UNIVERSITY of York

Department of Electronic Engineering

BEng Project Report 2017/2018

Student Name: Yudong Lu

Project Title: Neural Network, Brains and Neural Coding

Supervisors: Dr. David Halliday & Dr. Yuriy Zakharov

Department of Electronic Engineering
University of York
Heslington
York
YO10 5DD

Abstract

The human brain is a highly complex system whose working principle has not yet been fully understood. This project tries to find a method to imitate the neural network in human brains. Three major biological principles are investigated: spiking neural networks (SNN), spiking-time dependent plasticity(STDP) and dopamine-modulated STDP. An SNN controller for a robot is constructed based on these principles and corresponding simulation experiments are conducted for its performance analysis.

In further detail, a base line spiking neural network of six fully connected neurones is constructed to control a simulated 2D robot which can learn how to collect rewards more efficiently in a simulated environment. Several synaptic weight change methods have been encoded into the above SNN which can be selected to form different SNN configurations of method combination.

A simulation environment is constructed to support simulation experiments in graphical animation style. The implemented components include sensors, motor dynamics, robot position and facing angle computation algorithms, and light sources. All these components and the SNN are integrated into a PYTHON program which can be run repeatedly with different configurations and parameter settings.

Simulation experiments were conducted under different configurations and parameter settings. Results data was collected and the performance is compared between different configurations. Some interesting robot behaviours have been observed in these experiments. It is shown that simulated biological phenomena can be useful in solving autonomous navigation problems, and spiking neural network robots controlled with dopamine modulated STDP are capable of solving temporal difference reinforcement learning tasks.

The Statement of Ethics

Considering this whole project is completely based on the software simulation and after consideration of the University's code of practice and principles for good ethical governance no ethical issues were identified in this project.

Acknowledgement

First of all, I would like to present my sincere thanks to Dr. David Halliday who mentored me as my first project supervisor and provided immense help with his vast knowledge in neuroscience. His assistance guided me in the right direction and prevented me from making countless mistakes.

I am very grateful to Dr. Yuriy Zakharov, my second project supervisor who has given much valuable advice on report writing, such as how to present the contents in a clear and understandable way.

The Brian2 development team has provided a very helpful Python library which I used to construct spiking neural networks and consequently saved me precious time from having to make low-level programs.

Thanks to Dr. Marcel Stimberg who was very kind and patient with answering my questions about using the Brian2 library.

Many thanks to my friends Ryan van Wensveen and Jack Mckeown, who helped to proof read this final report. I also have gained many grammar suggestions from them.

Finally, I would like to thank my father Professor Huanzhang Lu. During my formative years, he shared the importance of the fundamentals of mathematics and other fascinating aspects of electronics. Unsurprisingly, I studied Electronics as my major in my undergraduate degree due to my inherent familiarity with it. I have gained plenty of transferable skills and problem solving strategies through his guidance during the previous three years, which has positively contributed to allowing me to complete my own research for this project fluently.

Contents

List of Figures

1 Introduction

The autonomous navigation of mobile robots is widely used in various military systems and civilian equipment. Traditionally, these mobile robots mostly rely on explicit instructions that allow them to only perform repetitive, simple tasks that are limited to a small fraction of situations. Since mathematically modelling an unstructured, unknown environment for mobile robots is very difficult, an artificial intelligence approach to solving these non-linear systems is highly in demand in the modern industry. Human brains are intricate systems composed of billions of inter-connected neurones that enable high-level behaviours, such as facial recognition, logical reasoning and emotional expression. In the light of this, through modelling various features in the human brain, an artificial intelligence approach to solving complex problems is theoretically possible. This brings us to the goal of this project: to solve an autonomous navigational problem with a robot using a biologically-inclined method.

1.1 Project Aim

In further detail, the aim of this project is to investigate whether is possible to use Spiking Neural Network (SNN) combined with spikes-time dependent plasticity (STDP) and reinforcement learning to control a successful phototactic robot in a simulated two-dimensional space. Specifically, a simulated robot controlled by a neuronal controller will be used to solve a distal reward problem, that is kind of autonomous navigation problem in which the reward usually is received with a delay after a sequence of behaviours is performed. All the modelling will be constructed completely in Python programming language. Based on this, several simulations will be conducted to investigate the main properties of SNN and reinforcement learning.

1.2 Project Implementation

As mentioned earlier, the project implementation revolves around the imitation of the human brain. Since the brain is highly complex, three major aspects involved with the brain's learning have been selected, which are:

- The cross-communication between neurones.
- The plasticity of the inter-neuronal synapses.
- The brain's natural reward system (pleasure).

Regarding the first point, real neurones communication through time-dependent spikes and the third generation neural network Spiking Neural Network is sharing some similarity to real neurones. Compared to actual neuronal behaviour, the spiking neurones characteristics in computation and communication assimilate spatial-temporal data from pulse coding [5]. Spiking neurones can express information of time and space, so it is more suitable for an actual dynamic environment than the Neural Networks that using continuous signals.

In address of the second point, another feature of real neural networks is the plasticity of the synapses strengths in human brains that makes the networks with the same structures can present a wide variety of completely different behaviours due to the differentiation of synaptic strengths [3]. This feature provides an unlimited potential to the neural network if the network size is big enough. Biologically-observed principle of Spike-Timing Dependent Plasticity refers to the simple rule that the synaptic strengths should be dynamically changeable with the precise temporal order and time interval between presynaptic and postsynaptic neurones firing [7]. Luckily, this principle can be easily expressed numerically by two decaying exponentials, which can largely reduce the implementation complexity and meanwhile provide a high biological plausibility for this neural robot controller.

Finally, it is important to note that neural networks in human brains can generate pleasures through dopamine neurones [10], which can affect the neural network synapse strengths globally and give the system a sense of what behaviour can result in a positive outcome or negative outcome [23]. The phenomenon is very similar to reinforcement learning which using reward mechanism to help the decision making in order to optimise its overall behaviour in long term. Izhikevich presented a solution for distal reward problem through linkage of STDP and dopamine signalling [18] this simulated dopamine effect share a lot of features with temporal difference learning rule in reinforcement learning. In the light of this, the spiking neural network can be implemented with a property of temporal difference learning rule. This will be detailed in the following chapters.

1.3 Summary of Tasks Undertaken

Base on the above implementation concepts, the three main contributions that have made through simulating the model are documented below.

- Constructed a base line spiking neural network of six fully connected neurones which can be used as a controller for a simulated 2D robot. This SNN controlled robot can learn how to collect rewards more efficiently in a simulated environment. Several synaptic weight changing methods have been encoded into the above SNN which can be selected to form different SNN configurations of method combination.

- Constructed a simulation environment to support simulation experiments in graphical animation style. The implemented components include sensors, motor dynamics, robot position and facing angle computation algorithms, and light sources. All these components and the SNN can be integrated into a PYTHON program which can run repeatedly with different configurations and parameter settings.

- Conducted simulation experiments under different configurations and parameter settings. Results data was collected and the performance compared between different configurations. Some interesting robot behaviours have been observed in these experiments. It is shown that simulated biological phenomena can be useful in solving autonomous navigation problems, and spiking neural network robots controlled with dopamine modulated STDP are capable of solving temporal difference reinforcement learning tasks.

1.4 Report Structure

To read this report, a basic understanding of differential equations and biological aspects of the model is needed.

In Chapter 2, a literature review is provided. This chapter begins with a introduction of previous generations of artificial neural networks, theoretical models can be used to model this project and previous relevant works.

In Chapter 3, the methodology of how the models introduced in chapter 2 can be used to model a phototactic robot will be discussed.

Results of the simulations and discussions will be present in chapter 4. In this chapter 4, some interesting behaviours and its learning performance will be discussed in details.

The end of the report followed with a conclusion. The outcome of implementing this projects including the nature of SNN, noise injected in SNN, stability of the network, reinforcement learning and the capability of learning will be documented in chapter 5.

In addition, the future work chapter will focus on the how to solve some issues in this project and it is in chapter 6.

2 Literature Review

In this chapter, a literature review regarding this project is provided. A comparative overview of three generations of neural networks (NNs) will be presented in order to emphasise the necessity of using the spiking neural network. Several approachable spiking neuronal models, STDP and the dopamine modulated STDP will be introduced. The potential problem will occur during the implementation will also be discussed and its solutions will be presented as well.

2.1 Background Knowledge

There are more than ten billion neurones in human brains, which are interconnected forming a highly complex biological architecture. Nowadays, neural scientists still cannot understand this complex architecture well enough to replicate it. However, the basic concept of how the neural networks work has been understood and many attempts to imitate the neurone structures behaviour have been successful.

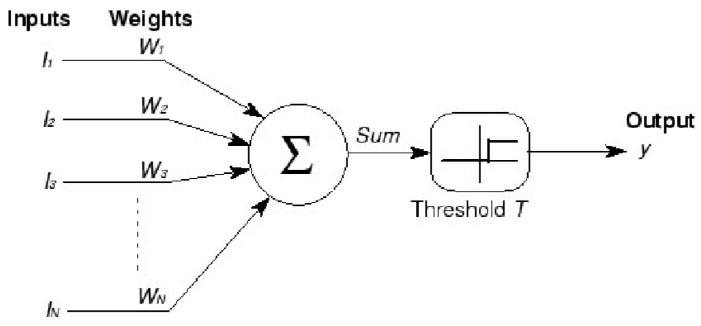

Figure 1: McCulloch and Pitts neuronal model [8], base on the net input to the unit neurone, a binary number is outputted via a Heaviside function.

In 1943, McCulloch and Pitts presented a highly simplified model of a neurone in their paper "A logical calculus of the ideas immanent in nervous activity" [8]. This well- known first generation neurone model computes the sum of the products of weights and inputs, and generates a binary output '1' or '0' depending on whether the sum is greater than the threshold, this operation is called Heaviside function. The McCulloach and Pitts neural model is shown in figure 1, where the net input is computed in the neurone model, in which the output of the neurone unit processes by the Heaviside function is a binary number.

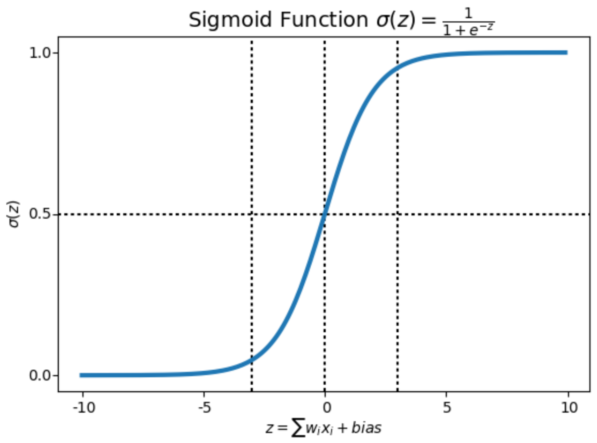

Figure 2: Activation function [33], with the aid of sigmoid function, the output from the unit neurone will be the probability of being '1'.

Instead of using Heaviside function, the second generation neural network uses the continuous activation function as its to process the net input. As shown in figure 2, the activation function outputs the probability of being '1' [33] instead of a binary number. Classic second generation NNs include Feed-Forward NNs; Recursive Sigmoidal NNs and Radial Basis Function (RBF) NNs [5]. Compared to the first generation NNs, the second generation NNs can not only use less neurones to implement logical models, but also can take analogue signals for inputs and outputs. The first two generation NNs have been widely successful applications in many different areas of industry, such as pattern recognition, classification and system control.

Unfortunately, previous generation NNs can only process continuous signals as the input, they are still not very similar to the behaviour of real neurone networks. Neurones in human brains communicate with each other by spikes trains, a neurone accumulates the incoming spikes from other neurones which alters the voltage of its cell membrane [17]. When the voltage value reaches the threshold voltage of the neurones, the neurones will release a spike (fires) then enter a brief refractory period in which the input spikes will be ignored. The spikes then travel through the most complex part of the neurone, the synapse. As depicted in figure 3, this is the connection between two neurones and can be compared to the signal pre-processors used in a computer, and the figure shows how the neurotransmitters are diffuse from presynaptic side to postsynaptic side via the synapse. When signals arrive at the presynaptic side of the synapse, which works as a neurone cell membrane, a channel of information carriers known as neurotransmitters are released. The neurotransmitters then diffuse through the synapse and are absorbed by the postsynaptic side, resulting in postsynaptic potential [34]. A neurone may take inputs from many synapses and be accumulated to the voltage potential of the cell membrane, and when the membrane voltage potential reaches the threshold, the neurone will fire. It is necessary to mention that synapses have plasticity, or the ability to locally adjust the connection strength among synapses over time [14]. It is widely known that plasticity is one of the fundamental principles of the learning process for Neural Network in the human brain. Moreover, neuroscientists also discovered that brains can generate pleasure to reward animal behaviour, ultimately optimising its behaviour in the long term [9].

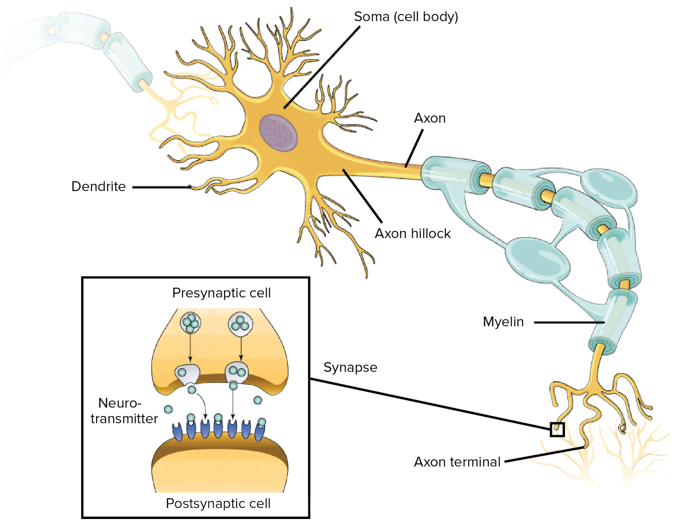

Figure 3: Real neurone and its synapse, synapse which connect two neurones and the figure shows how the neurotransmitters travel through the synapse from presynaptic cell to postsynaptic cell. [34]

During the late twentieth century, SNNs were developed as the third generation of neural network systems [9]. They usually use several differential equations, rather than activation functions to model the behaviour of neurones. These more detailed mathematical models represent higher biological plausibility, which means the neurones encode information using individual spike times, much like real neurones. This key feature allows SNNs to exhibit properties that enable them higher performance in applications requiring higher computational power, or where the timing information is important to the problem solving [5]. This is the model we will focus on, as it not only imitates the general architecture of real Neural Networks, but also to imitate the communication behaviour of real Neural Networks.

2.2 Theoretical Models to Implement an Intelligent Agent

In the above section, a brief introduction has been made to previous generation NNs and three major working principles of the Neural Networks in brains were given. The features of SNNs and their advantages in imitating the behaviour of brains are discussed. In this section, several theoretical models probably used to implement an intelligent agent will be discussed in details.

2.2.1 Neuronal Models and Model Selection

There are several mathematical models that can be applied as the neuronal model for SNNs. In this section, the three most common spiking neuronal models will be introduced in order of complexity, starting with the most complex model.

The Hodgkin-Huxley Model (HH) is a conductance-based model used to describe some highly biologically meaningful features. This model is based on Hodgkin and Huxley's experimental observation of the giant axon of a squid, where they discovered there are three different types of ion flows, namely sodium, potassium and a leak current which passes through the neurone cell membrane. The Hodgkin-Huxley equation presents a mathematical way of describing how the current flows through the cell membrane and charges the membrane voltage over time [14].

$$C\frac{dV}{dt} = -(g_{Na}m^3h(V - E_{Na}) + g_k n^4(V - E_k) + g_L(V - E_L)) + I(t) \tag{1}$$

Where V is membrane voltage, C is membrane capacitance and $I(t)$ is external current. g_{Na}, g_K are conductance values of sodium and potassium. g_L is a voltage-independent conductance which is used to describe the leakage channel. m, h, n are dimensionless parameters between 0 to 1 which can be treated as the probability that a channel is open. E_{Na}, E_k, E_L are the reversal potentials.

The Hodgkin-Huxley Model can be easily expressed numerically but is not very suitable for large-scale simulations due to its relatively higher computational complexity compared to other neuronal models [15].

The Izhikevich Model is much simpler than the Hodgkin-Huxley model, yet it still presents good biological plausibility. Izhikevich introduced bifurcation methodologies to omit some biophysical details from the Hodgkin-Huxley neuronal models in the form of ordinary differential equations [12].

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I \tag{2}$$

$$\frac{du}{dt} = a(bv - u) \tag{3}$$

With the auxiliary after-spike resetting, as shown in the equation (4)

$$\text{if v} \geq \text{30mV; then} \begin{cases} v = c \\ u = u + d \end{cases} \tag{4}$$

Where v is membrane potential, u is membrane recovery variable and I is synaptic current. a, b, c and d are dimensionless parameters.

The variable u accounts for the activation of potassium ionic current and inactivation of sodium ionic current, this variable provides negative feedback to membrane voltage v. Compared to the Hodgkin-Huxley, the Izhikevich Model not only presents similarly detailed neuronal behaviours but also exhibits better computational efficiency, thus it is widely used for the simulation of spiking neural networks on a large-scale [12].

The Integrate-and-Fire model (IF) [14] is one of the most frequently used models for neuroscience which is significantly simplified compared to previously discussed models. This model can be described using a simple RC circuit, where the current in the neurone consists of two parts. The current leaking out of the capacitor through the resistor and the current injected into the capacitor from the input.

$$C_m \frac{dv}{dt} = -\frac{V(t) - E_m}{R_m} + I \tag{5}$$

$$\text{if } V \geq V_{threshold}\text{; then } V = E_m$$

Where C_m is membrane capacitance, v is membrane potential and R_m is membrane resistance. E_m is equilibrium potential. If we assume the injected current is constant over time, then the equation (5) can be solved as shown in the equation (6).

$$V(t) - E_m = R_m I[1 - exp(-\frac{t}{C_m R_m})] \tag{6}$$

The term $C_m R_m$ is membrane time constant. According to the equation (6), if the injected current is constant and RI is greater than the threshold voltage, the neuron will fire periodically. The equation (5) and (6) can also describe the integrate and fire neuron in the form of an electric circuit as shown in figure 3.1.

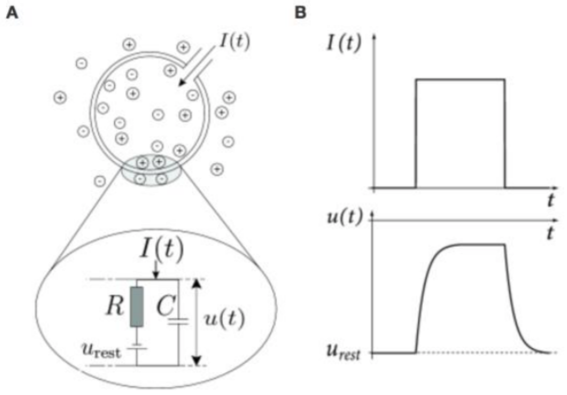

Figure 4: the cell membrane can be modelled as a capacitor in parallel with a resistor, charged by a current $I(t)$. The RC circuit is in line with a power supply of potential $V_{threshold}$. B, a current pulse (top) with a smooth voltage trace(bottom) [14]

The integrated-and-fire neurone is the simplest neuronal model, so it shows less biological meaningful features than previous two discussed neurones. However, because IF model is very simple, it is very popular for the simulations which care more about the behaviour of the network as whole. Since in this project, we care more about the network behaviour as a controller, the IF model is the most suitable model for this project [14].

2.2.2 Spike-Time-Dependent Plasticity (STDP)

Plasticity in the brain is widely believed to be a property by which the learning process takes place and information is stored in the brain. STDP is a biologically-observed principle, which indicates that the synaptic strength should be dynamically changeable with the precise temporal order and time interval between presynaptic and postsynaptic neurones firing. When a presynaptic neuronal firing followed by a postsynaptic neuronal firing, this results in a long-term potentiation. As depicted in figure in figure 5, the curve above the x-axis shows that the long-term potentiation will cause a weight value increasing. If a postsynaptic neuronal firing followed by a presynaptic firing, a long-term depression is occurred that will decrease the synaptic weight and it can be shown as the curve under the x-axis in figure 5. In this project, we need the STDP for training the network and the STDP can be expressed numerically, as shown in the equation (7) [15].

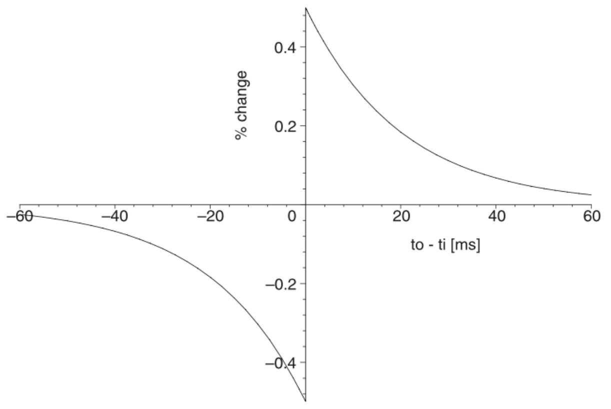

Figure 5: The STDP time window. the percentage and direction of weight value change are determined by the inter-spike time between presynaptic and postsynaptic spikes. [6]

$$F(\Delta t) = \begin{cases} A_{+}exp(-\frac{\Delta t}{\tau_{+}}) \text{ if } \Delta t \leq 0 \\ -A_{-}exp(\frac{\Delta t}{\tau_{-}}) \text{ if } \Delta t \geq 0 \end{cases} \tag{7}$$

Where Δt or $(t_2 - t_1)$ is the time of postsynaptic firing t_2 minus the time of presynaptic firing t_1. $F(\Delta t)$ is the weight change between pre and postsynaptic neuron, A_+ and A_- are the absolute maximum weight changing value when Δt is close to zero [6].

In practice, the weight value will be manually limited to a certain range in case the weights do not increase to infinite. However, this causes some problems as noted and solved in Paolo's paper. A weights value is constrained within the range [0,1] [6]. In this case, if weights have no damping, the simulation result will show that the weight always falls in a bimodal distribution, which means most weights have the value of the minimum or maximum. To avoid this problem, directional damping should be added to the equation (7), this can slow down the weights, increasing or decreasing when it is close to the boundary value. The directional damping can be expressed mathematically in the equation (8) .

$$F(\Delta t) = \begin{cases} A_+(W_{max} - W)exp(-\frac{\Delta t}{\tau_+}) \text{ if } \Delta t \leq 0 \\ -A_-(W - W_{min})exp(\frac{\Delta t}{\tau_-}) \text{ if } \Delta t \geq 0 \end{cases} \tag{8}$$

Where W is the current weight value, W_{max} and W_{min} are the maximum and minimum weights value respectively. Using this equation (8), a local dynamic neurones interaction can be simulated in software without explicit function, since it is composed of two decaying exponentials.

2.2.3 STDP and Network Stability

The core concept of STDP is when a presynaptic neurone fires followed by a postsynaptic neurone firing, the synapse strength from the presynaptic neurone to the postsynaptic neurone firing will be increased according to the time interval between these two firings. This implies that in the future these two neurones are more likely to fire in this sequence, and further implies that the synaptic strength between these two neurones will be further increased and more likely to be increased with a greater magnitude than previous times. This process works like a positive feedback loop and also means this process will quickly increase the synaptic strength from the presynaptic neurone to postsynaptic neurone to infinity, the time interval between two neuronal firings will also towards zero [37].

In any kind of system, it is always necessary to avoid positive feedback loops which can cause serious system stability problems. To avoid this situation, we need to study more biological features of real neurones. Firstly, real neurones have its natural mechanism which is refractory period [14]. Refractory periods are a short time interval following a firing or action where another firing or action cannot be generated. There are two types of refractory periods, one is the absolute refractory period where it is completely unlikely for another firing or action potential to be triggered, no matter how strong the stimulus is. Another refractory period is the relative refractory period where the membrane voltage is dropped down to a lower level potential, then the neurone is less likely to fire again unless the stimulus is strong enough. In human muscle fibre, the absolute refractory periods are normally between 2.2 and 4.6 ms. The refractory periods can prevent the firing time interval between a presynaptic neurone to a postsynaptic neurone towards zero [14].

However, the synaptic strength is still very likely to be increased towards infinity, thus we need to constraint the weight valued in a range that will not affect the network operations. The directional damping in the section 2.2.2 can do this job well, but synaptic weights can be quickly increased or decreased to its maximum or minimum values, a more systematic approach to avoid stability problems is needed when the neurones are connected to form a large size of network [6].

In Turrigiano's research shows that except the local mechanism of long-term potentiation and depression in which synaptic weights can be altered in a synapse-specific manner, there are an other mechanism, in which all synaptic weights can be changed globally as a function of firing rate of population of neurones [35]. This mechanism has the effect of when the firing rate of a population of the neurones is dramatically increased or is very high, then all of the synapse weights are decreased. When the firing rate of population of neurones is dramatically decreased or is very low, then all of then synapse weights are increased. This phenomenon we know as activity-dependent scaling (ADS) [35].

In a connected network with a high firing rate, the synaptic weights can be easily increased to a very high level through STDP, however, a higher value of synaptic weights can lead to an even higher firing rate. The global effect mechanism can prevent this problem, as the ADS is the negative feedback mechanism that can be used to neutralise the positive feedback mechanism of STDP.

Another consequence of this mechanism is that it can foster competition behaviours. For instance, if a group of neurones are representing two opposite behaviours and neurones for behaviour A have a high activity, whereas the neurones corresponding to behaviour B have a lower activity. Then, a reducing of all the synaptic weights will be triggered due to the high firing rate, which leads to further decreasing the activity of behaviour B. Since synaptic weights can be increased faster with higher firing rate through STDP, the neurones corresponding to behaviour A will become more activated, meanwhile, the behaviour B will be weakened. This process will be continued until only neurones for behaviour A are active [35].

2.2.4 Reinforcement Learning and Pleasure in Brains

As we discussed earlier, the natural reward system of brain (pleasure) share some similarity with reinforcement learning. The pleasure in human brains can be treated as a kind of reward system, the object of pleasure gives us a concept of what acts of behaviour are good or bad [10]. Seung presented a hypothesis that it is possible there exists "hedonistic synapses" in brains which respond to reward signals by changing their actions to pursue this reward [4]. However, our STDP model alone does not act like this as it has no concept of what has a positive or negative effect on its purpose.

Generally speaking, reinforcement learning is a process that strengthens the actions coincided with a desired good result. To do this, there are two things necessary to clarify. Firstly, the reward, how to calculate transmissions and secondly the actions, what is the action concerned and how to coincide them with the reward [26].

For this project, a model of combining a SNN and reinforcement learning is used to solve autonomous navigation problems. The reward is defined as light sources and the robot will be controlled by the neuronal controller to perform a sequence of behaviours in order to attain the reward. As the reward is received with a delay after a sequence of behaviours are performed, this kind of autonomous navigation problem is called distal reward problem. After the robot received the reward, this information of receiving reward should be broadcast to every synapse in the neural network and be used to strengthen the ones whose pre and post neurones firing patterns, this is because the purpose of the SNN is to generate a positive result by changing the synaptic weights. These are the actions at work here.

2.2.5 Temporal Difference Learning

A discussion regarding temporal difference learning is necessary, as we mentioned before, Izhikevich presented a notion of using dopamine modulated STDP to solve distal reward problem that is theoretically possible and is sharing some similarity with temporal difference learning. *Temporal difference learning* is the mostly used reinforcement learning rule for policy evaluation. Temporal difference learning methods are usually considered of combining two procedures, *Dynamic Programming* and *Monte Carlo Methods* [30].

Dynamic Programming is a divide and conquer approach which breaks down a relatively large problem into subproblems and use the stored solution of subproblems to solve the large problem, and is typically applied to optimisation problems that can be decomposed into subproblems. Since this method relies on perfect models of the environment [30].

Monte Carlo Methods are using randomness approach to solve the problem that might be deterministic in principle. The method repetitively takes a random sampling from experienced data in form of sequences of state-action-reward-samples that can predict future rewards. Thus, an environment is not required to be modelled completely, but this algorithm can only update the actions after a complete sequence available when the final state is reached [30].

Temporal difference learning methods are the method that learns directly from the experience, therefore, there is no need for a complete model of the environment, but the actions can be updated at every state of the incremental procedure. In temporal difference learning methods, there are a cost function reports back the distance between the predicted reward and actual reward is received at any time step [30]. The main job of the algorithm is to consistently adjust the parameters to minimise this cost function, this can help to make a strategy for the systems to gain more rewards in the future [26].

In neuroscience fields, researchers found that dopamine neurones work in a similar manner like temporal difference learning. Matsumoto and Hikosaka have shown a experiment via a monkey's dopamine cells [36]. In the beginning, the dopamine cells increased its firing rate when the monkey obtained its reward, this shows the difference in expected and actual rewards. After the monkey is trained properly, the fire rate was not increased when the monkey obtained its reward. This biological mechanism is very similar to temporal difference learning, thus we can apply temporal difference learning to model dopamine cells in brains, then a high biological realistic intelligent model can be created.

2.2.6 Eligibility Traces

In order to model temporal difference learning in spiking neural network, we need a mechanism enabling the SNN to remember previous actions. In Spiking Neural Networks, behaviours are encoded in neurones firing patterns, so we need to memorise the neurones firing patterns corresponding to its presynaptic and postsynaptic neurones firing order until the reward is received, so as to implement reinforcement learning. STDP is kind of mechanism triggered by inter-spike time and firing orders between presynaptic and postsynaptic neurones, an modulated the STDP described by an exponential decaying function can be used as eligibility traces.

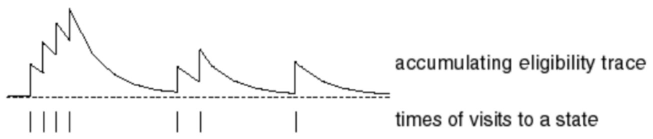

Figure 6: Eligibility traces used in reinforcement learning, when the state is visited by impulse input, the eligibility trace then be accumulated and it will always decays to zero exponentially [37].

Generally speaking, some reinforcement learning use lookup tables to memorise all the information, such as the Q-table used in Q Learning Algorithms. For Spiking Neural Networks, there are many synapses behaviours need to be memorised, it is very hard to implement in a form of table [26].

As shown in figure 6, the eligibility trace is a parameter used to memorise system behaviours, in which the times when the state is triggered are marked as a spikes, and every time the incoming spike leads to an accumulating of eligibility trace. The eligibility trace will eventually decay to zero, as the action happened a long time ago plays negligible effect [1]. Leak and Integrate circuit can be used to implement this memorising mechanism for each synapse which is usually called eligibility trace in reinforcement learning, it will integrate the input changes and decay exponentially with time when there are no weight changes injected. The output of this circuit is the mixture of previous neurones firing pattern, in which neurones firing patterns near the reward have a greater impact than the earlier ones. All these eligibility traces represent the whole behaviours of the network needed.

With aid of eligibility trace, the temporal difference method can be significantly sped up, since there is no need to revisited the previous actions in a lookup table [2].

2.2.7 Dopamine Modulated STDP

As we mentioned the importance of eligibility trace in the above section, a mathematical approach to modulate STDP into eligibility traces and how it will be used to manipulate synaptic weights will be discussed in this section.

In reinforcement learning, the reward is more likely to be received after a delay when a sequence of actions are performed, this is called distal reward problem. Izhikevich has found that dopamine modulated STDP can be used to model many aspects of temporal difference learning and is very suitable to solve distal reward problems [18]. Since the details of the kinetics of the intracellular process activated by dopamine and STDP are not fully understood, the dopamine modulated STDP is the simplest phenomenological model which captures the essence of dopamine modulation of STDP. The modulation can be done by simply adding several differential equations to STDP, in which the state of each synapse is described by two phenomenological variables: Synaptic weight s, and c which describes a relatively slow process on synapses just like eligibility trace. These two variables can be described by equation (9) and (10) respectively [18].

$$\frac{dc}{dt} = -\frac{c}{\tau_c} + STDP(\tau)\delta(t - t_{pre/post}) \tag{9}$$

$$\frac{ds}{dt} = cd \tag{10}$$

The presynaptic and postsynaptic firing order can be presented by this impulse function $\delta(t - t_{pre/post})$ and the step increase of c have the amount of $STDP(\tau)$, that is shown in equation (7). The value τ is the time interval between presynaptic and postsynaptic firing which determines the magnitude of step increase of the amount of STDP. The variable c will decay to zero exponentially with the time constant τ_c, the time constant have to be carefully set, as it determines how many previous behaviours the c has to record. The amount of weight changes at every time step is the product of current value of c and dopamine concentration d.

d is the variable that describes the dopamine concentration, and it is the same for all synapses in the network. This can be shown in equation (11).

$$\frac{dd}{dt} = -\frac{d}{\tau_d} + DA(t)\delta(t_{reward}) \tag{11}$$

where the dopamine concentration will also decay to zero exponentially with a relatively shorter time constant τ_d. The DA is the step-increase magnitude of dopamine concentration when the system receives the reward and the $\delta(t_{reward})$ impulse function constrains this step-increase to be added to the variable d only at the time when the reward is received.

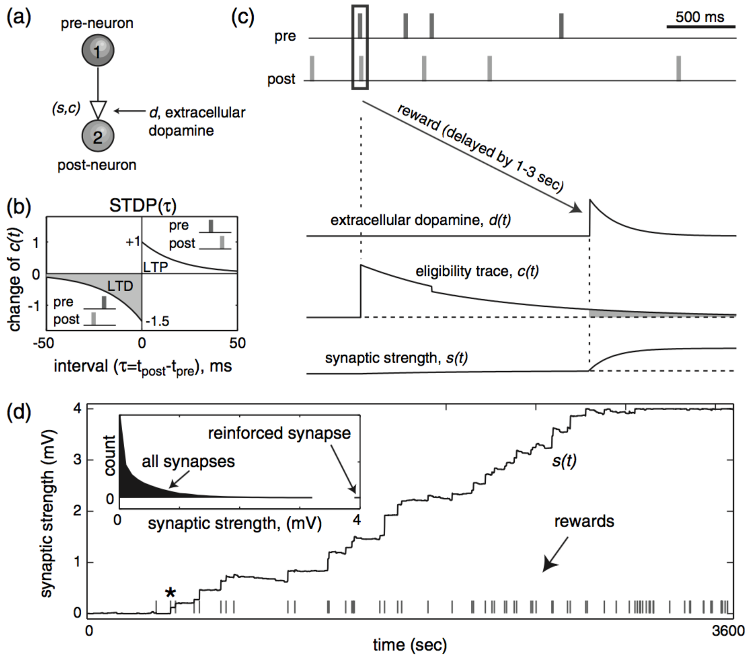

Figure 7: (a) shows the two dynamic variables on the synapse between two neurones, c and d. (b) shows the STDP window which determines the magnitude of step increase in c. (c) shows two decaying functions of dopamine concentration and eligibility trace of $c(t)$ and when the delayed reward is received, the synaptic strength is increased in function of the product of current value of c and d. (d) shows after the synaptic strength is reinforced, more rewards are received [18].

Izhikevich did a simulation of dopamine modulated STDP between two neurones. As shown in figure 7, how the delayed reward reinforce the synaptic strength. The eligibility trace does not change the synaptic strength, it only records the neuronal behaviours before the reward is received. When the reward is received, the value of c shows the weighted average of previous neuronal behaviours, in which earlier neuronal behaviours will have less impact than recent behaviours. This simulation looks artificial in context, but it provides a clear explanation of the dopamine modulation of STDP and the idea of this simulation was inspired from the monkey experiment mentioned in section 2.2.5 [18].

Applying dopamine modulated STDP, we can enable the mobile robot have a sense of reward and it is valuable for training of the Spiking Neural Network. Several parameters have to be carefully adjusted for the network we are using. Time constant τ_d and τ_c determine the decaying speed of variable d and c, they have to be set in a reasonable range as it will cause the learning too fast or too slow. The extracellular dopamine concentration DA also determines the network learning rate, more importantly, it can also represent the error value of cost function in temporal difference learning. Thus, DA should be decreased when the frequency of receiving rewards is increased and when the frequency of receiving rewards is decreased the DA should be increased back, that means when the system is properly trained and more rewards are being collected, the system should slow down the learning and when the system is not effectivity of receiving rewards anymore it should reaccelerate the learning rate.

In this chapter, the model for the spiking neurones, STDP and using ADS to maintain network stability are discussed. More detailed discussion regarding reinforcement learning, temporal difference learning and dopamine modulated STDP are provided, due to these theories are the core part of the training method. In the following section, a comparative discussion regarding three relevant previous work will be provided. Through studying these relevant work, some useful implementation and design strategies are applied to the methodology of this project.

2.3 Previous Work

There is a wide variety of research have been conducted into using Spiking Neural Network for robot controlling. Spiking Neural Network are very suitable for robot controlling or problems in a dynamic environment, as it can convey time-dependent information into spike trains. Researchers also point out that Spiking Neural Network can be implemented in hardware easily because the spiking signals are very similar to digital signals in electronic systems, hardware such as Field Programmable Logic Array (FPGA) are very suitable for this task [5].

The simplest solution to design an SNN for robot controlling can be done by explicitly wire up the synapses and set the weight values between neurones. For example, Lewis used this approach to control an artificial leg using only four neurones [38]. However, for more complex networks, this is impossible to be done or it is hard to tell whether the wiring is in its optimal state. Various approach regarding the weights training for SNN has been used, few typical ones are outlined below.

2.3.1 SNN Training using STDP

Xiuqing and Zeng-Guang have conducted an experiment using SNN for an obstacle avoiding mobile robot trained by STDP learning principle [5]. The mobile robot has ultrasonic sensors which can detect the distance to obstacles and feed this as the input to SNN. The network has a feed-forward architecture and output signals to control the angular velocities of robot's wheels. The simulation results show that the robot can be trained to avoid obstacle successfully. However, this method is limited to goal oriented tasks in which the robot have to perform some actions first before receiving the reward. In Xiuqing and Zeng-Guang' simulation, the robot relies on the action and response happened at the same time, that means when the sensor detects an obstacle is in front of the robot, the network will control the robot to perform some actions and at the same time the sensor will no longer detect any obstacles are in front of the robot.

2.3.2 Evolutionary Technique and Genetic Algorithm

Paolo designed a method which trained a successful phototactic robot using genetic programming [6]. In this simulation, the robot and a light source are placed in a two-dimensional environment, STDP learning rule is applied to the robot and the SNN has a fully connected architecture except for self-connection. Similarly, the robot has the same mechanism for distance detection sensors and wheels controller neurones like Xiuqing and Zeng-Guang' work [5]. The genetic algorithm is used to find the best parameters from a population of possible solutions, these parameters are the weights values and the sensor parameters. Paolo found that a model with a more complex plasticity rule is easier to be fit for this task, where a model with STDP principle has better fitness than a model with no plasticity. However, this approach requires a large number of robots and it is restricted to the problem that can only have one robot.

2.3.3 TD Learning Approach using Reward Modulated STDP

Chorley and Seth implemented a Braitenberg vehicle using reward modulated STDP [21]. The network has a relatively large size compare to the examples we discussed above, in which 1000 neurons are used. They have shown that it is possible to perform reinforcement learning via dopamine modulated STDP, and the robot can be trained to collect rewards at an adequate frequency. However, their work relies on explicit wiring for appreciated behaviour. As shown in figure 8, the left sensor is wired to the right motor and the right sensor is wired to the left motor.

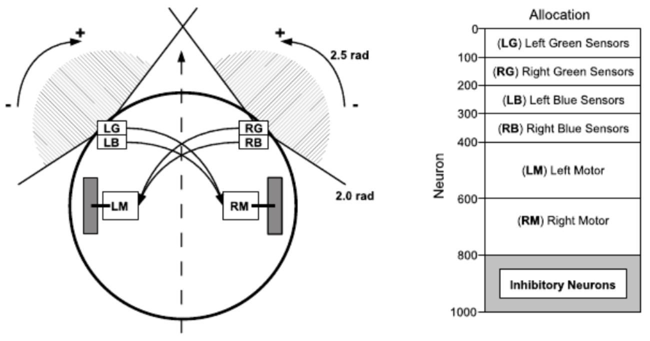

Figure 8: Chorley and Seth' Braitenberg vehicle and its architecture, the network is explicitly wired up for desired behaviours [21].

3 Methodology

In this chapter, the implementation details of this project will be outlined. A phototactic vehicle controlled by a Spiking Neural Network using reward modulated STDP will be simulated in a two dimensional environment. The architecture of the SNN is very simple, in which only six neurones are used and they are fully connected except self-connection, integrated and fire neuronal model is selected for this network and these ideas are cited from Paolo's work. The rewards in this environment are multiple simulated light sources, the SNN synapse weight values were not explicitly set for desired behaviours, the robot should be able to learn what is the good behaviour for collecting more rewards through random motion.

3.1 Robot Simulation

The robot has two sensors and two wheels. Two neurones will take inputs from each of the sensors, and the two corresponding neurones output signals to make each of the wheels move forward or backward respectively. Neurones were set to fire randomly to ensure the robot moves enough that it enables the robot to better explore the environment.

3.1.1 Robot Sensors

The simulated sensors are photoreceptors that read the light intensity corresponding to its current position, the light intensity is in inverse proportion to the square of the distance between the robot and the light source from its initial intensity value (4000). The product of captured light intensity and the sensor gain (60) is treated as the sensor input to the corresponding neurones, sensor neurones generated spike trains from the given signals using a Poisson process by linearly transforming the input value into the firing rate, the maximum rate is limited to 200 Hz.

The left and right sensors have an angle of acceptance of 60°, the angle between sensors is 18° [6].

3.1.2 Robot Motors

The output spike trains from motor neurones can be directly translated into velocities for left and right wheels.

$$\tau_{mot} = \frac{dM_{L,R}}{dt} = -M_{L,R} + M_G(\sum \delta(t - t_{forw}) - \delta(t - t_{back})) \tag{12}$$

where M_G is the motor gain has a value of 5, $\delta(t - t_{forw})$ and $\delta(t - t_{back})$ are the forward spike and backward spikes, the sum of these spikes in time interval τ_{mot} (30ms) determines the sign of the velocity for incrementing. $M_{L,R}$ stores the current motor values for left and right motors respectively [6].

3.1.3 Robot Velocity and Orbiting Behaviour in Coordinate System

In this simulation, the robot velocity is the resultant velocity of left and right velocities. This means if the left and right velocities are not equal, the robot will have an orbiting behaviour. To simulate this, the robot needs to have three variables, angle of facing, left and right velocities. The equation below shows a mathematical approach of how to update the robot position and the angle it faces in every time step through its current angle of facing, left and right velocities.

$$\begin{cases} \Delta\theta = \theta + \theta_{Max}(V_{right} - V_{left}) \\ \Delta x = x + cos(\theta)\frac{V_{right}+V_{left}}{2} \\ \Delta y = y + sin(\theta)\frac{V_{right}+V_{left}}{2} \end{cases} \tag{13}$$

Where θ, x and y are the current robot facing angle and position in coordinate, θ_{Max} is the maximum angle robot can turn in every time step. V_{right} and V_{left} are left and right velocities output from robot motor [39].

3.2 Environment Setup

As shown in figure 9, the environment has 14 light sources are plotted as blue dots, and they are arranged in a rectangular shape. The reason for using 14 light sources is for making the compromise between few rewards is hard to allow the robot to learn and too many rewards result in chaos. The rectangular shape of rewards arrangement is used to empty the centre part of the environment. The robot is then to be expected to avoid this area as much as possible. The robot is plotted as the black dot and its sensors are plotted as yellow lines. When the robot sensors can detect the light sources, it will only take the intensity of light from the closest light source. As shown in the figure 9 the green line shows which light source the robot is detecting.

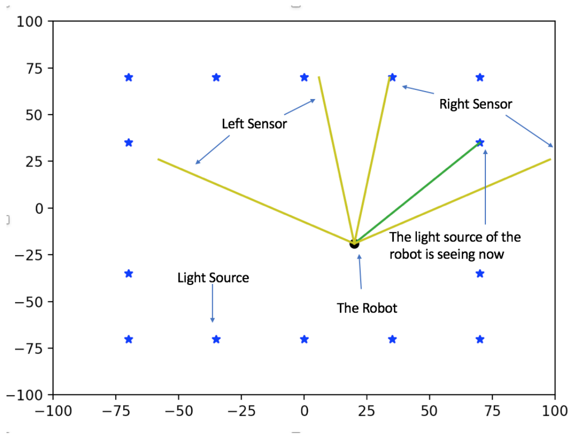

Figure 9: Simulation Environment with one robot and 14 light sources.

Figure 10, shows the desired behaviour of the robot that when the right sensor detects the light source, the robot is turning right to approach the light source. the black line shows the path taken by the robot for 100ms.

When the robot reaches a light source, the dot of the light source is moved to the empty position and the coordinate of this light source become the empty position. In this simulation, the initial empty position is (-70, 0). The environment has boundary effect, when the robot reaches the boundary of the environment it will turn until it is not moving towards the boundary.

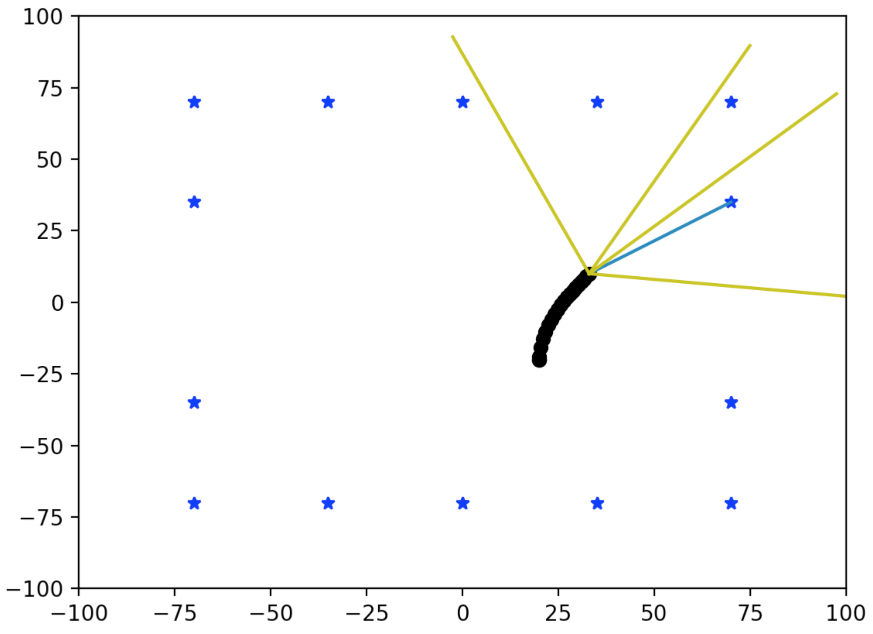

Figure 10: The desired robot behaviour, when the right sensor detect signals the associated behaviour is turning right.

3.3 Spiking Neuronal Controller

In this section, a detailed explanation regarding the neuronal model and architecture of the spiking neuronal controller will be provided. The parameters adjusting and its meaning of the dopamine modulated STDP will be discussed.

3.3.1 Integrate and Fire Neuronal Model

Integrate and Fire neuronal model is used for this project, due to our attention is on the network behaviours rather than individual neurone and the IF model is the simplest neuronal model. Paolo presented an IF neuronal model with the excitatory and inhibitory reversal potentials, in this project, the original formula has been slightly modified, as only the excitatory neurones are used in this project.

$$\tau_m \frac{dV}{dt} = V_{rest} - V + g_{ex}(t)(E_{ex} - V) \tag{14}$$

where τ_m is the membrane time constant and it has the value of 10ms. V is the current membrane voltage and the rest voltage V_{rest} is -70mV, the excitatory reversal potential E_{ex} is 0mV. When the membrane voltage V is greater than the threshold voltage which is -50mV, the neurone will release a spike and its membrane voltage will be reset to V_{rest} immediately. The absolute refractory time of these neurones is 4ms, which limits the maximum firing rate of a signal neurone to be 250 Hz [6].

In real neurones, neurones membrane voltage can be increased by accumulating spikes from other neurones, this can be modelled in a simple way. When a spike arrives at the postsynaptic neurone n_j from a presynaptic neurone n_i, the value of g_{ex} of n_j is increased by the current value of the synaptic weight: $g_{ex}(t) \to g_{ex}(t) + w_{ij}(t)$. The excitatory conductance itself is a decaying exponential function as shown in equation(15).

$$\tau_{ex} \frac{dg_{ex}}{dt} = -g_{ex} \tag{15}$$

Where τ_{ex} is the decaying time constant which has a value of 5ms [6].

3.3.2 Network Architecture

The network used for this project is consisted by six neurones and they are fully connected except self-connection. As shown in figure 11, the architecture of the neural network used in this project is presented. Six neurones are allocated to each robot component, the Poisson spike trains transformed from the left and right sensors are fed into neurone two (n2) and neurone three (n3) respectively. Neurone n0 and n1 output spike trains to control left and right motor respectively for moving forward and n4 and n5 output spike trains to control left and right motor respectively for moving backward. The output spike trains from motor neurones can be directly translated into velocities for left and right wheels using equation 12 discussed in above section [6]. This neural network architecture was cited from Paolo's work [6], in which he had used this network structure and genetic programming implemented a very successful phototactic robot. In this project, we not only need a simple network for the convenience of implementation but also need the confidence that a network is possible to allow a robot to perform reward-attractive behaviours, thus, the network in Paolo's work is an idea choice.

There are 30 synapses in this network, and all the weights are dynamic changeable through dopamine modulated STDP.

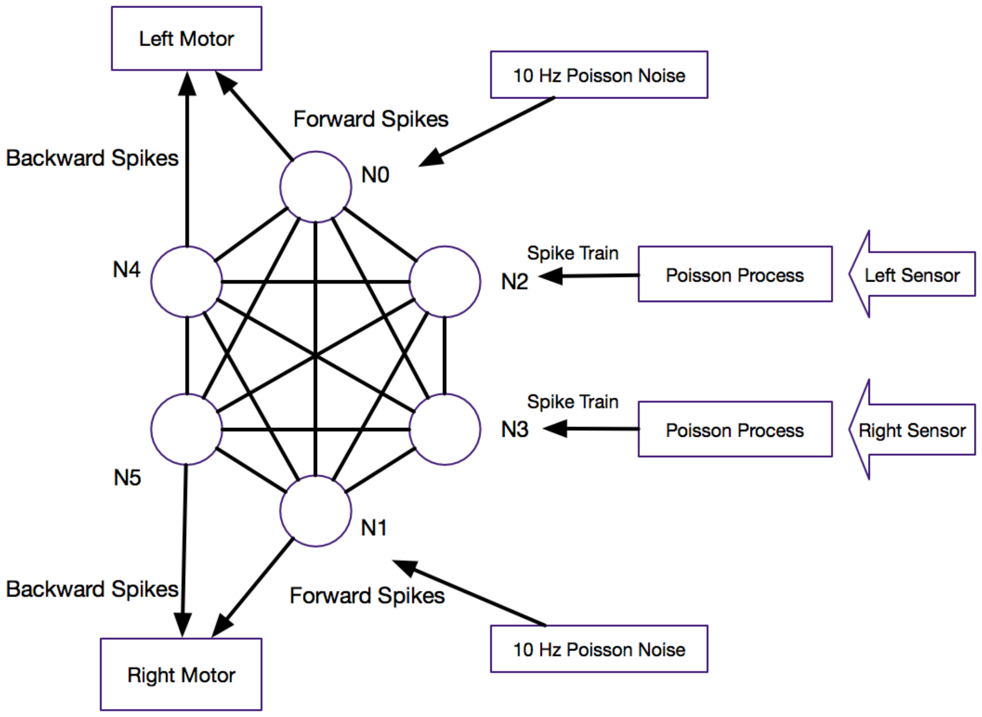

Figure 11: The Overview of Network Architecture, six neurones are representing six different robot components. Four neurones are motor neurones, two neurones are sensor neurones. The network is fully connected except self-connection.

3.3.3 Network Stability and Globally Controlled Activity-Dependent Scaling

There is a drawback of using this network which is the six neurones are fully connected that is possible to result in a network chaos. Because STDP is basically a positive feedback mechanism, a negative feedback mechanism should be applied to keep the system in its stable state. As shown in figure 12, the simulation of this network structure will have stability problem if there is no algorithm to maintain the system stability. At around 5800 ms, most of the weights are saturating at maximum values, this causes there is no any differentiation between weight values and no further learning is possible, due to the weight values are stuck at maximum values.

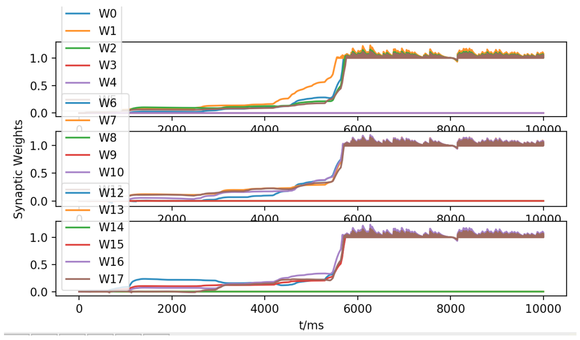

Figure 12: Unstable system without the Globally Controlled Activity-Dependent Scaling, weight values are saturating at maximum value after around 5800ms.

As we mentioned earlier, ADS is the global negative feedback mechanism observed from the real neural network which has the effect of decreasing all weights when the firing rate of the population of the neurones is dramatically increased or is very high. When the firing rate of the population of neurones is dramatically decreased or is very low, then all of the synapse weights are increased.

In this project, a naive algorithm of modelling this mechanism is designed. It was proved that it can maintain the network stability as well as foster competitive behaviours. The algorithm to express the key principle of ADS is expressed as follow.

Algorithm 1 Globally Controlled Activity-Dependent Scaling

```
1:  procedure ACTIVITY-DEPENDENT SCALING PROCEDURE
2:      $Clock = 0\ ms$
3:      $t = 0$
4:      $Spikes = 0$
5:  while True:
6:      if $Clock \leq 10\ ms$ then:
7:          $Spikes += \sum_{n=1}^{n=6} \delta(t - t_{firing})$
8:      else:
9:          $Clock = 0$
10:         $Spikes = 0$
11:     if $Spikes \geq Rate_{Max}$ then:
12:         $W -= 0.05$
13:         $W = clip$(W, 0 ,1).
14:     $Clock += 0.1ms$
15:     $t += 0.1ms$
```

where the algorithm shows that if the accumulated firing rate from six neurones ($\sum_{n=1}^{n=6} \delta(t - t_{firing})$) is exceed the maximum acceptance rate $Rate_{Max}$ in every 10 ms, the synaptic weights vector W will subtract 0.05. The command $W = clip$(W, 0 ,1) means the weight values will be constraint in a range from 0 to 1.

Through serval control experiments, the value of $Rate_{Max}$ is set to be 90. This value can allow the network have not only a reasonable activeness but also a good stability.

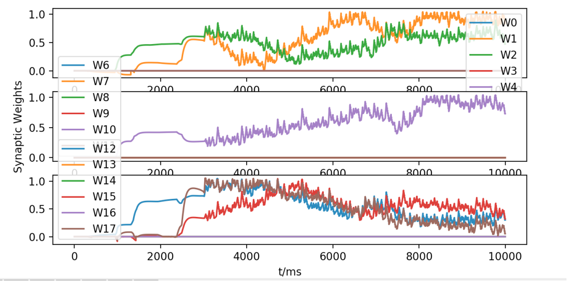

Figure 13: Stable system with the Globally Controlled Activity-Dependent Scaling, weight values are no longer saturating at maximum value.

As shown in figure 13, the naive algorithm of globally controlled activity-dependent scaling is applied to the system, and the synaptic weights are no longer stuck at its maximum value. It is also notable that only a few synaptic weights have non-zero values, that is because only a few synapses can make contributions to the rewards-attracting behaviours.

3.3.4 STDP and Eligibility Trace

When the system can be keep in a stable state, our concentration moved on how to allow the network to learn. The STDP principle applied in this project do not adjust the synaptic strength directly or immediately, it is only used to record the actions performed by the robot. Thus, the modulated STDP can be treated as eligibility trace that should be an exponential decay function and it only used to alter the synaptic weight when the reward has received. This works like a filter, in which eligibility traces record actions that robot have performed and when the reward is received the change of weight value Δw is the product of the current value of eligibility trace and dopamine value. So only the rewarded weights change representing actions with a contribution toward the light source can really be remembered

Through several control experiments, the time constant of τ_+ and τ_- in equation 7 are both set to be 25ms, the A_+ has a value of 0.025 and A_- is equal to $1.5A_+$. As shown in figure 14, this leads to the area under the curve of long term depression is greater than long term potentiation. This implies that the uncorrelated firings between neurones will be more likely to decay the synaptic strength to zero, and the correlated firings tend to increase the associated weight values over time. In general, the robot will be able to learn the correct behaviours from random movements with some reasonable noise resistance, due to the uncorrelated behaviours are easier to be forgotten.

As we mentioned earlier, STDP can be modelled easily, as it can be represented by two exponential decaying functions. As shown in figure 15, how the weight value between two neurones is modified through STDP. In this simulation, the presynaptic neurones are set to fire at 10ms and postsynaptic is set to fire at 20ms. When the presynaptic neurone fires, a step increase is added to the presynaptic exponential function, When the postsynaptic neurone fires the synaptic strength from the presynaptic neurone to the postsynaptic neurone is increased by the current value of presynaptic exponential function. It is notable that a negative step increase is added to the postsynaptic exponential function when the postsynaptic neurone is firing, if there is an uncorrelated firing occurs after the postsynaptic firing, the synaptic

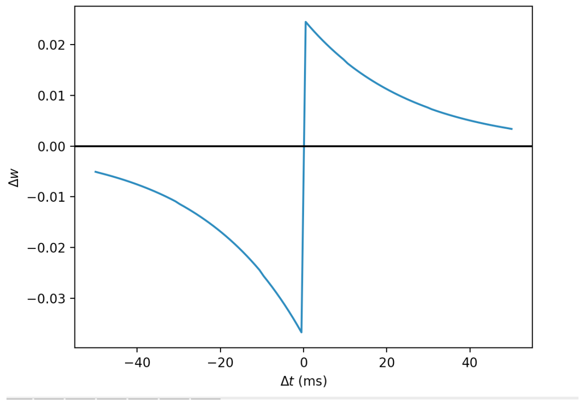

Figure 14: STDP model used in this project, in which the long term depression has greater area under the curve

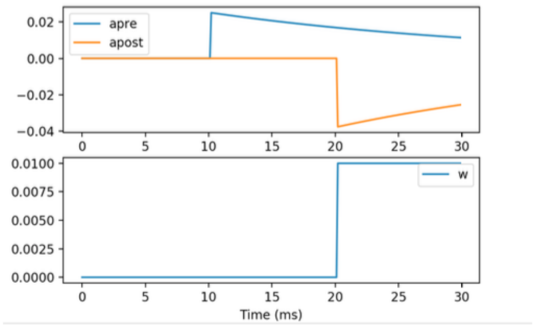

Figure 15: STDP modelling with two exponential decaying functions

weight will be decreased to zero immediately, as A_- is greater than A_+.

However, the STDP together with ADS present a poor performance when dealing with distal reward problems, due to their conflicting natures. STDP is a positive feedback mechanism; it will lead to dramatic increase of weight values. However, every time, if the change of weight value is too big, ADS will force it to decrease. Thus, the weight values will be stuck at a certain range. This results in no further learning and the current weight values are not good enough to allow the robot to collect rewards effectively. This can be seen in Figure 16, which shows that before 4000 ms, the robot is only using STDP and all the weight values have a very small increase. After 4000ms, several weights values increase dramatically, which triggers ADS and causes fluctuations at a fixed point.

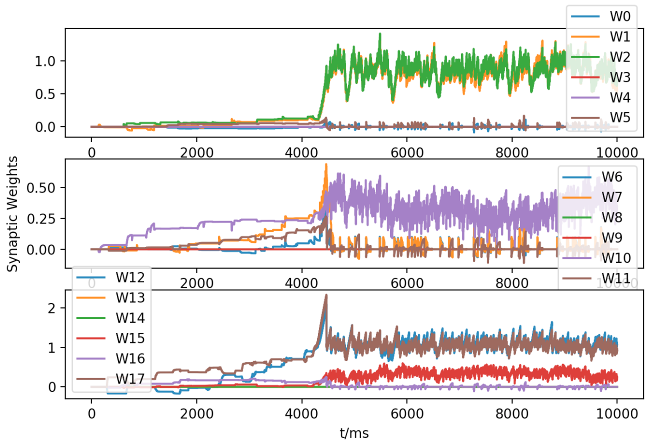

Figure 16: Synaptic weights of the neural robot controller with only STDP is applied

3.3.5 Dopamine Modulated STDP Simulation and Implementation

Dopamine modulated STDP is the core part to allow the robot to be able to learn, which is simulating the dopamine in real neurones to imitate the temporal difference reinforcement learning rule. The robot usually needs to perform series of actions to obtain the reward. These actions are memorised by the eligibility trace (modulated STDP) and it drops off exponentially, due to the time constant τ_c in equation (9) is 500ms, the step increase in eligibility trace will drop to negligible around after1000ms. If the distance between the robot and the target is roughly about 50 units, the motor setting can allow the robot to reach the light source in from 150ms to 300ms if the robot is performing a correct orbiting behaviour. The SNN will change its synaptic weights only when the robot reaches its reward. With these setting, the robot will only learn from the recent actions, as the easier actions are filtered out.

As shown in figure 17, the simulation results of dopamine modulated STDP between two neurones (N1, N2) is presented. In the beginning, the N1 fired at 1ms and N2 fired at 3ms, the inter-spike time is 2 ms and the eligibility trace is jumped to 0.055 from zero. 198ms later, a reward is received and dopamine concentration is increased to 0.45 immediately and decay to zero with a time constant of 100ms. The ΔS is the product of the current value of eligibility trace and the dopamine concentration, thus synaptic strength from N1 to N2 increased rapidly at the beginning and then the gradient dropdown to zero when dopamine concentration is dropped down to zero and the synaptic weight no longer increases.

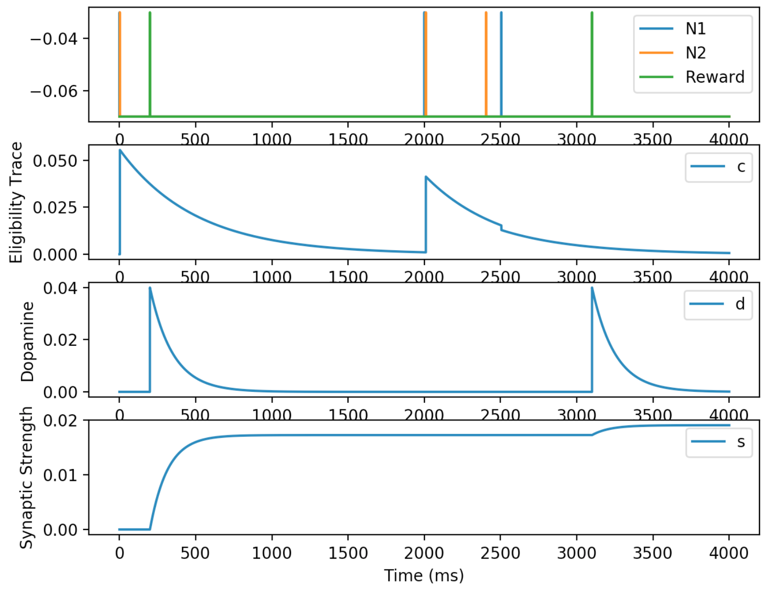

Figure 17: Simulation of dopamine modulated STDP between two neurones.

From this example, we can see both the magnitude of the step increase for dopamine and time constant τ_d determine the learning rate of the model. If the magnitude of the step increase of dopamine is greater, the synaptic weight will have more rapid increase at the beginning, and greater time constant τ_d will allow the synaptic weight have a longer time to increase, as the dopamine decays slower. These parameters then determine the learning rate of dopamine modulated STDP.

At 2000ms, the second example shows when the inter-spike time is greater (10ms), there is smaller step increase in eligibility trace. At the time from 2400ms to 2500ms, an uncorrelated spikes occur, this results in a step negative increase in the eligibility trace. The reward is received at 3000ms, roughly delayed about 1000ms and it shows a negligible increasing of synaptic weights occurs with this much delay.

This simulation is assuming the action will lead to obtaining rewards. However, our actual robot simulations show that it is possible some learned behaviour has a very poor performance to collect rewards, as the random signals are injected to motor neurones, and its possible to have uncorrelated spike pairs. To deal with this we can add negative baseline to the dopamine equation (equation 11) and that will allow the robot to develop some opposite behaviours if it is unable to receive any rewards in a relatively long time. The modified equation 11 with baseline is shown below.

$$\frac{dd}{dt} = \frac{(baseLine - d)}{\tau_d} + DA\delta(t_{reward}) \tag{16}$$

where we $baseLine$ has its value of -0.04 and the step increase of dopamine concentration DA has it value of 0.45. As shown in figure 18, the eligibility trace records the action the robot had taken and if the robot is unable to receive rewards in a long time, the negative baseline will force the synaptic weights to change in opposite way to the actions in eligibility trace.

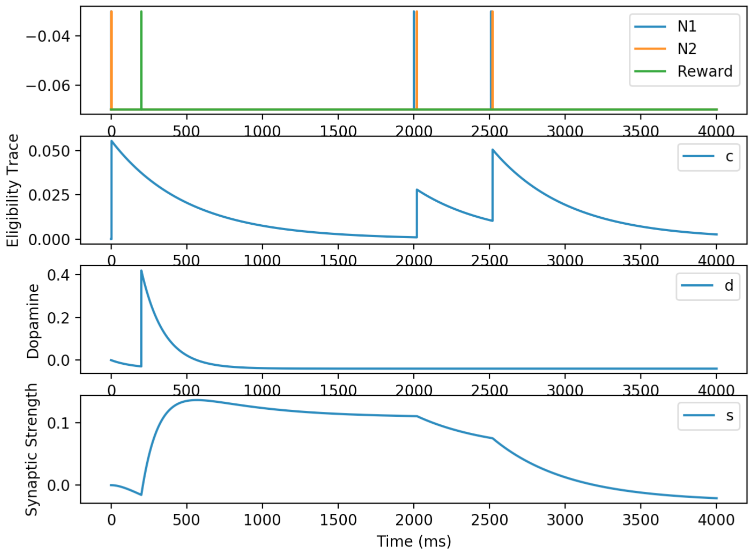

Figure 18: Simulation of dopamine modulated STDP between two neurones with negative dopamine baseline.

3.3.6 Applying the Dopamine Modulated STDP to the Robot

An experiment was conducted to investigate how the weight values are changed in the network after the robot received its reward. A noise was injected into the motor neurone and this noise will force the robot to turn right. A reward is placed on the path the robot will take, so the robot was explicitly programmed to collect its reward.

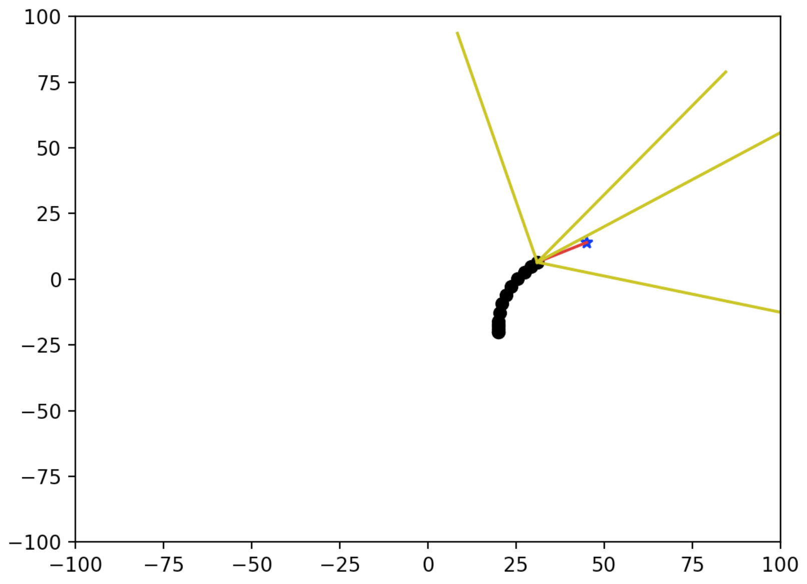

Figure 19: The robot moving path for the simulation in figure 20.

The figure 19 shows how the robot is explicitly programmed to approach the reward and this experience can let the robot to gain the awareness of when the right sensor detects the rewards, turning right is very likely to allow itself to reach the reward. From this example, it is not hard to say that the sensor gain and motor gain are interacted with each other, they have to be in a reasonable ratio to let the robot learn successfully. In this project, the sensor gain was set to be 60 and motor gain is set to be 5.

As shown in figure 20, at 90ms, the robot received its reward and the learning not only increased the required weight (w10) but also increased other unrelated weight (w1).

The weight W1 connected neurone N0 to neurone N1 which means if the left forward motor neurone (N0) fires it will cause the right forward motor neurone fire

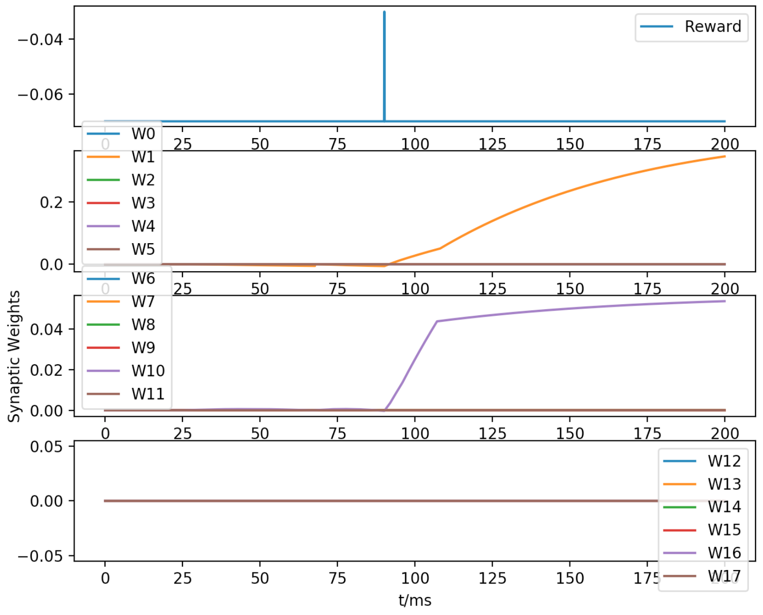

Figure 20: A noise was injected to the motor neurone that make the robot turning right to obtain the reward, the dynamics of weight values during this 200ms simulation is plotted.

(N1). This is not the behaviour we expected, due to every time the robot is starting to turn right, then the left motor neurone will then be triggered. The W10 connected n3 to n0, this implies when the right sensor (N3) detect signals it will cause the left motor to move forward, then the robot will turn right to approach the reward. This is the behaviour we are expected to have through this experiment.

This experiment shows how the undesired behaviour is learned when the correct behaviour is performed. It is important to note that, the negative dopamine baseline is used to remove this kind of undesired behaviour and it will be discussed in next chapter.

3.4 Neuronal Motor Noise and Exploration

In order to allow the robot to explore the environment properly, both forward motor neurones were injected a Poisson noise of 10 Hz [6]. Figure 21 shows the exploration path taken by the robot for 3000ms which is long enough to show the exploration behaviours of the robot. The robot will explore the environment forever if there is no any light source in the environment.

There is a contradict requirement of the noise setting and learning performance. At the beginning of the simulation, the robot will collect some rewards through random motions and the dopamine modulated STDP can allow the network to learn the good behaviours which represented the association between sensors and the actions of motors. The reward based learning is dependent on the random motions, but when the robot has properly learned, the noise will disturb the correct behaviours. For example, the robot has already known to turn right when the signals are detected by its right sensor, in fact, the robot turn left sometimes at this condition due to the noise triggered the right motor. This lead to the problem that the robot is often unable to reach the reward even the correct behaviours are learned.

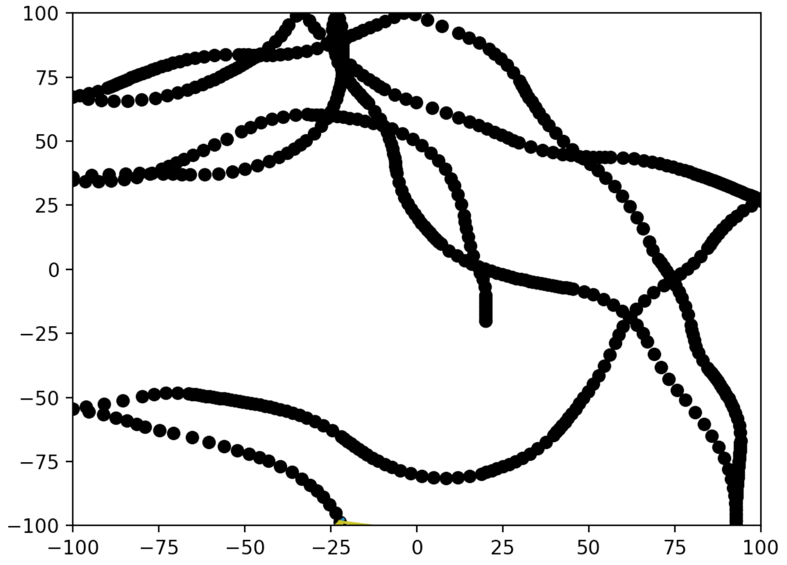

Figure 21: The exploration path taken by the robot for 3000ms

To deal with this issue, the noise frequency has to be carefully chosen. A relatively greater noise frequency can allow the robot to learn faster as it can perform more efficient exploration behaviours, but it will disturb the correct collecting behaviours significantly. A relatively smaller noise will not disturb the correct collecting behaviours significantly, but the robot will take a long time to learn. Through several testing experiments, a good motor noise frequency is found to be 20 times smaller than the maximum sensor input frequency. When the maximum sensor input frequency is 200 Hz, the motor neuronal noise is selected as 10 Hz.

4 Results and Discussion

Regarding simulation results, firstly, several control experiments will be conducted to compare the rewards collecting performance among situations of completely random motions, STDP + ADS, and dopamine modulated STDP + ADS.

In the second experiment, we set light sources as rewards and punishment alternatively in different time regions, to check whether the dopamine modulated STDP can allow the robot to learn dual behaviours: rewards-attracting and punishments-avoiding.

4.1 Performance Comparison Experiment Between Different Configurations

The control experiment is conducted to measure how much the dopamine modulated STDP can improve the rewards collection efficiency compared to when only random motions are applied or STDP and ADS are applied.

4.1.1 Experiment Set Up

The robot motor neurones (N0 and N1) were injected a Poisson Noise of 10 Hz. In this case, the robot can only move randomly and no learning occurs at all.

For the second set of simulations, only STDP and ADS were applied to the network. Because the main learning principle is STDP, so the equation (8) was used, this enables the directional damping to the STDP. Another thing is the learning rate of the STDP model are determined by A_+ and A_-, so the A_+ was set to be 1 and $A_- = 1.5A_+$. Similarly, 10 Hz Poisson neuronal noise was injected to both forward motor neurones.

The dopamine modulated STDP was applied by combing differential equations (9), (10) and (16). As we mentioned in above sections, DA, τ_c and τ_d are all determine the learning rate of the model. So the value of these parameters was carefully chosen, where DA is 0.45, τ_c is 300 ms and τ_d is 100 ms. Because the DA we used for this simulation is relatively big, thus A_+ was set to a smaller value of 0.025 compared to the simulation with only applying STDP and ADS, and similarly $A_- = 1.5A_+$.

The value for baseLine was set to be -0.04 and 10 Hz Poisson neuronal noise was injected to both forward motor neurones.

4.1.2 The Comparison of Rewards Collection Performance

Firstly, the total rewards received from each of the simulations are analysed and the statistical results from each type of the learning principle are as shown in table 1 and figure 22.

Table 1: The statistical results of total rewards collected by using different learning principles among 15 trials.

Rewards Count	Random Movement	STDP+ADS	DA-STDP+ADS
Mean	31.6	36.2	54.27
Median	31	36	54
Standard Deviation	4	14	12

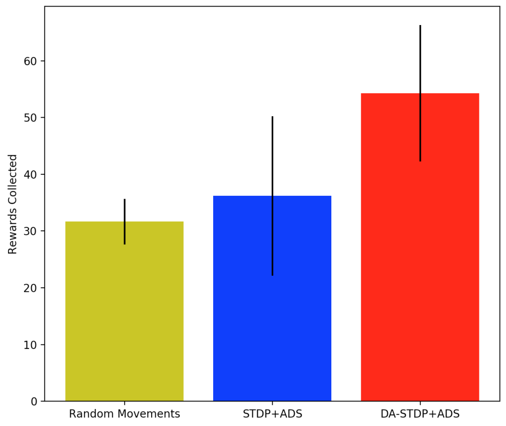

Figure 22: The average total rewards received from 15 simulations of random motions, STDP+ADS and DA-Modulated STDP + ADS, more complex learning method contributes to greater total amount of rewards collected.

As the result shown in table 1 and figure 22, the model using the more complex learning principle can receive more rewards over 10s. It also shows that the standard deviation of the model using random movement has the lowest value, that it receives a similar amount of rewards from every simulation. However, the model using

STDP+ADS and Dopamine Modulated STDP have greater and similar standard deviation which means the performance of these learning principles are not very stable. The model using STDP+ADS have slightly higher mean and median value of rewards collected than the model just using random movement and its standard deviation is high as 14, thus it is not very certain to say that the model using STDP+ADS can have better rewards collection efficiency through its learning than the model only using random movement.

The model using the Dopamine Modulated STDP has almost 40% higher mean value of rewards collected, but its performance is not very stable, as the standard deviation is high as 12. Thus, it is not guaranteed to learn and achieve the good rewards collecting efficiency every trial, but it can have a significantly higher collection efficiency in most of the time through its learning, as its mean and median rewards collected are about 40% higher than the model without learning.

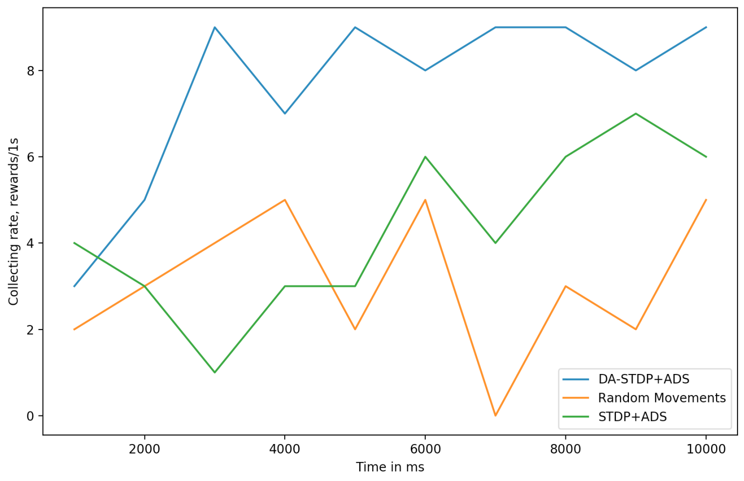

Figure 23: The curves of collecting rate plotted over time of three different methods applied to the agent, the curve of DA-STDP+ADS shows the idea curve in which the collection rate first jump to a high value then maintain this rate in the rest of the time.

However, the table 1 and figure 22 can only show the total rewards received and the stability of collecting rewards of the model from using three different learning principles, but it cannot show how the learning principle improves the rewards collection efficiency over time. As shown in figure 23, the rewards collection rate over time from the three different learning performance applied are presented. It is very obvious that the model using dopamine modulated STDP improves its collecting rate very quickly and it reaches about 8 rewards per second at 3000ms. The collecting rate then fluctuates slightly around 8 rewards per seconds and its performance then become stable after 3000 ms.

The model applying STDP+ADS and only random movements have a very unstable performance curve and both have significant lower collection rate than the model using dopamine modulated STDP. It is also obvious that the collection rate of STDP+ADS has a higher value than random movements after about 5000 ms, but it is fluctuating a lot, thus it is uncertain to say that the model using STDP+ADS can improve its collection efficiency.

As we expected, the STDP alone cannot perform good learning to the distal reward problem, but the dopamine modulated STDP can allow the robot to learn effective behaviours for rewards collection and its performance is relatively more stable as observed from the performance curve in figure 23. However, the model using dopamine modulated STDP cannot have a similar amount of collected rewards every time, that means even the dopamine modulated STDP is able to allow the robot to learn and achieved better collection performance, but the model itself is not very sophisticated, as its learning performance is not stable enough.

4.1.3 The Behaviours Learned by the Dopamine Modulated STDP

The idea behaviour of the robot is when the right sensor detect the rewards it will trigger the left motor to allow itself turning right to approach the rewards. If the left sensor detect the rewards it also should be able to trigger the right motor. Therefore, the weight values of between right sensor neurone and left motor neurone and between left sensor and right motor should be close to its maximum value.

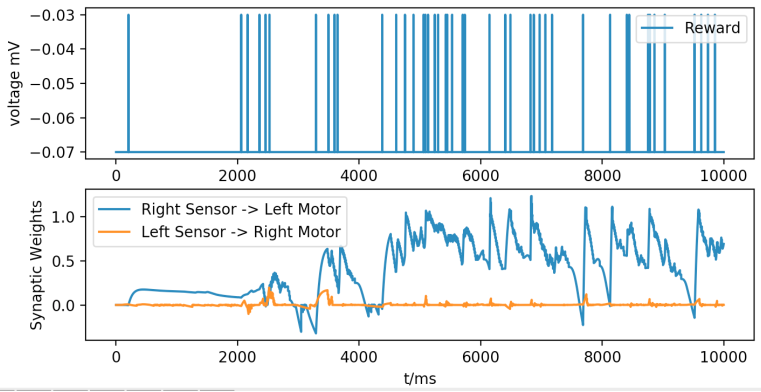

Figure 24: Case A, a simulation result of desired weights and the rewards collected. The graph on the top shows the rewards collected at every time step, this is recorded by a neurone and rewards are presented in the form of spikes, the second graph shows the changing of the desired weights, only the behaviour of "Right Sensor → Left Motor" is learned

As shown in figure 24 and 25, in every simulation the robot can only learn either "Right Sensor to Left Motor" or "Left Sensor to Right Motor" behaviour, but it cannot learn both. Which behaviour the robot will learn is depending on how the first reward is collected. If the first reward is obtained by turning right then the "Right Sensor to Left Motor" behaviour will be learned and be further enhanced in future learning, but the second behaviour ("Left Sensor to Right Motor") will not be learned.

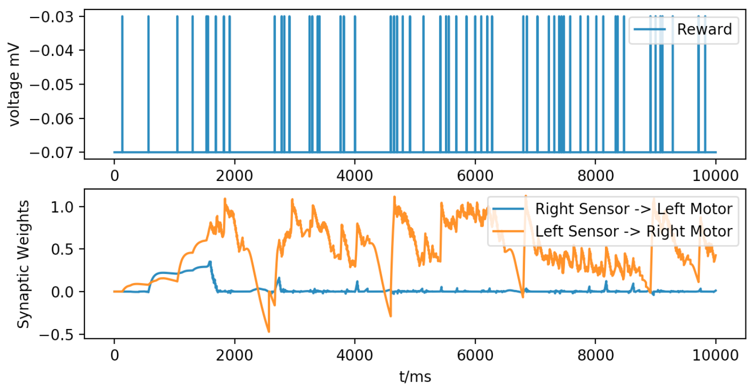

Figure 25: Case B, a simulation result of desired weights and the rewards collected. The graph on the top shows the rewards collected at every time step, this is recorded by a neurone and rewards are presented in the form of spikes, the second graph shows the changing of the desired weights, only the behaviour of ”Left Sensor → Right Motor” is learned

4.1.4 Learned Orbiting Behaviour

As we discussed in the above section, the results are not as what we expected, as every time only one idea behaviour can be learned. If we only consider the robot is approaching a signal reward, the weights for "Left Sensor → Right Motor" and "Right Sensor → Left Motor" will be the desired behaviour, as it will allow the robot to turn and move towards the reward. The weights for "Left Sensor → Left Motor" and "Right Sensor → Right Motor" will force the robot to move away from the reward and they are undesired. In figure 26, these four behaviours are plotted and it clearly shows that only the desired weight for right sensor has leaned (Right Sensor → Left Motor) and the left sensor has leaned the undesired weights (Right Sensor → Right Motor).

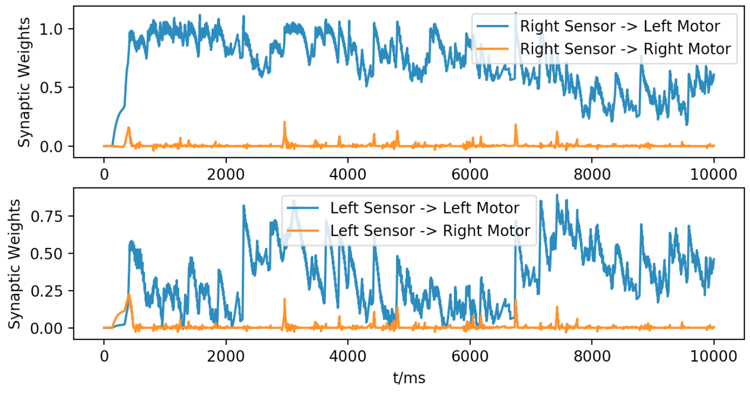

Figure 26: The simulation shows the robot learned some interesting behaviours that allow robot have a circular motion around the environment. This allow the robot be able to collect rewards constantly as the rewards are arranged in a rectangular shape, the weight values that cause the circular motion of the robot are plotted.

This looks not very idea, but when the robot moving path is visualised, we can see this learned behaviour is very suitable for this environment. As shown in figure 27, the path taken by the robot is plotted, due to the weights are shown in figure 26. As we can see, the robot is moving in a circular path, and the rewards are arranged in a rectangular shape, thus, the robot with this orbiting behaviour can collect rewards constantly.

The path shows that the robot will be able to gain rewards through this orbiting behaviour. The simulation also exhibited that if the circular path shape has a radius, start location or angular velocity which is not very efficient to gain rewards, the

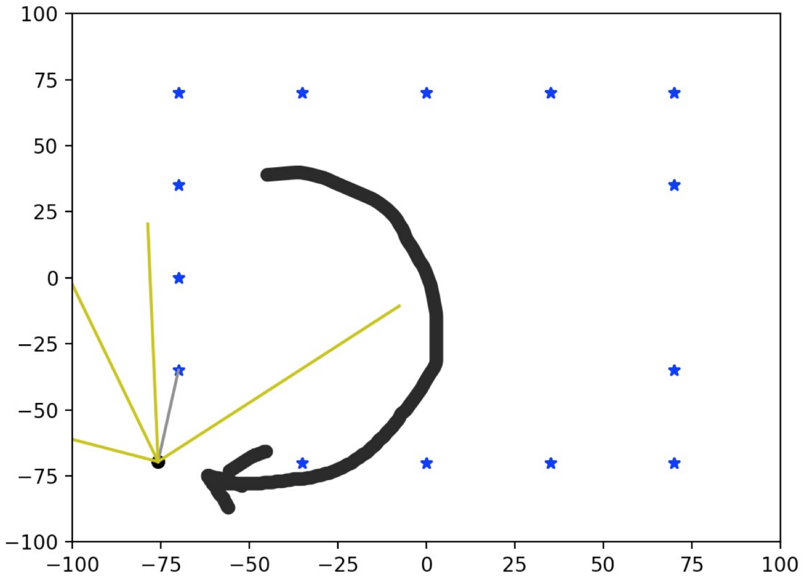

Figure 27: The visualised circular motion of robot with the weight values in figure 26, the robot moving in circular path and it is efficient to collect rewards for this environment.

weights in the neural network will dynamically change the three parameters to edit the circular path, which could result in more efficiency behaviour. For example, if the robot is doing orbiting behaviour in the middle of the environment and the path circuit have a small radius. After a period of time, if the robot cannot receive any rewards, weights in the neural network will be changed and the robot will move to another location and perform a different circular motion to order to gain more rewards.

4.1.5 Rectify the Moving Path Automatically When Collection Performance is Poor

As we mentioned in the above section, if the robot is unable to obtain reward constantly through its current circular motion, the robot will then rectify its moving path. It is possible that the undesired behaviour is accidentally learned and it should be removed. As depicted in figure 28, the changing of 18 weight values over time are presented with the rewards collection rate. At 3000ms, the collection rate starts dropping down, the network quickly realised that the current weight values are not good at rewards collecting, then all the learned weights value start dropping down

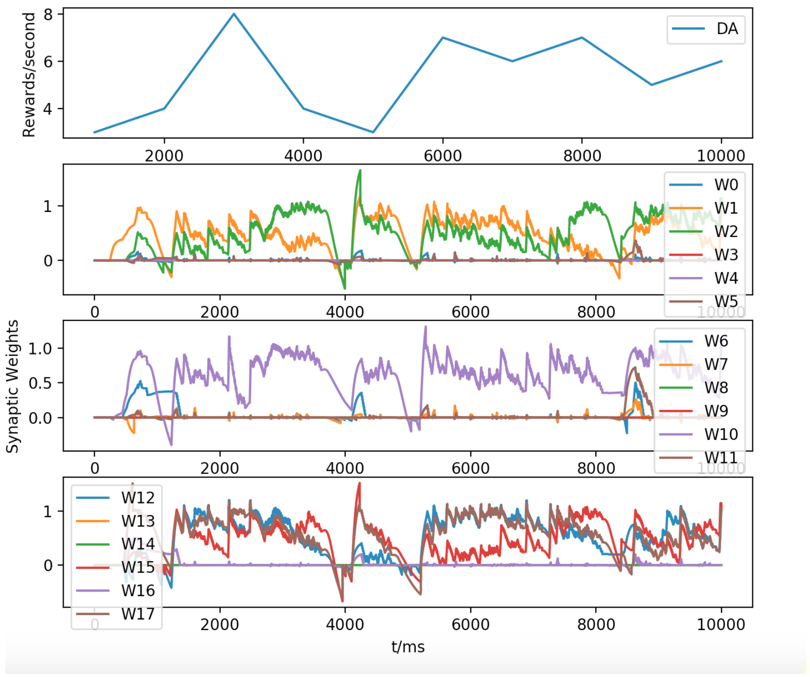

Figure 28: At around 5000ms, the collection rate dropped down significantly, the network then recognised this low performance of collecting rewards, the relearn capability of weight values when the current behaviour is not capable of collecting rewards efficiently is plotted.

rapidly due to its negative dopamine concentration baseline. At about 5000ms, the robot restarts its learning then the new weight values contributed to a high and stable rewards-collecting rate in the rest of the simulation.

There is a possibility that the learned weight values are not good enough for collecting rewards, so when that happens, an opposite behaviour is expected by applying the negative baseline. However, the results shows the negative baseline is only initialised all the weights value when the robot is not able to collect rewards in a relatively long time, this is not exactly what we expected but it can rectify the robot moving path if the behaviours learned currently are not efficient at receiving rewards.

In addition, it is also obvious that the weight values are fluctuating a lot and it is not very stable that is because the robot sensors are not very accurate. The sensors can only tell the rewards is on the right hand side or left hand side, thus even if the correct behaviour is learned the robot still very likely to miss the rewards, as it only has a rough idea of where to turn. When the reward is missed, the weight represent the learned behaviour will drop down, due to the behaviour is consider to be not efficient to collect rewards. It will increase back only if the robot collected rewards through this behaviour again. This contribute to the weights fluctuating and it affect the performance of learning.

4.1.6 Network Firing Rate and Robot Speed

There is no doubt that higher moving speed of the robot will allow the robot more likely to gain rewards. Figure 29 has shown us a clear picture of the network firing frequency overtime. As we can see, the firing frequency quickly rise to about 16k Hz and it fluctuating around 10k Hz. This shows the increasing and decreasing of the robot moving speed, as the robot motor is triggered by the neuronal firing frequency. The robot is always moving much faster than its initial speed, this can allow the robot to collect rewards much more efficient.

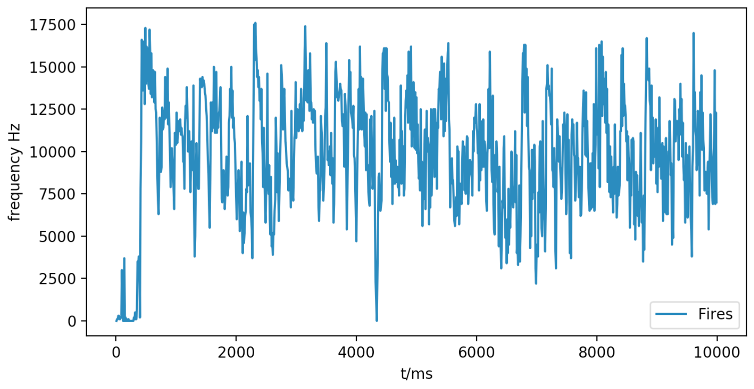

Figure 29: The overall firing rate of the network over time, which determines the robot moving speed

4.2 Punishments and Rewards

The dopamine modulated STDP can have the capability to perform both rewards collecting and punishments avoiding behaviours. This can be done by changing the sign of DA ($DA* = -1$). To prove this, we simulated another 10 seconds simulation. In the first 5 seconds, the targets in the environment were all set to be rewards ($DA = +0.45$), and the later 5 seconds the targets in the simulation were all set to be punishments($DA = -0.45$).

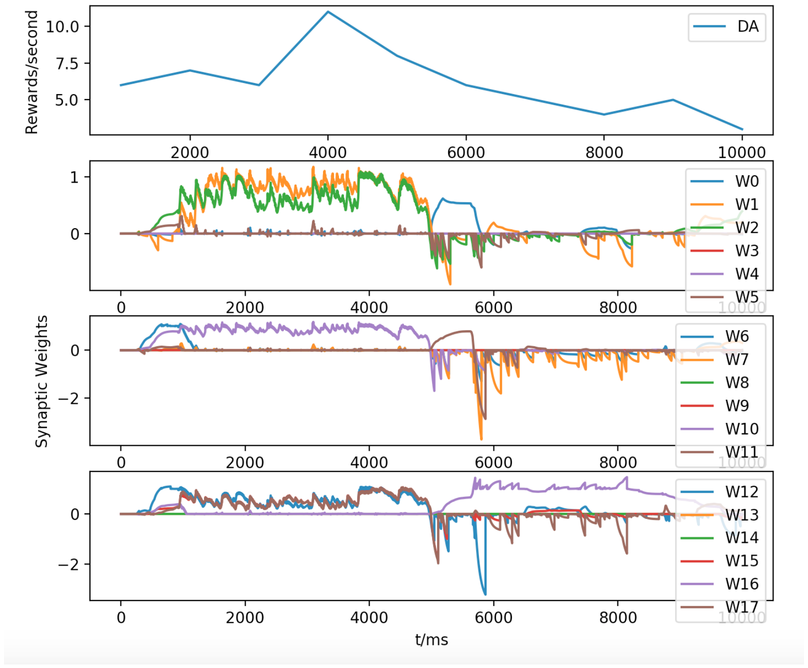

Figure 30: The plotted weights values and rewards collecting rate of the robot, rewards was change to be punishments at 5000 ms. After the targets are set to be punishments from rewards, the collection rate dropped down dramatically and all the weights also dropped down. W16 increased after the change, due to W16 is representing an avoiding behaviour.

The synaptic weights and the rewards collection rate are plotted in figure 30, as we can see at 5000ms the targets become punishments from rewards, all the learned synaptic weight values drop down very quickly, and few weights increased up then quickly dropped down. The interesting thing is the weight 16 increases to 1 and the weight 16 (W16) is the synaptic connect Right Sensor → Right Motor, which is classified as avoiding behaviour and the combining effect of removing learned behaviours and emerged new behaviours contribute to a significant drop down of rewards collecting rate. This proves that not only the approaching behaviours are removed after the rewards were set to be punishment, but also some avoiding behaviours emerged.

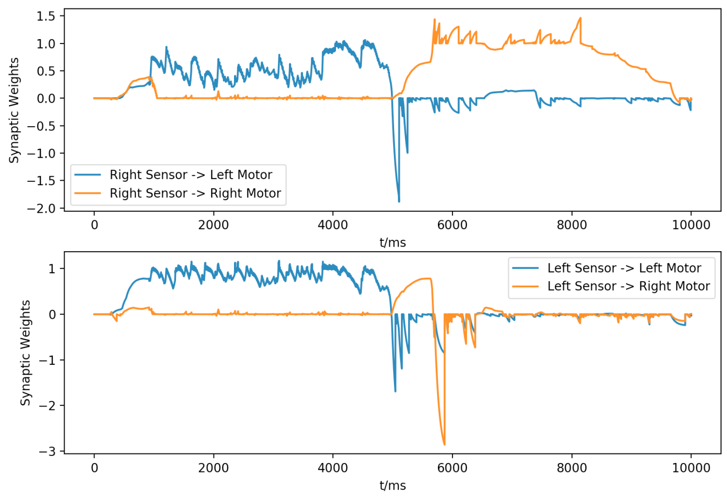

Figure 31: Rewards were change to be punishments at 5000 ms, the weights plot in blue trace is representing a circular motion and the circular motion immediately disappeared after the change. One avoiding behaviour is emerged after the change, which is Right Sensor → Right Motor.

Figure 31 shows a clear view about how these behaviours were changed after the rewards were set to be punishments, in which when the rewards were set to be punishments, behaviour Right Sensor → Left Motor and Left Sensor → Right Motor are quickly unlearned. This implies the when the robot collected some punishments, it immediately stopped its circular motion. Then the behaviour Right Sensor → Right Motor emerges. That means the robot leaned one avoiding behaviour. Figure 32 also shows that the firing rate of the neural network drops down dramatically when the rewards were set to be punishments immediately, which results in a significantly lower speed of the robot and the robot moving slower cause a much lower efficiency of collecting targets. This further demonstrates that the dual behaviour can be learned if the nature of target is immediately changed.

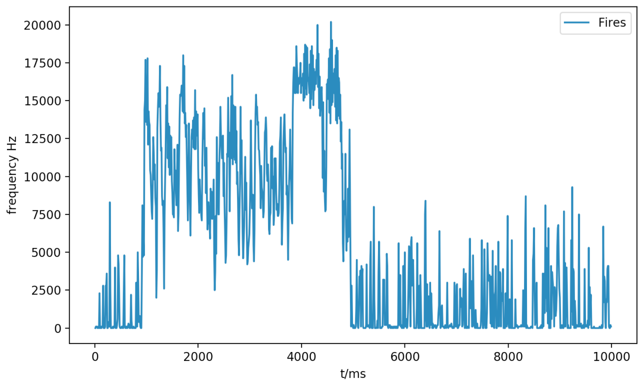

Figure 32: The plotted firing rate of the neural network used in the robot, rewards were change to be punishments at 5000 ms. The firing rate increased to a high level after 1000ms, after the change the firing rate drops down immediately. This implies that the robot starts to avoid targets by lower down its moving speed.

5 Conclusion

This project aims to investigate spiking neural networks and implement an SNN based motion controller for a light sensitive robot. The literature was viewed, the core concepts and theory summarised, a design for the SNN controller was made and finally several experiments were conducted on the performance of this design. From the experiments observations and results, the following conclusions about the SNN controller can be made.

5.1 The Nature of the SNN

The SNN is a fully connected network based on local synaptic weight changing mechanism, where the synaptic weight is determined only by the firing rates of its two connected neurones. So it is difficult to learn behaviours of global meaning, such as more efficient reward collecting. To this, a global mechanism should be added to the network, a dopamine-like reinforcement learning is used to modulate the STDP and achieve better performance due to its function of behaviour filtering according to rewards collected.

5.2 Noise Injected to the SNN

A noise process is added the neurones to make the robot move randomly. This is necessary for an autonomous robot of learning ability, otherwise, the robot will stay in motion when it does not face any light sources and will never have the opportunity to learn. But the magnitude of noise must be carefully selected because it will cause the robot to move along a path biased from the desired one.

5.3 Stability of the SNN

In our network, the weight values will suddenly saturate at maximum value, this causes a stability problem where the weights will be stuck at a fixed value and no further learning is possible. This problem is solved through adding another biological principle: activity-dependent scaling (ADS). With the aid of ADS, the changing of weights in the system will always be possible and some competitive behaviours can be fostered.

5.4 Reinforcement Learning

The dopamine modulated STDP shares several similarities with temporal difference reinforcement learning. The eligibility trace used in this project is correlated to the temporal difference reinforcement learning rule, the neurone firing orders are stored as an exponentially decaying function and can be used to filter out the undesired behaviour by changing synaptic weights when the delayed reward is received. The eligibility trace plays a role to speed up the learning process.

The major difference between temporal difference reinforcement learning and the model implemented using dopamine modulated STDP lies in the method of updating the weights: temporal difference (TD) learning updates the values of weights based on the cost function which is based on the difference between the expected reward and received reward. Dopamine modulated STDP updates the values of weights based on the received reward only. Dopamine modulated STDP always encounters the problem of over learning when too many rewards have been collected. TD learning can effectively avoid this problem since the expected rewards can be adjusted according to the sum of the rewards collected. This is the one of the main issues which affect the learning performance of the model.

5.5 Limitations

Some additional problems are also need to be addressed. The robot model and network structure proposed in this project are not sophisticated enough to perform a stable and high performance behaviours at reward collecting. The robot sensors were simulated in a naive way, as it can only sense the target using left or right sensors. The angle resolution is too low, thus it cannot define how much the robot should turn according to the sensor readings. The major limitation of this project is that the network is too simple and it cannot develop more sophisticated behaviours.

6 Future Work

In this implementation, several limitations were found which affected the learning performance of the agent.

- Naive Sensors cannot detect targets in a precise way.
- Six neurones network structure is too simple to self organise into several subgroups with different functionality.
- Dopamine modulated STDP always encounters the problem of over learning when too many rewards have been collected.

Further improvements can be conducted around these three limitations.

6.1 More Sensors

Multiple sensors need to be implemented rather than just two, that can pass the information of what angle the robot needs to turn rather than just a rough idea of left or right. With this more accurate robot model, the agent will have a greater potential to perform rewards collecting or punishment-avoiding task.

6.2 More Sophisticated Network Structure

The most important thing of improving the agent learning performance is to apply a more complex neural network which has the capability to allow a network to self organise into several subgroups with different functionality.

One of the solutions is to apply the network architecture used in model implemented by Chorley & Seth [21]. Firstly, the neural network has a scale of 1000 neurones, this can allow a group of neurones to self-organise into several subgroups with different functionality, then multiple behaviours can be learned and remembered. For the second issue, the dopamine value used in this project is a fixed value, which causes the overfitting problem when the learned behaviour is already good. Thus, a method to calculate cost function in between expected reward in TD reinforcement learning and perceived rewards is necessary. As depicted in figure 33, the inhibitory neurones are interacting with the stimulus neurones until the system is stable and it can perform short-term memory for cost function in TD reinforcement learning. Further study regarding this theory is needed [21].

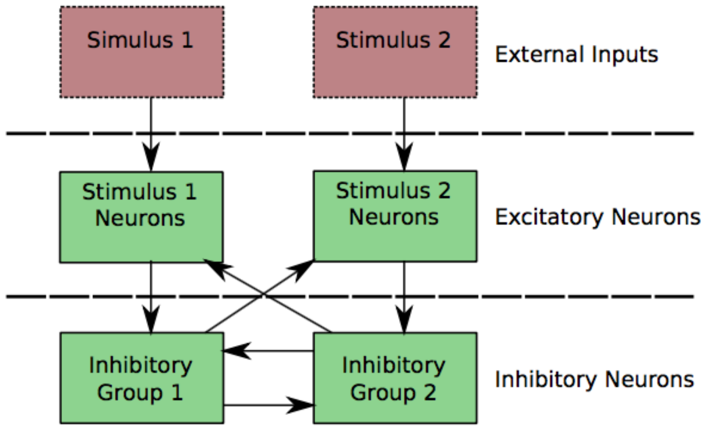

Figure 33: The neural network structure of a winner-takes-all mechanism. Inhibitory neurones are needed for removing the behaviour leaned which is not contribute to optimising agent performance [39].

6.3 Hardware Implementation

Furthermore, neurones in spiking neural network are communicating with each other using time-dependent spike trains, this is similar to some digital systems. Implementation of SNN in field programmable gate array FPGA has been frequently studied, an implementation of this neural control on hardware like FPGA which could possible to provide contributions to autonomous system control industry.

References

[1] S. P. a. S. R. S. Singh, ”Reinforcement learning with replacing eligibility traces. Machine Learning,” vol. 22, pp. 123-158, 1996

[2] A. Reis’s, ”Reinforcement Learning: Eligibility Traces and TD(lambda),” 2 November 2017. [Online]. Available: https://amreis.github.io/ml/reinf-learn/2017/11/02/reinforcement-learning-eligibility-traces.html. [Accessed 3 May 2018].

[3] S. Song, ”Competitive Hebbian learning through spike-timing-dependent synap-ticplasticity,” Nature Neuroscience, vol. 3, pp. 919-926, 2000.

[4] H.Seung, ”Leaning in Spiking Neural Networks by Reinforcement of Stochastic Synaptic Transmission,” Cell Press, vol. 40, no. 6, pp. 1063-1073, 18 December 2003.

[5] X. Wang, ”Mobile robots’ modular navigation controller using spiking neural networks,” Neurocomputing, vol. 134, pp. 230-238, 2014.

[6] E.D.Paolo, ”Spike-Timing Dependent Plasticity for Evolved Robots,” Adaptive Behaviour, vol. 10, no. 3-4, pp. 243-263, 2002.

[7] D. F. D. Shulz, ”Chapter 9- Spike Timing-Dependant Plasticity,” in Neural Circuit Development and Function in the Brain, Berkeley, Academic Press, 2013, pp. 155-181.

[8] W. S. M. &. W. Pitts, ”A logical calculus of the ideas immanent in nervous activity,” The Bulletin of Mathematical Biophysics, vol. 5, no. 4, pp. 115-133, 1943.

[9] T. P. Trappenberg, Fundamentals of Computational Neuroscience, New York: Oxford University Press, 2002.

[10] K. C. Berridge, ”Pleasure Systems in the Brain,” Neuron, vol. 86, no. 3, pp. 646-664, 2015.

[11] R. A. &. A. A. A.busnaina, ”Spiking Neuron Models: A Review,” International Journal of Digital Content Technology and its Applications, vol. 8, no. 3, pp. 14-21, 2014.

[12] E. M. Izhikevich, "Simple Model of Spiking Neurons," Transactions on Neural Networks, vol. 14, no. 6, pp. 1569-2572, November 2003.

[13] E. Orhan, "The Leaky Integrate-and-Fire Neuron Model," 20 November 2012. [Online]. Available: http://www.cns.nyu.edu/ eorhan/notes/lif-neuron.pdf. [Accessed 2018 Janunary 21].

[14] W. Gerstner, "Neuronal Dynamics online book," Cambridge University Press, 8 September 2014. [Online]. Available: http://neuronaldynamics.epfl.ch/online/index.html. [Accessed 2018 Janunary 21].

[15] D. J. Sjostrom, "Scholarpedia," 10 February 2010. [Online]. Available: $http://www.scholarpedia.org/article/Spike-timing_dependent_plasticity$. [Accessed 21 Janunary 2018].

[16] E. A. D. Paolo, "Evolving spike-timing-dependent plasticity for single-trial learning in robots," The Royal Society, vol. 361, no. 1811, pp. 2299-2319, 18 August 2003.

[17] D. O. HEBB, The Organization of Behavior, New York: Wiley & Sons, 1949.

[18] E. M. Izhikevich, "Solving the distal reward problem through linkage of STDP and dopamine signaling," BMC Neuroscience, vol. 8, no. 2, p. 15, 2007.

[19] R. Legenstein, "A Learning Theory for Reward-Modulated Spike-Timing- Dependent Plasticity with Application to Biofeedback," PLoS Computational Biology, vol. 4, no. 10, pp. 100-180, 2008.

[20] N. Fremaux, "Functional Requirements for Reward-Modulated Spike-Timing-Dependent Plasticity," Journal of Neuroscience, vol. 30, no. no, pp. 13326-13337, 2010.

[21] P. a. S. Chorley, "Closing the sensory-motor loop on dopamine signalled reinforcement learning," Proceedings of the 10th international conference on Simulation of Adaptive Behavior: From Animals to Animats, pp. 280-290, 2008.

[22] W. S. R. W. J. a. H. B. Pan, "Dopamine cells respond to predicted events during classical conditioning: Evidence for eligibility traces in the reward-learning network," The Journal of Neuroscience, vol. 25, pp. 6235-6242, 2005.

[23] T. A. P. a. S. W. Ljungberg, "Responses of monkey midbrain dopamine neurons during delayed alternation performance," Brain Research, vol. 567, pp. 337-341, 1991.

[24] P. a. S. A. Chorley, "Dopamine-signalled reward predictions generated by competitive ex- citation and inhibition in a spiking neural network model," Frontiers in Computational Neuroscience, vol. 5, 2011.

[25] A. a. L. L. Gupta, "Hebbian learning with winner take all for spiking neural networks," Neural Networks, p. 1054 ?1060, 2009.

[26] R. a. B. A. Sutton, "Reinforcement Learning: An Introduction," The MIT Press, 1998.

[27] W. Schultz, "Predictive reward signal of dopamine neurons," Journal of Neurophysiology, vol. 80, pp. 1-27, 1990.

[28] J. a. V. E. D. Maunsell, "Functional properties of neurons in middle temporal visual area of the macaque monkey. i. selectivity for stimulus direction, speed, and orientation," Journal of Neurophysiology, vol. 49, pp. 1127-1147, 1983.

[29] A. a. H. Hodgkin, "A quantitative description of membrane current and its application to conduction and excitation in nerve," The Journal of Physiology, vol. 117, p. 500?544, 1952.

[30] F. Kunz, "An Introduction to Temporal Difference Learning," Department of Computer Science TU Darmstadt, [Online]. Available: http://www.ias.informatik.tu-darmstadt.de/uploads/Teaching/AutonomousLearningSystems/Kunz_ALS_2013.pdf. [Accessed 2 May 2018].

[31] R. R. K. a. A. N. Burkitt, "Delay Selection by Spike-Timing-Dependent Plasticity in Recurrent Networks of Spiking Neurons Receiving Oscillatory Inputs," PLoS Computational Biology, vol. 9, no. 2, p. e1002897, 2013.

[32] T. Iakymchuk, "Simplified spiking neural network architecture and STDP learning algorithm applied to image classification," EURASIP Journal on Image and Video Processing, 2015.

[33] S. Raschka, "What is the Role of the Activation Function in a Neural Network?", Michigan State University, [Online] https://www.kdnuggets.com/2016/08/role-activationfunctionneuralnetwork.html [Accessed 4 May 2018].

[34] KHANACADEMY, Overview of neuron structure and function, [Online] https://www.khanacademy.org/science/biology/humanbiology/neuron-nervoussystem/a/overviewofneuronstructureandfunction. [Accessed 2 May 2018].

[35] Turrigiano, G. Activity-dependent scaling of quantal amplitude in neocortical neurons, nature, vol. 391, page 892, 1998.

[36] Matsumoto, M. & Hikosaka, Two types of dopamine neuron distinctly convey positive and negative motivational signals, Nature, vol. 459, page 837-841.

[37] Kevin Frans, Model-Free Prediction and Control, [Online] http://kvfrans.com/modelfreepredictionandcontrol/ [Accessed 4 May 2018].

[38] Lewis, M, Etienne Cummings, R., "Cohen, and Hartmann, M. Toward biomorphic control using custom avlsi cpg chips", Robotics and Automation, vol. 1, page. 491-500, 2000.

[39] R Evans,"Reinforcement Learning in a Neurally Controlled Robot Using Dopamine Modulated STDP", Imperial College London, Final Project for MSc Degree in Advanced Computing of Imperial College London, 2012.