned_id nneuronid]
172    report returned_id
173  end
174
175  ;;; Create a new synapse between pre-synaptic neuron: #ppresynneuronlabel and
176  ;;; post-synaptic neuron: #ppossynneuronlabel
177  ;;;
178  to setup-synapse [#ppresynneuronlabel #ppossynneuronlabel #psynapseefficacy
179   #pexc_or_inh #psyndelay #plearningon? #pefficacymodon? #modulated_learning?]
180     let presynneuid get-neuronid-from-label #ppresynneuronlabel
181     let possynneuid get-neuronid-from-label #ppossynneuronlabel
182     let postsynneu normalneuron possynneuid
183     ask normalneuron presynneuid [
184          create-normal-synapse-to postsynneu [
```

```
185             set presynneuronlabel #ppresynneuronlabel
186             set possynneuronlabel #ppossynneuronlabel
187             set presynneuronid presynneuid
188             set possynneuronid possynneuid
189             set synapseefficacy #psynapseefficacy
190             set exc_or_inh #pexc_or_inh
191             set synapsedelay #psyndelay
192             set joblist []
193             set learningon? #plearningon?
194             set efficacymodon? #pefficacymodon?
195             set modulated_learning? #modulated_learning?
196             ifelse (#pexc_or_inh = inhibitory_synapse)
197             [
198               set color red
199             ]
200             [
201               set color grey
202             ]
203          ]
204       ]
205   end
206
207   ;;; Process incoming pulse from input neuron
208   ;;;
209   to receive-input-neuron-pulse [ #psnefficacy #pexcinh ];;called by Neuron
210     if ( nneuronstate != neuron_state_refractory )
211     [
212         ;;Adjust membrane potential:
213         ifelse ( #pexcinh = excitatory_synapse )
214         [
215             set nmembranepotential nmembranepotential + #psnefficacy  ;;increase
216             ; membrane potential
217         ]
218         [
219             set nmembranepotential nmembranepotential - #psnefficacy ;;decrease
220             ; membrane potential
221             if (nmembranepotential < nminmembranepotential) ;; Floor for the
222             ; membrane potential in case of extreme inhibition
223             [
224                set nmembranepotential nminmembranepotential
225             ]
226         ]
227     ]
228     set nlastspikeinput ticks
229   end
230
```

```
231 ;;; Neuron pstneuronid# processes incoming pulse from neuron #prneuronid
232 ;;;
233 to receive-pulse [ #prneuronid #snefficacy #excinh #plearningon? ];;called by neuron
234   if ( nneuronstate != neuron_state_refractory )
235   [
236       ;;Perturb membrane potential:
237       ifelse ( #excinh = excitatory_synapse )
238       [
239           set nmembranepotential nmembranepotential + #snefficacy ;;increase
240           ; membrane potential
241       ]
242       [
243           set nmembranepotential nmembranepotential - #snefficacy ;;decrease
244           ; membrane potential
245           if (nmembranepotential < nminmembranepotential) ;; Floor for the
246           ; membrane potential in case of extreme inhibition
247           [
248              set nmembranepotential nminmembranepotential
249           ]
250       ]
251   ]
252   ;;Remember last input spike:
253   set nlastspikeinput ticks
254   ;; If plasticity is activated then store pulse info for further processing and
255   ; apply STDP
256   if (#plearningon? and istrainingmode?)
257   [
258      let pulseinflist[] ;;Create list of parameters and populate it;
259      set pulseinflist lput #prneuronid pulseinflist ;;Add Presynaptic neuron Id
260      set pulseinflist lput #snefficacy pulseinflist ;;Add Synaptic efficacy
261      set pulseinflist lput #excinh pulseinflist ;;Add excitatory or inhibitory info.
262      set pulseinflist lput ticks pulseinflist ;;Add arriving time
263      set pulseinflist lput false pulseinflist ;;Indicate if pulse has been
264      ; processed as an EPSP following a Postsynaptic spike ( Post -> Pre,
265      ; negative hebbian)
266      ;;Add list of parameters to list of incoming pulses:
267      set nincomingspikes lput pulseinflist nincomingspikes
268      ;;Apply STDP learning rule:
269      apply-stdp-learning-rule
270   ]
271 end
272
273 ;;;
274 ;;; Neuron fires
275 ;;;
276 to prepare-pulse-firing ;;Called by Neurons
```

```
277     ifelse (nsystemtype = 1) ;efficacy modulator
278     [
279        if (efficacy_modulation?) [ set syneff_down_reg_factor syneff_min_modulator ]
280     ]
281     [
282       ifelse (nsystemtype = 2) ;Plasticity modulator
283       [
284          set plasticity_modulator max_plasticity_modulator
285       ]
286       [
287         if (nsystemtype = 3) ;Efficacy and Plasticity modulator
288         [
289           if (efficacy_modulation?) [ set syneff_down_reg_factor syneff_min_modulator ]
290           set plasticity_modulator max_plasticity_modulator
291         ]
292       ]
293     ]
294     ;;if neuron has neuromodulatory properties then apply them:
295     ;if (nplasticity_modulating_factor > 0) [ set plasticity_modulator
296     ; nplasticity_modulating_factor ]
297     ;;Remember last firing time
298     set nlast-firing-time ticks
299     ;;Apply learning rule and after that empty incoming-pulses history:
300     apply-stdp-learning-rule
301     empty-pulse-history-buffer
302     ;;transmit Pulse to postsynaptic neurons:
303     propagate-pulses
304     ;;Set State to refractory
305     set nneuronstate neuron_state_refractory
306     ;;initialize counter of refractory period in number of iterations
307     set nrefractorycounter nintervrefractoryperiod
308   end
309
310   ;;; Kernel for inhibitory post-synaptic potential
311   ;;;
312   to-report ipsp-kernel ;;Called by Neurons
313     report 1
314   end
315
316   ;;; Kernel for excitatory post-synaptic potential
317   ;;;
318   to-report epsp-kernel ;;Called by Neurons
319     report 1
320   end
321
322   ;;;
```

```
323 ;;; Kernel for membrane decay towards resting potential (If current membrane pot. > Resting pot.
324 ;;;
325 to-report negative-decay-kernel ;;Called by Neurons
326   report (exp ( -( ticks - nlastspikeinput ) / 5 ) + ndecayrate)
327 end
328
329 ;;;
330 ;;; Kernel for membrane decay towards resting potential (If current membrane
331 ; pot. < Resting pot.)
332 ;;;
333 to-report positive-decay-kernel ;;Called by Neurons
334   report (exp (3 - ( ticks - nlast-firing-time ) ^ 0.8) + 0.3)
335 end
336
337 ;;;
338 ;;; Bring membrane potential towards its resting state
339 ;;;
340 to decay ;;Called by Neurons
341     ;;Move membrane potential towards resting potential:
342     ifelse (nmembranepotential > nrestingpotential )
343     [
344         let expdecay negative-decay-kernel ;
345         ifelse ((nmembranepotential - expdecay) < nrestingpotential)
346         [
347           set nmembranepotential nrestingpotential
348         ]
349         [
350           set nmembranepotential nmembranepotential - expdecay
351         ]
352     ]
353     [
354       let expdecay positive-decay-kernel ;
355       ifelse ((nmembranepotential + expdecay) > nrestingpotential)
356       [
357         set nmembranepotential nrestingpotential
358       ]
359       [
360         set nmembranepotential nmembranepotential + expdecay
361       ]
362     ]
363 end
364
365 ;;;
366 ;;; Process neuron dynamics according to its machine state
367 ;;;
368 to do-neuron-dynamics ;;Called by Neurons
```

```
369    ifelse ( nneuronstate = neuron_state_open )
370    [
371      if (nmembranepotential != nrestingpotential )
372      [
373        ;;Check if membrane potential reached the firing threshold
374        ifelse( nmembranepotential >= nfirethreshold )
375        [
376          prepare-pulse-firing
377          set color red
378        ]
379        [
380           ;;Move membrane potential towards resting potential:
381           decay
382        ]
383        ;;Don't let incoming-pulses history-buffer grow beyond limits (delete oldest spike):
384        check-pulse-history-buffer
385      ]
386    ]
387    [
388      ;;Not idle and not firing, then refractory:
389      set color pink ;;Restore normal colour
390      ;;Decrease timer of absolute refractory period:
391      set nrefractorycounter nrefractorycounter - system_iter_unit
392      ;;Set membrane potential with refractory potential:
393      set nmembranepotential nrelrefractorypotential
394      if ( nrefractorycounter <= 0) ;;End of absolute refractory period?
395      [
396        ;;Set neuron in open state:
397        set nneuronstate neuron_state_open
398      ]
399      ;;Don't let incoming-pulses history-buffer grow beyond limits (delete oldest spike):
400      check-pulse-history-buffer
401    ]
402    ;;Continue with Axonal dynamics independently of the neuron state:
403    do-synaptic-dynamics
404  end
405
406  ;;;
407  ;;; Delete history of incoming spikes
408  ;;;
409  to empty-pulse-history-buffer ;;Called by neurons
410    set nincomingspikes[]
411  end
412
413  ;;;
414  ;;; Apply the Spike Timing Dependent Plasticity rule
```

```
;;;
to apply-stdp-learning-rule ;;Call by neurons
  ;Apply rule: Ap.exp(dt/Tp); if dt < 0; dt = prespt - postspt
  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  let currspikeinfo[]
  let itemcount 0
  let deltaweight 0
  while [itemcount < ( length nincomingspikes ) ]
  [
     set currspikeinfo ( item itemcount nincomingspikes ) ;;Get spike info:
       prneuronid[0], snefficacy[1], excinh[2], arrivaltime[3],
        processedBynegHebb[4]

     ifelse ( item 2 currspikeinfo ) = excitatory_synapse ;;Is the spike coming
     ; from an excitatory synapsis?
     [
       let deltaspike ( item 3 currspikeinfo ) - nlast-firing-time
       if ( deltaspike >= nneg_time_window and deltaspike <= npos_time_window)
        ;;Is spike within learning window?
       [
           ;;Calculate learning factor:
          ifelse ( deltaspike <= 0 ) ;;Increase weight
          [
            set deltaweight  npos_hebb_weight * exp(deltaspike /
             npos_syn_change_interval )
            ask normal-synapse  ( item 0 currspikeinfo ) nneuronid
             [update-synapse-efficacy deltaweight [nmax_synaptic_weight] of
              myself [nmin_synaptic_weight] of myself]
          ]
          [
            if (( item 4 currspikeinfo ) = false) ;;if spike has not been
             ;processed then compute Hebb rule:
            [
              set deltaweight (- nneg_hebb_weight * exp(- deltaspike /
               nneg_syn_change_interval )) ;;Turn positive delta into a negative
               ; weight
              set currspikeinfo replace-item 4 currspikeinfo true ;Indicate that
              ; this pulse has already been processed as a EPSP after
              ; Postsyn neuron has fired (negative hebbian)
              set nincomingspikes replace-item itemcount nincomingspikes currspikeinfo
              ask normal-synapse  ( item 0 currspikeinfo ) nneuronid
               [update-synapse-efficacy deltaweight [nmax_synaptic_weight] of
                myself [nmin_synaptic_weight] of myself]
            ]
          ]
       ]
```

```
461       ]
462       [
463         ;;Inhibitory Synapses: Plasticity in inhibitory synapses not implemented yet
464
465       ]
466       set itemcount itemcount + 1
467     ]
468   end
469
470   ;;;
471   ;;; Don't store more pulses than the specified by PulseHistoryBuffSize
472   ;;;
473   to check-pulse-history-buffer ;;Call by neurons
474     if( length nincomingspikes > PulseHistoryBuffSize )
475     [
476       ;; Remove oldest pulse in the list
477       set nincomingspikes remove-item 0 nincomingspikes
478     ]
479   end
480
481   ;;;
482   ;;; Propagate pulse to all post-synaptic neurons
483   ;;;
484   to propagate-pulses ;;Call by neurons
485     ;; Insert a new pulse in all synapses having the current neuron as presynaptic
486     ask my-out-normal-synapses
487     [
488       add-pulse-job
489     ]
490   end
491
492   ;;;
493   ;;; Process synaptic dynamics of synapses with pre-synaptic neuron: presynneuronid
494   ;;;
495   to do-synaptic-dynamics ;;Call by neurons
496     ;; Process all synapses with presynaptic neuron = nneuronid and pulses in their job-list
497     ask my-out-normal-synapses with [ length joblist > 0 ]
498     [
499       process-pulses-queue
500     ]
501   end
502
503   ;;;
504   ;;; Enqueue pulse in synapse
505   ;;;
506   to add-pulse-job ;;Call by link (synapse)
```

```
507      ;;Add a new Pulse with its delay time at the end of the outgoing-synapse joblist
508      set joblist lput synapsedelay joblist
509  end
510
511  ;;; Change synaptic weight
512  ;;;
513  to update-synapse-efficacy [ #deltaweight #pmax_synaptic_weight
514   #pmin_synaptic_weight] ;;Call by synapse
515
516    if (modulated_learning?)
517    [
518      set #deltaweight #deltaweight * plasticity_modulator
519    ]
520
521    ifelse ( synapseefficacy + #deltaweight ) > #pmax_synaptic_weight
522    [
523      set synapseefficacy #pmax_synaptic_weight
524    ]
525    [
526      ifelse ( synapseefficacy + #deltaweight ) < #pmin_synaptic_weight
527      [
528        set synapseefficacy #pmin_synaptic_weight
529      ]
530      [
531        set synapseefficacy synapseefficacy + #deltaweight
532      ]
533    ]
534  end
535
536  ;;;
537  ;;; For each traveling pulse in synapse check if pulse has already arrived at
538  ; the post-synaptic neuron
539  ;;;
540  to process-pulses-queue ;;Call by synapse
541    let efficacyfactor 1
542    if (efficacymodon?)
543    [
544      set efficacyfactor syneff_down_reg_factor
545    ]
546    set joblist map [ ? - 1  ] joblist ;;Decrease all delay counters by 1 time-unit
547    foreach filter [? <= 0] joblist
548    [
549         ;;Transmit Pulse to Postsyn Neuron:
550         ask other-end [receive-pulse [presynneuronid] of myself (([synapseefficacy] of
551         myself) * efficacyfactor) [exc_or_inh] of myself [learningon?] of myself]
552    ]
```

```
553    ;;Keep only "traveling" pulses in the list :
554    set joblist filter [? > 0] joblist
555  end
556
557  ;;;
558  ;;; Create one input neuron and attach it to neuron with label
559  ; #ppostsynneuronlabel (input neurons have one connection only)
560  ;;;
561  to setup-input-neuron [#pposx #pposy #label #ppostsynneuronlabel
562   #psynapseefficacy #pcoding #pnumofspikes]
563    let postsynneuronid get-neuronid-from-label #ppostsynneuronlabel
564    set-default-shape inputneurons "square"
565    create-inputneurons 1
566    [
567       set layernum 0
568       set neuronid who
569       set neuronstate neuron_state_open
570       set pulsecounter 0
571       set interspikecounter 0
572       set numberofspikes #pnumofspikes
573       set postsynneuron postsynneuronid
574       set encodedvalue input_value_empty
575       set isynapseefficacy #psynapseefficacy
576       setxy #pposx #pposy
577       set color green
578       set label #label
579       set neuronlabel #label
580       setup-input-synapse
581    ]
582  end
583
584  ;;;
585  ;;; Process pulses in input neuron
586  ;;;
587  to do-input-neuron-dynamics ;;Called by inputneurons
588    if ( pulsecounter > 0 ) ;;process only if input-neuron has something to do
589    [
590      set interspikecounter interspikecounter + 1
591      if (interspikecounter > encodedvalue)
592      [
593      ;;Transmit pulse to Post-synaptic Neuron;
594        ask out-input-synapse-neighbors [receive-input-neuron-pulse [isynapseefficacy] of
595        myself [excitatory_synapse] of myself]
596        set interspikecounter 0
597        set pulsecounter pulsecounter - 1
598      ]
```

```
599    ]
600  end
601
602  ;;;
603  ;;; Encode input value (integer number) into pulses
604  ;;;
605  to-report set-input-value [#pencodedvalue] ;;Called by inputneurons
606    ;;Check if input neuron is ready to receive input
607    let inputready false
608    if ( pulsecounter = 0 )
609    [
610      set encodedvalue #pencodedvalue
611      set pulsecounter numberofspikes ;;Initialize counter with the number of pulses to
612      ; transmit with the encoded value
613      set interspikecounter 0
614      set inputready true
615    ]
616    report inputready
617  end
618
619  ;;;
620  ;;; Ask input neuron with id = #pneuronid to accept and encode a new input value
621  ;;;
622  to feed-input-neuron [#pneuronid #pencodedvalue];;Called by observer
623     ask inputneuron #pneuronid
624     [
625       let inputready set-input-value #pencodedvalue
626     ]
627  end
628
629  ;;;
630  ;;; Ask input neuron with label = #pneuronlabel to accept and encode a new input value
631  ;;;
632  to feed-input-neuron_by_label [#pneuronlabel #pencodedvalue];;Called by observer
633     ask one-of inputneurons with [ neuronlabel = #pneuronlabel ]
634     [
635       let inputready set-input-value #pencodedvalue
636     ]
637  end
638
639  ;;;
640  ;;; Create link to represent synapse from input neuron to post-synaptic neuron: postsynneuron
641  ;;;
642  to setup-input-synapse ;;Call from inputneurons
643     let psnneuron postsynneuron
644     let postsynneu one-of (normalneurons with [nneuronid = psnneuron])
```

```
645     create-input-synapse-to postsynneu
646  end
647
648  ;;;
649  ;;; Create and initialize neuron
650  ;;;
651  to setup-normal-neuron [#playernum #pposx #pposy #label #pneurontypeid]
652    set-default-shape normalneurons "circle"
653    let returned_id nobody
654    create-normalneurons 1
655    [
656       set nlayernum #playernum
657       set nneuronid who
658       set nneuronstate neuron_state_open
659       set nrefractorycounter 0
660       set nincomingspikes[]
661       set nnumofsynapses 0
662       set nlastspikeinput 0
663       setxy #pposx #pposy
664       set color pink
665       set label #label
666       set nneuronlabel #label
667       set returned_id nneuronid
668    ]
669    set-neuron-to-neurontype #pneurontypeid returned_id
670  end
671
672  ;;;
673  ;;; Process neural dynamics
674  ;;;
675  to do-network-dynamics
676   ask inputneurons [ do-input-neuron-dynamics ] ;with [pulsecounter > 0]
677   foreach sort-on [nlayernum] normalneurons [ ;;with [nneuronstate !=
678    neuron_state_open];; with [nmembranepotential != nrestingpotential]
679     ask ? [
680       do-neuron-dynamics
681     ]
682   ]
683
684    if (syneff_down_reg_factor < 1)
685    [
686       set syneff_down_reg_factor syneff_down_reg_factor + (syneff_down_reg_factor
687        * 0.016);0.012
688    ]
689
690    if ( plasticity_modulator > min_plasticity_modulator )
```

```
691    [
692       set plasticity_modulator plasticity_modulator - ((1.03 - plasticity_modulator) * 0.018)
693    ]
694  end
695
696  ;;;
697  ;;; Show information about neuron with id: #srcneuron
698  ;;;
699  to show-neuron-info [#srcneuron] ;;Called by observer
700     ask normalneurons with [nneuronid = #srcneuron]
701     [
702        print ""
703        write "Neuron Id:" write nneuronid print ""
704        write "Layer " write nlayernum print ""
705        write "Membrane potential: " write nmembranepotential print ""
706        write "Spike threshold: " write nfirethreshold print ""
707        write "Resting potential: " write nrestingpotential print ""
708        write "Refractory potential: " write nrelrefractorypotential print ""
709        write "Last input-pulse received at:" write nlastspikeinput print ""
710        write "Last spike fired at: " write nlast-firing-time print ""
711        write "Current state: " write (ifelse-value ( nneuronstate =
712         neuron_state_open ) ["idle"] ["refractory"] ) print "" print ""
713     ]
714  end
715
716  ;;;
717  ;;; Show information about synapse with pre-synaptic neuron: #srcneuron and
718  ; post-synaptic neuron: #dstneuron
719  ;;;
720  to show-synapse-info-from-to [#srcneuron #dstneuron] ;;Called by observer
721     ask normal-synapses with [presynneuronid = #srcneuron and possynneuronid = #dstneuron]
722     [
723        print "Synapse from:"
724        write "Neuron " write #srcneuron write " to neuron " write #dstneuron print ""
725        write "Weight: " write synapseefficacy print ""
726        write "Delay: " write synapsedelay write "iteration(s)" print ""
727        write "Excitatory or Inhibitory: "
728        write (ifelse-value ( exc_or_inh = excitatory_synapse ) ["Excitatory"] ["Inhibitory"] )
729        print ""
730     ]
731  end
732
733  breed [testcreatures testcreature]
734  breed [visualsensors visualsensor]
735
736  testcreatures-own [
```

```
737    creature_label
738    creature_id
739    reward_neuron
740    pain_neuron
741    move_neuron
742    rotate_neuron
743    creature_sightline
744  ]
745
746  ;;;
747  ;;; Create insect agent
748  ;;;
749  to-report create-creature [#xpos #ypos #creature_label #reward_neuron_label
750   #pain_neuron_label #move_neuron_label #rotate_neuron_label]
751    let reward_neuron_id get-input-neuronid-from-label #reward_neuron_label
752    let pain_neuron_id get-input-neuronid-from-label #pain_neuron_label
753    let move_neuron_id get-neuronid-from-label #move_neuron_label
754    let rotate_neuron_id get-neuronid-from-label #rotate_neuron_label
755    let returned_id nobody
756    create-testcreatures 1 [
757      set shape "bug"
758      setxy #xpos #ypos
759      set size 2
760      set color yellow
761      set creature_label #creature_label
762      set reward_neuron reward_neuron_id
763      set pain_neuron pain_neuron_id
764      set move_neuron move_neuron_id
765      set rotate_neuron rotate_neuron_id
766      set creature_id who
767      set returned_id creature_id
768    ]
769    report  returned_id
770  end
771
772  visualsensors-own [
773    sensor_id
774    perceived_stimuli
775    distance_to_stimuli
776    relative_rotation ;;Position relative to front
777    attached_to_colour
778    attached_to_neuron
779    attached_to_creature
780  ]
781
782  ;;;
```

```
783 ;;; Create photoreceptor and attach it to insect
784 ;;;
785 to create-visual-sensor [ #psensor_id #pposition #colour_sensitive
786  #attached_neuron_label #attached_creature] ;;Called by observer
787   let attached_neuron_id get-input-neuronid-from-label #attached_neuron_label
788   create-visualsensors 1 [
789      set sensor_id #psensor_id
790      set relative_rotation #pposition ;;Degrees relative to current heading -
791       Left + Right 0 Center
792      set attached_to_colour #colour_sensitive
793      set attached_to_neuron attached_neuron_id
794      set attached_to_creature #attached_creature
795      ht
796   ]
797 end
798
799 ;;;
800 ;;; Ask photoreceptor if there is a patch ahead
801 ;;; (within insect_view_distance) with a perceivable colour (= attached_to_colour)
802 ;;;
803 to view-world-ahead ;;Called by visualsensors
804   let itemcount 0
805   let foundobj black
806   ;;;;;;;;;;;;;;;Take same position and heading of creature:;;;;;;;;;;;;;;;
807   let creature_px 0
808   let creature_py 0
809   let creature_heading 0
810   ask  testcreature attached_to_creature [set creature_px xcor set creature_py
811    ycor set creature_heading heading];
812   set xcor creature_px
813   set ycor creature_py
814   set heading creature_heading
815   rt relative_rotation
816   let view_distance insect_view_distance
817   let xview 0
818   let yview 0
819   while [itemcount <= view_distance and foundobj = black]
820   [
821     set itemcount itemcount + 1
822     ask patch-ahead itemcount [
823        set foundobj pcolor
824        set xview pxcor
825        set yview pycor
826     ]
827
828   ]
```

```
829    update-creature-sightline-position attached_to_creature xview yview
830    ifelse (foundobj = attached_to_colour) ;;Found perceivable colour?
831    [
832      set distance_to_stimuli itemcount
833      set perceived_stimuli foundobj
834    ]
835    [
836      set distance_to_stimuli 0
837      set perceived_stimuli 0
838    ]
839  end
840
841  ;;;
842  ;;; Process Nociceptive, reward and visual sensation
843  ;;;
844  to perceive-world ;;Called by testcreatures
845   let nextobject 0
846   let distobject 0
847   let onobject 0
848   ;; Get color of current position
849   ask patch-here [ set onobject pcolor ]
850   ifelse (onobject = white)
851   [
852      ifelse (noxious_white) ;;is White attached to a noxious stimulus
853      [
854        feed-input-neuron pain_neuron 1 ;;induce Pain
855        if (istrainingmode?)
856        [
857          ;;During training phase move the creature forward to avoid infinite rotation
858          move-creature 0.5 ;;
859          set error_free_counter 0
860        ]
861      ]
862      [
863          feed-input-neuron reward_neuron 1 ;;induce happiness
864          ask patch-here [ set pcolor black ] ;;Eat patch
865      ]
866   ]
867   [
868      ifelse (onobject = red)
869      [
870        ifelse (noxious_red) ;;is Red attached to a noxious stimulus
871        [
872          feed-input-neuron pain_neuron 1 ;;induce Pain
873          if (istrainingmode?)
874          [
```

```
875             ;;During training phase move the creature forward to avoid infinite rotation
876             move-creature 0.5
877             set error_free_counter 0
878           ]
879         ]
880         [
881             feed-input-neuron reward_neuron 1 ;;induce happiness
882             ask patch-here [ set pcolor black ]  ;;Eat patch
883         ]
884       ]
885       [
886         if (onobject = green)
887         [
888            ifelse (noxious_green) ;;is Green attached to a noxious stimulus
889            [
890              feed-input-neuron pain_neuron 1 ;;induce Pain
891              if (istrainingmode?)
892              [
893                ;;During training phase move the creature forward to avoid infinite rotation
894                move-creature 0.5
895                set error_free_counter 0
896              ]
897            ]
898            [
899                feed-input-neuron reward_neuron 1 ;;induce happiness
900                ask patch-here [ set pcolor black ]  ;;Eat patch
901            ]
902         ]
903       ]
904    ]
905    ask visualsensors [propagate-visual-stimuli]
906  end
907
908  ;;;
909  ;;; Move or rotate according to the active motoneuron
910  ;;;
911  to do-actuators ;;Called by Creature
912    let dorotation? false
913    let domovement? false
914    ;;Check rotate actuator
915    ask normalneuron rotate_neuron [
916      if (nlast-firing-time = ticks);
917      [
918        set dorotation? true
919      ]
920    ]
```

```
921   ;;Check move forward actuator
922   ask normalneuron move_neuron[
923     if (nlast-firing-time = ticks);
924     [
925       set domovement? true
926     ]
927   ]
928   if (dorotation?)
929   [
930     rotate-creature 4
931   ]
932   if (domovement?)
933   [
934     move-creature 1
935   ]
936 end
937
938 ;;;
939 ;;; Photoreceptor excitates the connected input neuron
940 ;;;
941 to propagate-visual-stimuli ;;Called by visual sensor
942   if (attached_to_colour = perceived_stimuli) ;;Only produce an action potential
943   ; if the corresponding associated stimulus was sensed
944   [
945      feed-input-neuron attached_to_neuron distance_to_stimuli;
946   ]
947 end
948
949 ;;;
950 ;;; Move insect (#move_units) patches forward
951 ;;;
952 to move-creature [#move_units]
953   if (leave_trail_on?) [Leave-trail]
954   fd #move_units
955 end
956
957 ;;;
958 ;;; Rotate insect (#rotate_units) degrees
959 ;;;
960 to rotate-creature [#rotate_units]
961   rt #rotate_units
962 end
963 breed [sightlines sightline]
964 directed-link-breed [sight-trajectories sight-trajectory]
965
966 ;;;
```

```
967  ;;; Show a sightline indicating the patch the insect is looking at
968  ;;;
969  to-report create-sightline
970    let sightline_id nobody
971    create-sightlines 1
972    [
973       set shape "circle"
974       set size 0.5
975       set color orange
976       set sightline_id who
977       ht
978    ]
979    report sightline_id
980  end
981
982  to update-creature-sightline-position [#creatureid #posx #posy]
983    ifelse (show_sight_line?)
984    [
985      let attached_sightline 0
986      ask testcreature #creatureid [set attached_sightline creature_sightline]
987      ask sightline attached_sightline [setxy #posx #posy]
988      ask sight-trajectories [show-link]
989    ]
990    [
991      ask sight-trajectories [hide-link]
992    ]
993  end
994
995  to attach-sightline-to-creature [#creature_id #sightline_id]
996    let sightline_agent sightline #sightline_id
997    ask sightline_agent [setxy [xcor] of testcreature #creature_id [ycor] of
998     testcreature #creature_id]
999    ask testcreature #creature_id [
1000      set creature_sightline #sightline_id
1001      create-sight-trajectory-to sightline_agent [set color orange set thickness 0.4]
1002    ]
1003  end
1004
1005  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
1006  breed [itrails itrail]
1007  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
1008  ;;;
1009  ;;; Leave a yellow arrow behind the insect indicating its heading
1010  ;;;
1011  to leave-trail ; [posx posy]
1012    hatch-itrails 1
```

```
1013    [
1014       set shape "arrow"
1015       set size 1
1016       set color yellow
1017       set ttl 2000
1018    ]
1019  end
1020
1021  ;;;
1022  ;;; Check if it is time to remove the trail
1023  ;;;
1024  to check-trail
1025    set ttl ttl - 1
1026    if ttl <= 0
1027    [
1028       die
1029    ]
1030  end
1031
1032  itrails-own
1033  [
1034    ttl
1035  ]
1036
1037  globals [
1038    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;SNN Module globals;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
1039    neuron_state_open;; describe state machine of the neuron
1040    neuron_state_firing;; describe state machine of the neuron
1041    neuron_state_refractory;; describe state machine of the neuron
1042    excitatory_synapse
1043    inhibitory_synapse
1044    system_iter_unit ;;time steps of each iteration expressed in milliseconds
1045    plot-list
1046    plot-list2
1047    PulseHistoryBuffSize ;;Size of pulse history buffer
1048    input_value_empty ;;Indicate that there is no input value
1049    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;Insect globals;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
1050    pspikefrequency ;;Number of spikes emitted by an input neuron in response to one stimulus
1051    error_free_counter ;;Number of iterations since the insect collided with a noxious stimulus
1052    required_error_free_iterations ;;Number of error-free iterations necessary to stop training
1053    syneff_down_reg_factor
1054    syneff_min_modulator
1055    syneff_down_reg_decay_rate
1056    plasticity_modulator
1057    max_plasticity_modulator
1058    min_plasticity_modulator
```

```
1059    ;synplast_
1060  ]
1061
1062  ;;;
1063  ;;; Create neural circuit, insect and world
1064  ;;;
1065  to setup
1066    clear-all
1067    RESET-TICKS
1068    initialize-global-vars
1069    random-seed 47822
1070    ;;;;;;;;;;Draw world with white, green and red patches;;;;;;;;
1071    draw-world
1072    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
1073    ;;;;;;;;;;;;;;         Setup Neuron types       ;;;;;;;;;;;;;;
1074    ;;;;;;;;;;;;;;;;Neuron type1:
1075    setup-neurontype 1 -65 -55 0.5 -75 1 -75 0
1076    set-neurontype-learning-params 1 0.09 55 0.09 -35 9 1 8 18;15 ;[
1077     ;#pneurontypeid #ppos_hebb_weight #ppos_time_window #pneg_hebb_weight
1078      ;#pneg_time_window #pmax_synaptic_weight #pmin_synaptic_weight
1079       ;#ppos_syn_change_interval #pneg_syn_change_interval]
1080    ;;;;;;;;;;;;;;;;Neuron type2:
1081    setup-neurontype 2 -65 -55 0.5 -70 1 -70 0;(typeid restpot threshold decayr
1082     refractpot refracttime)
1083    set-neurontype-learning-params 2 0.09 55 0.09 -25 9 1 8 15
1084
1085    ;;;;;;;;;;;;;;;;Neuron type3: (Efficacy Modulator)
1086    setup-neurontype 3 -65 -55 0.2 -70 1 -70 1;(typeid restpot threshold decayr
1087     refractpot refracttime)
1088    set-neurontype-learning-params 2 0.09 55 0.09 -25 9 1 8 15
1089    ;;;;;;;;;;;;;;;;Neuron type4: (Plasticity Modulator)
1090    setup-neurontype 4 -65 -55 0.5 -75 1 -75 2;(typeid restpot threshold decayr
1091     refractpot refracttime)
1092    set-neurontype-learning-params 4 0.09 55 0.09 -25 9 1 8 15
1093    ;;;;;;;;;;;;;;;;Create the Neural circuit (brain) of the insect
1094
1095    ;;;;;;;;;;;;;;Layer 1: Afferent neurons with receptors ;;;;;;;;;;;;;;
1096    setup-normal-neuron 1 15 10  11 1 ;;[layernum pposx pposy pid pneurontypeid]
1097    setup-input-neuron 5 10  1  11 20 1 pspikefrequency ;;[pposx pposy pid
1098     ppostsynneuron psynapseefficacy pcoding pnumofspikes]
1099    setup-normal-neuron 1 15 15  12 1 ;;setup-normal-neuron [pposx pposy pid
1100     pthreshold  prestpot pdecayr prefractpot pintrefrectp]
1101    setup-input-neuron    5 15  2  12 20 1 pspikefrequency ;;[pposx pposy pid
1102     ppostsynneuron ipsynapseefficacy pcoding pnumofspikes]
1103    setup-normal-neuron 1 15 20  13 1 ;;setup-normal-neuron [pposx pposy pid
1104     pthreshold  prestpot pdecayr prefractpot pintrefrectp]
```

```
1105    setup-input-neuron    5 20  3  13 20 1 pspikefrequency ;;[pposx pposy pid
1106     ppostsynneuron ipsynapseefficacy pcoding pnumofspikes]
1107    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
1108
1109    ;;;;;;;;;;;;;;; Layer 2: First hidden layer ;;;;;;;;;;;;;;;;;;;;;;
1110    ;Motoneuron with rotate actuator:
1111    setup-normal-neuron 2 25 14  21 1
1112    setup-normal-neuron 1 25 10  51 4
1113    setup-input-neuron   25 5  4  51 20 1 pspikefrequency
1114    setup-synapse  51  21 20 excitatory_synapse 1 false false false
1115    ;Motoneuron with move actuator:
1116    setup-normal-neuron 2 25 19  22 1
1117    setup-normal-neuron 1 25 23  52 4
1118    setup-input-neuron   25 30  5  52 20 1 pspikefrequency
1119    setup-synapse  52  22 20 excitatory_synapse 1 false false false
1120    ;;;;;;;;;;;;;;; Synapses from Layer 1 to Layer 2 ;;;;;;;;;;;;;;;;;;
1121    ;;Synapse from afferent neuron 1001 to Motoneurons:
1122    setup-synapse   11  21 5 excitatory_synapse 1 true true true
1123    setup-synapse   11  22 5 excitatory_synapse 1 true true true
1124    ;;Synapse from afferent neuron 1002 to Motoneurons:
1125    setup-synapse   12  21 5 excitatory_synapse 1 true true true
1126    setup-synapse   12  22 5 excitatory_synapse 1 true true true
1127    ;;Synapse from afferent neuron 1003 to Motoneurons:
1128    setup-synapse   13  21 5 excitatory_synapse 1 true true true
1129    setup-synapse   13  22 5 excitatory_synapse 1 true true true
1130    ;;;;;;;;;;;;;;;  Layer 3: Output layer ;;;;;;;;;;;;;;;;;;
1131    ;;Actuator move forward:
1132    setup-normal-neuron 3 35 19  31 1
1133    ;;Actuator rotate:
1134    setup-normal-neuron 3 35 14  32 1
1135    ;;;;;;;;;;;;;; Synapses from Layer 2 to Layer 3 ;;;;;;;;;;;;;;;;;;;;
1136    ;;Mutual inhibitory synapses between Motoneurons:
1137    ;setup-synapse  21  22 22 inhibitory_synapse 1 false false false
1138    ;setup-synapse  22  21 8 inhibitory_synapse 1 false false false
1139    setup-synapse  31  21 12 inhibitory_synapse 1 false false false
1140    setup-synapse  32  22 12 inhibitory_synapse 1 false false false
1141    ;;Positive Synapsis from Nociceptive Motoneuron to rotate actuator (no plasticity):
1142    setup-synapse  21  32 11 excitatory_synapse 1 false false false
1143    ;;Positive Synapsis from Reward Motoneuron to move forward actuator (no plasticity):
1144    setup-synapse  22  31 11 excitatory_synapse 1 false false false
1145    ;;;;;;;;;;;;;;;;;;;; Oscillator (Pacemaker) ;;;;;;;;;;;;;;;;;;;;;;
1146    setup-normal-neuron 2 16 25  23 2
1147    setup-normal-neuron 2 22 25  24 2
1148    setup-synapse  23  24 15 excitatory_synapse 2 false false false ;(no plasticity needed)
1149    setup-synapse  24  23 15 excitatory_synapse 3 false false false;(no plasticity needed)
1150    setup-input-neuron 11 25  6  23 20 1 pspikefrequency ;;Voltage clamp to start pacemaker
```

```
1151    ;;Synapse from Pacemaker to Reward Motoneuron:
1152    setup-synapse  23  31 5.5 excitatory_synapse 3 false false false;4.62
1153    ;;;;;;;;;;;;;;;;;;;;Efficacy Modulatory Neuron ;;;;;;;;;;;;;;;;;;;;
1154    setup-normal-neuron 2 30 25  41 3
1155    ;;Synapse from Pacemaker to Efficacy modulator Motoneuron:
1156    setup-synapse  23  41 3.5 excitatory_synapse 3 false false false;3.5;4.62
1157    ;;Inhibitory synapse from reward neuron to efficacy modulator:
1158    setup-synapse  22  41 22 inhibitory_synapse 1 false false false
1159    ;;Mutual inhibitory synapses between Motoneurons:
1160    setup-synapse  31  32 22 inhibitory_synapse 1 false false false
1161    setup-synapse  32  31 22 inhibitory_synapse 1 false false false
1162    ;; Start insect hearth
1163    feed-input-neuron_by_label  6 1
1164    ask patches with [ pxcor = 102 and pycor = 46 ] [set pcolor black]
1165    let creatureid create-creature 102 46 1  5  4  31  32;;[#xpos #ypos
1166     #creature_id #reward_neuron #pain_neuron #move_neuron #rotate_neuron]
1167    ;;;;;;;;;;Create Visual sensors;;;;;;;;;
1168    create-visual-sensor  1 0  white  1 creatureid;[ psensor_id pposition
1169     colour_sensitive attached_neuron attached_creature]
1170    create-visual-sensor  2 0 red  2 creatureid;[ psensor_id pposition
1171     colour_sensitive attached_neuron attached_creature]
1172    create-visual-sensor  3 0 green  3 creatureid;[ psensor_id pposition
1173     colour_sensitive attached_neuron attached_creature]
1174    ;;;;;;;;;;Create Sightline ;;;;;;;;;;;;;
1175    let sightlineid create-sightline
1176    attach-sightline-to-creature creatureid sightlineid
1177    ;; Activate training mode
1178    set istrainingmode? true
1179  end
1180
1181  ;;;
1182  ;;; Set gloval variables with their initial values
1183  ;;;
1184  to initialize-global-vars
1185    set system_iter_unit 1 ;; each simulation iteration represents 1 time unit
1186    set neuron_state_open 1 ;;State machine idle
1187    set neuron_state_firing 2 ;;State machine firing
1188    set neuron_state_refractory 3 ;;State machine refractory
1189    set excitatory_synapse 1
1190    set inhibitory_synapse 2
1191    set plot-list[] ;;List for spike history
1192    set plot-list2[] ;;List for spike history
1193    set PulseHistoryBuffSize 30
1194    set input_value_empty -1
1195    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;Insect globals;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
1196    set pspikefrequency 1
```

```
1197    set error_free_counter 0
1198    set required_error_free_iterations 35000
1199    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;Modulation;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
1200    set syneff_down_reg_factor 1
1201    set syneff_min_modulator 0.01
1202    set syneff_down_reg_decay_rate 0.001
1203    ;;;;Plasticity:
1204    set plasticity_modulator 0.990
1205    set max_plasticity_modulator 0.990 ;1.01
1206    set min_plasticity_modulator 0.01
1207  end
1208
1209  ;;;
1210  ;;; Generate insect world with 3 types of patches
1211  ;;;
1212  to draw-world
1213     ask patches with [ pxcor >= 80 and pycor = 1 and pxcor <= 120 ] [ set pcolor
1214      white  ]
1215     ask patches with [ pxcor >= 80 and pycor = 60 ] [ set pcolor white ]
1216     ask patches with [ pycor >= 1 and pxcor = 80 and pxcor <= 120] [ set pcolor
1217      white ]
1218     ask patches with [ pycor >= 1 and pxcor = 120 ] [ set pcolor white ]
1219     let ccolumns 0
1220     while [ ccolumns < 20 ]
1221     [
1222       set ccolumns ccolumns + 3
1223       ask patches with [ pycor >= 4 and pycor <= 75 and pxcor = 82 + ccolumns * 2
1224        ] [ set pcolor white ]
1225     ]
1226     ask patches with [ pxcor > 80 and pxcor < 120 and pycor > 1 and pycor < 60 ]
1227     [
1228       let worldcolor random(10)
1229
1230       ifelse (worldcolor = 1)
1231       [
1232         set pcolor red
1233       ]
1234       [
1235         if (worldcolor >= 2 and worldcolor <= 4)
1236         [
1237           set pcolor green
1238         ]
1239       ]
1240     ]
1241  end
1242
```

```
1243  ;;;
1244  ;;; Don't allow the insect to go beyond the world boundaries
1245  ;;;
1246  to check-boundaries
1247     if (istrainingmode?)
1248     [
1249        set error_free_counter error_free_counter + 1
1250        if(error_free_counter > required_error_free_iterations)
1251        [
1252         ;set istrainingmode? false
1253        ]
1254        ask testcreatures [
1255          if (xcor < 80 or xcor > 120) or (ycor < 1 or ycor > 60)
1256          [
1257            setxy 102 30
1258          ]
1259        ]
1260     ]
1261  end
1262
1263  ;;;
1264  ;;; Run simulation
1265  ;;;
1266  to go
1267   if (awakecreature?)
1268   [
1269     ask itrails [ check-trail ]
1270     ask visualsensors [ view-world-ahead ] ;;Feed visual sensors at first
1271     ask testcreatures [ perceive-world]; do-actuators]
1272     do-network-dynamics
1273     ask testcreatures [do-actuators]
1274   ]
1275
1276   check-boundaries
1277   tick
1278  end
```

Bibliography

[1] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[2] John Von Neumann. First draft of a report on the edvac. *IEEE Annals of the History of Computing*, 15(4):27–75, 1993.

[3] D. O. Hebb. The Organization of Behaviour. *Organization*, page 62, 1949.

[4] Igor Aleksander, WV Thomas, and PA Bowden. Wisard a radical step forward in image recognition. *Sensor review*, 4(3):120–124, 1984.

[5] Ezequiel A Di Paolo. Evolving spike-timing-dependent plasticity for single-trial learning in robots. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 361(1811):2299–2319, 2003.

[6] Alexandros Bouganis and Murray Shanahan. Training a spiking neural network to control a 4-DoF robotic arm based on spike timing-dependent plasticity. In *Proceedings of the International Joint Conference on Neural Networks*, 2010.

[7] James Humble, Susan Denham, and Thomas Wennekers. Spatio-temporal pattern recognizers using spiking neurons and spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 6(October):84, 2012.

[8] Zafeirios Fountas and Murray Shanahan. A cognitive neural architecture as a robot controller. In *Conference on Biomimetic and Biohybrid Systems*, pages 371–373. Springer, 2013.

[9] Nicolas Frémaux and Wulfram Gerstner. Neuromodulated Spike-Timing-Dependent Plasticity and Theory of Three-Factor Learning Rules. *Frontiers in Neural Circuits*, 9(85):85, 2016.

[10] DA Baxter and JH Byrne. Serotonin-modulated membrane currents in aplysia tail sensory neurons. In *Soc. Neurosci. Abstr*, volume 12, page 765, 1986.

[11] DA Baxter and JH Byrne. Modulation of membrane currents and excitability by serotonin and camp in pleural sensory neurons of aplysia. In *Soc. Neurosci. Abstr*, volume 13, page 1440, 1987.

[12] Georg Hertting and Hanns-Christof Spatz. *Modulation of synaptic transmission and plasticity in nervous systems*, volume 19. Springer Science & Business Media, 2013.

[13] D James Surmeier and Robert Foehring. A mechanism for homeostatic plasticity. *Nature neuroscience*, 7(7):691–2, 2004.

[14] Gina G. Turrigiano and Sacha B. Nelson. Homeostatic plasticity in the developing nervous system. *Nature reviews Neuroscience*, 5(9):97–107, 2004.

[15] Gina Turrigiano. Homeostatic synaptic plasticity: Local and global mechanisms for stabilizing neuronal function. *Cold Spring Harbor Perspectives in Biology*, 4(1), 2012.

[16] Joshua L Lillvis and Paul S Katz. Parallel evolution of serotonergic neuromodulation underlies independent evolution of rhythmic motor behavior. *The Journal of Neuroscience*, 33(6):2709–2717, 2013.

[17] M. Chistiakova, N. M. Bannon, M. Bazhenov, and M. Volgushev. Heterosynaptic Plasticity: Multiple Mechanisms and Multiple Roles. *The Neuroscientist*, 20(5):483–498, 2014.

[18] Simon Haykin. *Neural Networks and Learning Machines*, volume 3. Pearson Upper Saddle River, NJ, USA:, 2008.

[19] Larry F Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain research bulletin*, 50(5):303–304, 1999.

[20] W Gerstner, R Kempter, J L van Hemmen, and H Wagner. A neuronal learning rule for sub-millisecond temporal coding. *Nature*, 383(6595):76–81, 1996.

[21] A L Hodgkin and A F Huxley. A quantitative description of membrane current and its applications to conduction and excitation in nerve. *Journal of Physiology*, 117(1-2):500–544, 1952.

[22] Wulfram Gerstner and Werner M Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[23] E D Adrian. The impulses produced by sensory nerve endings. *The Journal of physiology*, 61(1):49–72, 1926.

[24] Eric R Kandel, James H Schwartz, Thomas M Jessell, Steven A Siegelbaum, and AJ Hudspeth. *Principles of neural science*, volume 4. McGraw-hill New York, 2000.

[25] Simon J. Thorpe, Karl R. Gegenfurtner, Michele Fabre-Thorpe, and Heinrich H. Buelthoff. Detection of animals in natural images using far peripheral vision. *European Journal of Neuroscience*, 14(5):869–876, 2002.

[26] W Maass and C M Bishop. *Pulsed Neural Networks*, volume 275. The MIT Press, 1999.

[27] Donald Olding Hebb. *The organization of behavior: A neuropsychological approach*. John Wiley & Sons, 1949.

[28] Alain Artola and W Singer. Long-term depression of excitatory synaptic transmission and its relationship to long-term potentiation. *Trends in neurosciences*, 16(11):480–7, 1993.

[29] Robert C Malenka and Roger a Nicoll. Long-Term Potentiation A Decade of Progress. *Science*, 285(September):1870–1874, 1999.

[30] John Hertz, Anders Krogh, and Richard G Palmer. *Introduction to the theory of neural computation*, volume 1. Basic Books, 1991.

[31] W. Gerstner, W.M. Kistler, R. Naud, and L. Paninski. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, 2014.

[32] Richard Kempter, W. Gerstner, and J.L. Van Hemmen. Hebbian learning and spiking neurons. *Physical Review E*, 59(4):4498–4514, 1999.

[33] Henry Markram, Joachim Lübke, Michael Frotscher, and Bert Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. *Science*, 275(5297):213–215, 1997.

[34] G Q Bi and M M Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 18(24):10464–10472, 1998.

[35] Guo-qiang Bi and Mu-ming Poo. Synaptic Modification By Correlated Activity: Hebb's Postulate Revisited. *Annual Review of Neuroscience*, 24:139–166, 2001.

[36] P. J. Sjostrom, Ede a Rancz, Arnd Roth, and M. Hausser. Dendritic Excitability and Synaptic Plasticity. *Physiological Reviews*, 88(2):769–840, 2008.

[37] S Song, K D Miller, and L F Abbott. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience*, 3(9):919–926, 2000.

[38] Wulfram Gerstner and Werner M. Kistler. Mathematical formulations of Hebbian learning. *Biological Cybernetics*, 87(5-6):404–415, 2002.

[39] Abigail Morrison, Markus Diesmann, and Wulfram Gerstner. Phenomenological models of synaptic plasticity based on spike timing. *Biological Cybernetics*, 98(6):459–478, 2008.

[40] Natalia Caporale and Yang Dan. Spike timing-dependent plasticity: a Hebbian learning rule. *Annual review of neuroscience*, 31:25–46, 2008.

[41] Chong Liu and Jonathan Shapiro. Implementing classical conditioning with spiking neurons. In Joaquin Marques de Sa, LuisA. Alexandre, Wlodzislaw Duch, and Danilo

Mandic, editors, *Artificial Neural Networks - ICANN 2007*, volume 4668 of *Lecture Notes in Computer Science*, pages 400–410. Springer Berlin Heidelberg, 2007.

[42] Lovisa Irpa Helgadottir, Joachim Haenicke, Tim Landgraf, Raul Rojas, and Martin P. Nawrot. Conditioned behavior in a robot controlled by a spiking neural network. In *International IEEE/EMBS Conference on Neural Engineering, NER*, pages 891–894, 2013.

[43] André Cyr and Mounir Boukadoum. Classical conditioning in different temporal constraints: an stdp learning rule for robots controlled by spiking neural networks. *Adaptive Behavior*, page 1059712312442231, 2012.

[44] Xiuqing Wang, Zeng Guang Hou, Feng Lv, Min Tan, and Yongji Wang. Mobile robots' modular navigation controller using spiking neural networks. *Neurocomputing*, 134:230–238, 2014.

[45] Cristian Jimenez-romero, David Sousa-rodrigues, and Jeffrey H Johnson. A Model for Foraging Ants , Controlled by Spiking Neural Networks and Double Pheromones. *IEEE: UK Workshop on Computational Intelligence*, pages 1–7, 2015.

[46] Joachim Haenicke, Evren Pamir, and Martin Paul Nawrot. A spiking neuronal network model of fast associative learning in the honeybee. *Frontiers in Computational Neuroscience*, (149), 2012.

[47] Ívan Petrovítsj Pavlov and GV Anhrep. *Conditioned reflexes: an investigation of the pysiological activity of the cerebral cortex*. Oxford University Press, 1927.

[48] Claudia Clopath, Lars Büsing, Eleni Vasilaki, and Wulfram Gerstner. Connectivity reflects coding: a model of voltage-based STDP with homeostasis. *Nature Neuroscience*, 13(3):344–352, 2010.

[49] H Markram, W Gerstner, and PJ Sjöström. Spike-timing-dependent plasticity: a comprehensive overview. *Front Synaptic Neurosci*, 4:8, 2012.

[50] Kenneth D. Miller and David J. C. MacKay. The Role of Constraints in Hebbian Learning. *Neural Computation*, 6(1):100–126, 1994.

[51] Kenneth D Miller. Synaptic economics: competition and cooperation in synaptic plasticity. *Neuron*, 17(3):371–374, 1996.

[52] C H Bailey, M Giustetto, Y Y Huang, R D Hawkins, and E R Kandel. Is heterosynaptic modulation essential for stabilizing Hebbian plasticity and memory? *Nature reviews. Neuroscience*, 1(1):11–20, 2000.

[53] Sébastien Royer and Denis Paré. Conservation of total synaptic weight through balanced synaptic depression and potentiation. *Nature*, 422(6931):518–522, 2003.

[54] Christopher M Lee, Carl Stoelzel, Marina Chistiakova, and Maxim Volgushev. Heterosynaptic plasticity induced by intracellular tetanization in layer 2/3 pyramidal neurons in rat auditory cortex. *The Journal of physiology*, 590(10):2253–2271, 2012.

[55] Friedemann Zenke, Guillaume Hennequin, and Wulfram Gerstner. Synaptic plasticity in neural networks needs homeostasis with a fast rate detector. *PLoS Comput Biol*, 9(11):e1003330, 2013.

[56] Jen-Yung Chen, Peter Lonjers, Christopher Lee, Marina Chistiakova, Maxim Volgushev, and Maxim Bazhenov. Heterosynaptic plasticity prevents runaway synaptic dynamics. *The Journal of Neuroscience*, 33(40):15915–15929, 2013.

[57] Gregg A Phares and Texas Medical. Heterosynaptic Modulation of Synaptic Efficacy. *Life Sciences*, pages 1–9, 2001.

[58] J Dudel and S W Kuffler. PRESYNAPTIC INHIBITION AT THE CRAYFISH Neuromuscular junction. *Journal of Physiology*, 155:543–562, 1961.

[59] Zhonghui Guan, Joung-Hun Kim, Stavros Lomvardas, Kerri Holick, Shiqin Xu, Eric R Kandel, and James H Schwartz. p38 map kinase mediates both short-term and long-term synaptic depression in aplysia. *The Journal of Neuroscience*, 23(19):7317–7325, 2003.

[60] JC Eccles, Rosamond M Eccles, and F Magni. Central inhibitory action attributable to presynaptic depolarization produced by muscle afferent volleys. *The Journal of physiology*, 159(1):147, 1961.

[61] John H Byrne and Eric R Kandel. Presynaptic facilitation revisited: state and time dependence. *Journal of Neuroscience*, 16(2):425–435, 1996.

[62] Stephane Marinesco and Thomas J Carew. Serotonin release evoked by tail nerve stimulation in the CNS of aplysia: characterization and relationship to heterosynaptic plasticity. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 22(6):2299–2312, 2002.

[63] Jonathan Levenson, John H Byrne, and Arnold Eskin. Levels of serotonin in the hemolymph of aplysiaare modulated by light/dark cycles and sensitization training. *The Journal of neuroscience*, 19(18):8094–8103, 1999.

[64] D Johnston, SH Williams, R Gray, and RE Fisher. Cholinergic and noradrenergic modulation of long-term potentiation in hippocampal ca3 neurons. In *Brain Signal Transduction and Memory*. Academic Press, Inc San Diego, CA, 1989.

[65] J. E. Bradler and G. Barrionuevo. Heterosynaptic correlates of long-term potentiation induction in hippocampal CA3 neurons. *Neuroscience*, 35(2):265–271, 1990.

[66] Marina Chistiakova and Maxim Volgushev. Heterosynaptic plasticity in the neocortex. *Experimental brain research*, 199(3-4):377–390, 2009.

[67] Xiaohui Xie and H Sebastian Seung. Learning in neural networks by reinforcement of irregular spiking. *Physical Review E*, 69(4):041909, 2004.

[68] Dorit Baras and Ron Meir. Reinforcement learning, spike-time-dependent plasticity, and the bcm rule. *Neural Computation*, 19(8):2245–2279, 2007.

[69] Nicolas Frémaux, Henning Sprekeler, and Wulfram Gerstner. Functional requirements for reward-modulated spike-timing-dependent plasticity. *The Journal of Neuroscience*, 30(40):13326–13337, 2010.

[70] Eugene M Izhikevich. Solving the distal reward problem through linkage of stdp and dopamine signaling. *Cerebral cortex*, 17(10):2443–2452, 2007.

[71] Zuzanna Brzosko, Wolfram Schultz, and Ole Paulsen. Retroactive modulation of spike timing-dependent plasticity by dopamine. *Elife*, 4:e09685, 2015.

[72] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[73] Jonathan Baxter and Peter L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

[74] Danilo Jimenez Rezende and Wulfram Gerstner. Stochastic variational learning in recurrent spiking networks. *Frontiers in computational neuroscience*, 8(April):38, 2014.

[75] Q Gu. Neuromodulatory transmitter systems in the cortex and their role in cortical plasticity. *Neuroscience*, 111(4):815–835, 2002.

[76] Wolfram Schultz. Getting formal with dopamine and reward. *Neuron*, 36(2):241–263, 2002.

[77] Michael E Hasselmo. The role of acetylcholine in learning and memory. *Current opinion in neurobiology*, 16(6):710–715, 2006.

[78] Paolo Calabresi, Barbara Picconi, Alessandro Tozzi, and Massimiliano Di Filippo. Dopamine-mediated regulation of corticostriatal synaptic plasticity. *Trends in neurosciences*, 30(5):211–219, 2007.

[79] María Esther Olvera-Cortés, Patricia Anguiano-Rodríguez, Miguel Ángel López-Vázquez, and José Miguel Cervantes Alfaro. Serotonin/dopamine interaction in learning. *Progress in brain research*, 172:567–602, 2008.

[80] E E Steinberg, R Keiflin, J R Boivin, I B Witten, K Deisseroth, and P H Janak. A causal link between prediction errors, dopamine neurons and learning. *Nat Neurosci*, 16(7):966–973, 2013.

[81] Johanni Brea and Wulfram Gerstner. Does computational neuroscience need new synaptic learning paradigms? *Current Opinion in Behavioral Sciences*, 11:61–66, 2016.

[82] M.F. Bear, B.W. Connors, and M.A. Paradiso. *Neuroscience:*. Neuroscience: Exploring the Brain. Lippincott Williams & Wilkins, 2007.

[83] Cristian Jimenez-Romero and Jeffrey Johnson. Spikinglab: modelling agents controlled by spiking neural networks in netlogo. *Neural Computing and Applications*, pages 1–10, 2016.

[84] Andres Upegui, C. A. Peña Reyes, and Eduardo Sanchez. An FPGA platform for on-line topology exploration of spiking neural networks. *Microprocessors and Microsystems*, 29(5):211–223, 2005.

[85] Ankur Gupta and Lyle N Long. Hebbian learning with winner take all for spiking neural networks. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pages 1054–1060. IEEE, 2009.

[86] A. Cyr and M. Boukadoum. Classical conditioning in different temporal constraints: an STDP learning rule for robots controlled by spiking neural networks, 2012.

[87] Seth Tisue and Uri Wilensky. Netlogo: A simple environment for modeling complexity. In *International conference on complex systems*, pages 16–21. Boston, MA, 2004.

[88] John Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[89] Gary Bradski and Adrian Kaehler. The opencv library. *Doctor Dobbs Journal*, 25(11):120–126, 2000.

[90] Lego - mindstorms. http://mindstorms.lego.com. Online; accessed 12-September-2015.

[91] Ip camera adapter. http://ip-webcam.appspot.com/. Online; accessed 03-December-2015.

[92] Stefano Fusi. Hebbian spike-driven synaptic plasticity for learning patterns of mean firing rates. *Biological Cybernetics*, 87(5-6):459–470, 2002.

[93] Stefano Fusi and L F Abbott. Limits on the memory storage capacity of bounded synapses. *Nature Neuroscience*, 10(4):485–493, 2007.