PALADYN Journal of Behavioral Robotics

VERSITA

Research Article · DOI: 10.2478/s13230-012-0003-6 · JBR · 2(3) · 2011 · 126-133

Modular non-computational-connectionist-hybrid neural network approach to robotic systems*

C.D. Bamford, R.J. Mitchell†

Cybernetics, School of Systems Engineering, University of Reading, Reading, UK

Received 2011/11/11

Accepted 2012/01/26

Abstract

Spiking neural networks are usually limited in their applications due to their complex mathematical models and the lack of intuitive learning algorithms. In this paper, a simpler, novel neural network derived from a leaky integrate and fire neuron model, the 'cavalcade' neuron, is presented. A simulation for the neural network has been developed and two basic learning algorithms implemented within the environment. These algorithms successfully learn some basic temporal and instantaneous problems. Inspiration for neural network structures from these experiments are then taken and applied to process sensor information so as to successfully control a mobile robot.

Keywords

neural networks · robotics · spiking neurons · hybrid systems

1. Introduction

The control of robotic systems has been a large area of research for at least twenty years. Whether the robot system is a bipedal walker, a wheeled car-like robot or simply a robot manipulator, each specific system has its own array of mathematical complexities, nonlinearities and system disturbances which need to be taken into account in order to create a viable outcome.

There are many different approaches to reducing these inherent unknowns that have been explored. For example, computational techniques have been adopted in order to successfully adapt simple PD controllers to unknown errors [1]. However when using techniques like this, an exact or highly accurate initial mathematical model of the system is required, then a control method (in this case, PD), must be tailored to the particular error, then the errors are fed back into the control scheme and the parameters adapted to cancel out the various errors.

These parameterisation techniques can be very successful on specific problems, however if the system dynamics change significantly, such as a new degree of freedom being added, the design process has to be re-visited and can potentially make the original control scheme invalid.

Neural network based control schemes inherently attempt to absorb the issues of unknown disturbances by using learning algorithms. However, these methods sometimes present new issues. It can be difficult to train a neural network to dynamic environments. "Supervised" learning techniques can be implemented to create a basis of functionality [2] but this generally limits the usefulness of the network to features similar to those in the training. Learning algorithms commonly use methods such as gradient descent which can fall into local-minima and therefore produce sub-optimal or entirely invalid control.

Neural networks can also have a high computational cost, so they are usually implemented alongside computational methods and only focus on specific tasks, for example visual recognition [3–5] or physical limitations [6].

Some methods have been produced to try to limit the computational use by modifying the structure of the neural network to reduce the number of calculations [7, 8]. Once this structure has been produced, it is generally used for a specific task only, and has to be modified to compensate for environmental changes. This issue is known as the no free lunch theorem [9].

In a biological neural network, different structures in the brain activate during different tasks, and those parts of the brain can adapt to take into account disturbances in the task space in real-time. Neural network methods to adapt to dynamic environments have been developed, but they again are designed for specific problems and usually contain computational components. In [10] a 2D topological arrangement of parallely connected neurons is used to detect obstacles in the joint-space of a robot manipulator. This approach can adapt to avoid various objects in the environment, however as the network is topological, the environment in which conditions can be altered is fixed. A similar approach is taken for real-time path planning in [11]. [12] proposes a control approach which stores configurations of network controllers trained for specific environmental disturbances and uses an algorithm to select the most appropriate network as the environment changes. This is a perfect example of a hybrid approach of computational and connectionist systems.

Other on-line learning approaches to dynamic environments, where system parameters are unknown or unmeasured have been proposed with successful results [13, 14]. Again [13] uses a hybrid of computational and neural network techniques as it involves the use of a nonlinear state observer, whereas in [14] the neural network is used in place of the state observer.

Evolutionary algorithms can also be used to define the weights in a neural network in order to produce desirable functionality in robotics [15]. These can be computationally expensive.

This paper intoduces the idea of non-computational-intensive connectionist-hybrid modular neural network system that inherently

*This paper has been partially published in the IEEE 9th International Conference on Cybernetic Intelligent Systems, 2010, Reading, UK

†E-mail: r.j.mitchell@reading.ac.uk

reduces computational and systematic complexity and allows a neural network to fully control every aspect of the control of a mobile robot system, from inputs to outputs, with no global optimisation or feedback in place. Similar to the sytem proposed by Alnajjar and Murase [16], where a simple biologically inspired neural network structure with multiple sensing abilities is employed to control a mobile robot, the autonomy of the robot system proposed in this paper is achieved by the multiple input neural networks competing for control over the robot's motor driving outputs.

Two potential learning algorithms for this type of neural network system are described. These algorithms essentially "grow" the connections between neurons based on supervised learning algorithms in [8]. More is said about these here.

The network was first tested on a simple logic problem: some details of which are presented here. It was subsequently used successfully in an integrated masters project. The overall aim of this project was to develop a rugged robot capable of exploring an unknown environment. A controller was required for the robot, and the neural network described here was deemed suitable. This paper explains how the network could be used so that the robot can explore the environment avoiding obstacles.

This paper is organised as follows. Section 2 describes the novel neural network with its so called 'cavalcade' neurons. Section 3 considers some possible applications of the network in the field of mobile robotics. Section 4 describes the learning algorithms used for the network, including the results when applied to the logic problem. Section 5 describes how the network was used successfully on the actual robot. The paper ends with a conclusion and considerations of how the network could be used in other applications.

2. Cavalcade neural network

Leaky Integrate-and-fire neurons are used in modelling for their relative mathematical simplicity, their biological relevance, and ability to be analysed in terms of spikes and information flow [17, 18].

The cavalcade neural network (CNN) comprises a set of cavalcade neurons, CNs. A CN is a special case of a leaky integrate and fire neuron where the input current is simplified as an impluse. This method of processing input spikes means inputs to the neuron can be modelled as discrete events in time, and this can reduce computational complexity.

2.1. Analysis of CNNs

In [8], the basic mathematical model of CNNs is described. In the event of a pre-synaptic spike arriving at a neuron, the neuron action potential υ is increased by the spike amplitude α the action potential then decays over time depending on a decay constant ε.

$$\upsilon_k = \upsilon_{k-1} \cdot e^{-\frac{(t_k - t_{k-1})}{\varepsilon}} + \alpha_k \tag{1}$$

Equation 1 describes the state of the system at k, where k represents the index of the discrete spike event.

If additional spikes are presented to the neuron, then the current action potential is added to the incoming spike amplitudes. This value is then the new value to be decayed over time. Figure 1 shows the variation of action potential following the addition of repetitive pikes.

In [8] the authors show that, when the CN is sent pulses at a fixed frequency f for an infinite amount of time, the action potential can be described using a geometric progression. In this paper an improved

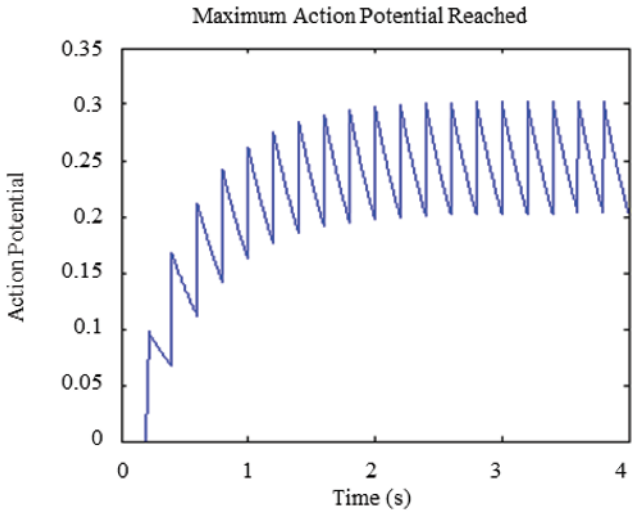

Figure 1. Fixed amplitude and frequency action potential dynamics of a typical cavalcade neuron.

analysis is given, which can cope with spike trains whose amplitude and timings are not fixed.

This analysis builds on [8], which shows that spike trains can be described as dirac-comb functions with varying timings between individual spikes. This allows a concise mathematical description of the cavalcade neuron.

For a spike train of fixed frequency $\frac{1}{t_f}$ and fixed spike amplitude $\propto$, equation (2) can be used to show the action potential of a neuron at any time after the spike train is introduced to the neuron.

$$\upsilon(t) = \propto \sum_{k=-\infty}^{\infty} \left[\int_{-\infty}^{t} \partial(x - k t_f)\, dx \cdot e^{-\frac{(t - k t_f)}{\varepsilon}} \right] \tag{2}$$

However (2) only allows for fixed amplitudes and timing between spikes. In order to modify this equation to allow any spike amplitudes and spike timings, the k terms can be replaced by a vector ϕ containing spike times (3), and a vector A containing correspinding spike amplitudes (4).

$$A = [\propto_0 \quad \ldots \quad \propto_n] \tag{3}$$

$$\phi = [s_0 \quad \ldots \quad s_n] \tag{4}$$

Equation (2) can then be split into two separate parts. Firstly, the calculated dirac comb integral is described by (5).

$$D(t, \phi) = \int_{-\infty}^{t} \partial(x - \phi)\, d\phi \tag{5}$$

Secondly the "leak" function (6), which decays the action potential , is given by:

$$E(t, \phi) = e^{-\frac{(t - \phi)}{\varepsilon}} \tag{6}$$

Taking the Hadamard product of $D(t, \phi)$ and the amplitude matrix A and then multiplying that by the transpose of $E(t, \phi)$ gives an exact

value of the action potential at time t with the given incoming spike train.

$$\upsilon(t, \phi, A) = (A \circ D(t, \phi))\, E(t, \phi)^T \tag{7}$$

Calculating the action potential of a neuron at any time, given that it is subject to spikes with random timing and amplitudes, would be time consuming and non-trivial using equations proposed in [8]. For example, consider the following situation:
A neuron has n incoming synapses with spike trains of unknown amplitude and frequency, the equations (8) and (9) below, given in [8], only account for fixed spike frequency.

$$\upsilon = \frac{\alpha\left(1 - e^{-\frac{nt_f}{\epsilon}}\right)}{\left(1 - e^{-\frac{t_f}{\epsilon}}\right)} \tag{8}$$

Equation (8) shows the calculation for the neuron's action potential after the n^{th} pre-synaptic spike at a fixed time period t_f. This is derived from the fact that at fixed frequency and amplitude, the peaks of action potential of a cavalcade neuron at the time of a pre-synaptic spike is identical to a geometric series.

$$\nu = \frac{\alpha}{1 - e^{-\frac{t_f}{\epsilon}}} \tag{9}$$

Equation (9) is also derived from the geometric progression observable at fixed frequency and amplitude. These two equations, although useful at fixed frequency and amplitude, due to the fact that they are based around geometric progressions, do not work when amplitudes and spike timings are variable.
However the new method given above for calculating action potentials makes analysis of neurons with multiple inputs very simple to implement in mathematics software such as MATLAB. Figure 1 shows action potential $\upsilon(t)$ derived from equation (2) with $\alpha = 0.1$, $\varepsilon = 500$, $t_f = 200$.
Also, due to the discrete spiking and a deterministic decay function, implementation of a real-time system is fairly simple for both iterative and event-driven neural network algorithm approaches.

2.2. Connecting CNs

A Cavalcade neural network, CNN, comprises a network of CNs in any structure, It could include mixtures of layered, fully connected and recursive ganglions. When a CN fires, it causes all axons connected from that neuron to raise a post synaptic spike in the neurons to which they are connected, the amplitude of which depends on the strength of the axon. If the spike causes the action potential in that neuron to exceed the threshold of the destination neuron, it will cause instantaneous firing there, otherwise the firing of the destination neuron will depend on the frequency of the pre-synaptic spikes.
It should be noted that, like 'real' neurons, connections can be either excitatory or inhibitory – depending on whether their α value is positive or negative.
In a CNN, the neurons are able to connect to any other neuron in the network: its structure can be set dynamically. To avoid potential confusion as to the direction signals flow, the 'notation' in Figure 2 is used to show a network. Here CN_1 and CN_2 both send signals to CN_3: the circle on the line from CN_2 shows that it is acting inhibitorally.

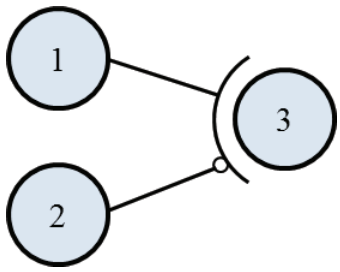

Figure 2. Two CNs connected to a third.

3. Applications

This form of network can be used with a mobile robot in various ways. Described below are ways the network could be used for object detection using ultra-sonic detectors, speed measurement and pulse width modulation.

3.1. Object detection

Due to their effectiveness and simplicity, ultra-sonic sensors are often used to detect objects in the path of any robot. Using some hardware pre-processing, CNs can be configured to fire when the ultrasonic sensors detect an object at a certain distance away.
If ultrasonic pulses are sent out regularly and echoes received shortly after each pulse, the distance from the object that the ultrasonic sensor is detecting will be proportional to the frequency at which pulses are being received. The frequencies at different distances can then be used to configure CNs to fire sequentially as the objects reaches certain distances from the robot. These sequential fires can then be connected to decision making parts of the network to adjust motor speeds, or to change direction.
Using a CNN, this method of object detection requires no onboard calculation of timings between outgoing pulses and incoming received pulses in order to calculate the distance. This is because the relationship between frequency input and saturation level of action potentials in a CN is linear.
A robot may want to map an environment, so it would be useful to be able to detect objects at various distances. In which case, a system could be set up using three CNs which fire at distances 5 m, 2 m and 1 m, respectively. This system could then detect the distance of objects and that would be fed to the computer generating the map. The same configuration could also provide the information to a robot controller whose aim was to explore an environment avoiding obstacles.

3.2. Speed processing

As CNs function by receiving pulses at frequency intervals, or pulses for specific events, calculating the speed of a motor, or any rotating link is a very simple task.
First what is required is a rotary encoder, which generates a series of pulses. These pulses are then sent to a CN, whose action potential saturation υ will be proportional to the speed of the motor itself.
Specifically, using equation (9), ε and α could be calculated so that the saturation potential ν is directly proportional to the angular velocity of the motor and in turn the speed the robot moves. This is achieved by generating a pre-synaptic spike at the frequency produced by the rotary

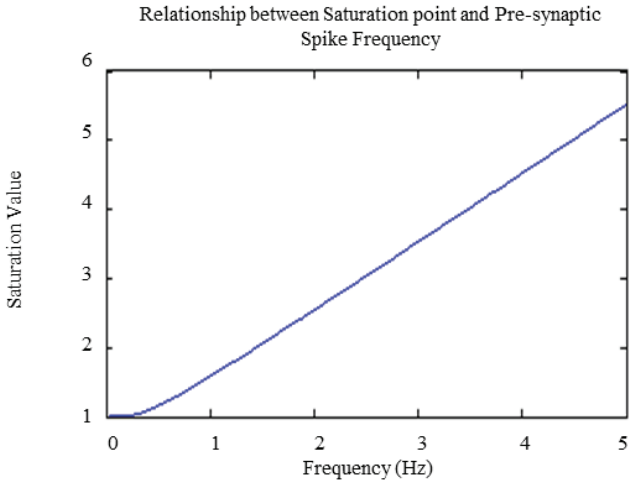

*Figure 3. The linear relationship between input frequency and CN saturation voltage **v**.*

encoder transition through an infared sensor. This linear relationship is shown in Figure 3.

3.3. Pulse Width Modulation

Most electronic systems require some form of post processing from a neural network's output to drive them. This can be the case with a CN. Outputs of the network such as the saturation level of the neuron, or the frequency at which it is firing, can be used as a control signal and then post-processed into an analogue or digital waveform.

For controlling the speed of a motor, a pulse width modulated signal can be used. This can be achieved readily with CNs, with the addition of a basic latch circuit.

Consider two CNs, each with identical characteristics and parameters, each CN is pulsed at a frequency that will cause them to reach their firing threshold at the 30th input pulse and then they will fire themselves. These two neurons will fire at exactly the same time consistently and their action potentials will always be equal.

However, if one of these neurons has a few pulses interrupted just after it has fired, its firing will become out of phase with the firing of the other neuron. The neuron's frequencies, and describing characteristics will all be the same, but one will fire out of phase with the other.

Suppose two out of phase neurons are used with one of the neurons setting an output high when it fires and the other neuron setting the output low when it fires, as shown in Figure 3. The result can be used as a pulse width modulating drive.

To set the phase shift of the two neurons' firing rates, the impulses going to both synchronously have to be interrupted, or inhibited for one of the two neurons. This can be done either by sending inhibitory pulses, or by extending the refractory period of the neuron.

This can be achieved by the CNN in Figure 5. The PWM output is derived from a Set-Reset Latch, where one CN sets the output high, the other resets it.

This method is dependent on the number of impulses required to fire the neurons, the pulse width can only be changed by a discrete amount. Higher resolutions of pulse width modulation would require faster processor speed.

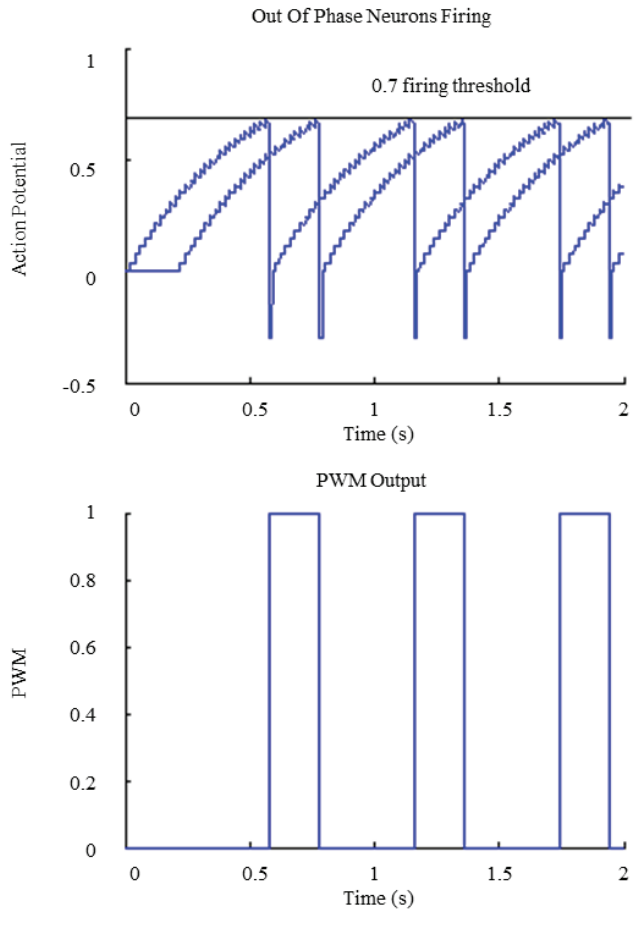

Figure 4. Pulse width modulation generated by two phase shifted CNs.

4. Learning Algorithms

Before these networks can be implemented, methods are needed to configure a network and set parameters. So, as an experiment into neuroplasticity, two learning algorithms have been produced for the cavalcade neural network. These two algorithms rely on being given a set of inputs with their respective outputs and, with a number of randomly placed neurons, would try to match the input data to the output data.

A simulation environment was designed and programmed in C++ to generate a graphical representation of the neural network. The OpenGL software library was used to create a 3-dimensional space in which individual neurons and axons are placed. This allows the structure and the functioning of the neural networked to be viewed and studied as it learns various problems in order to test the outcomes of the experimental learning algorithms.

Although the swarm robot developed in the project associated with this work does not have sufficient processing capabilities to run the learning algorithms onboard, the results of the learning algorithms have been used as inspiration for the control systems in the final robot control system.

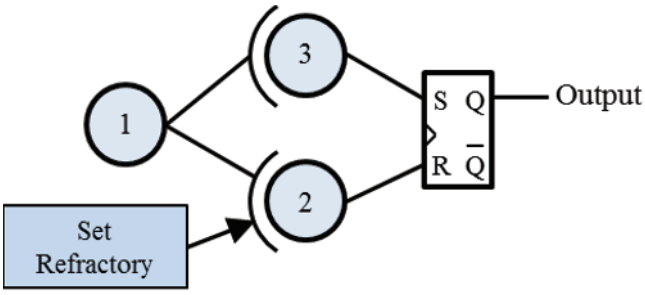

Figure 5. *CNN diagram of PWM generating network, the block "set refractory" causes neuron 2's refractory period of be altered, this putting it's post-synaptic spikes out of phase with the firing of neuron 3.*

4.1. Axon Genesis

The idea of the CNN learning algorithms is to start with no connections (axons) between any neurons, and the algorithms then generate the required axons and configure their strengths to generate the desired outputs. The algorithm to generate the axons for a given problem is known as the 'axon genesis' algorithm.

The algorithm works by assigning values of 'supply' and 'demand' to input and output neurons, the 'demanding' neurons are connected to 'supplying' neurons in order to spread the flow of pulses throughout the network. Supply and demand are time based functions so if a neuron has a high supply or a demand at one moment in time, that value will then decay if no addition 'supplies' or 'demands' are introduced to those neurons. The 'supply' value is proportional to the firing frequency of a neuron and 'demand' is proportional to the desired firing frequency of that neuron.

The condition under which an axon is created can be defined with the following, where N_x denotes neuron x, $Level_{xs}$ and $Level_{xd}$ are the supply and demand values for neuron x, and T_j is the 'joining threshold', C_{nd} is the contribution due to the distance between neurons, C_{np} is the contribution from neuron SDPD, and SDPD is the difference between the target neuron's supply and its demand.

$$C_{nd} = e^{-\frac{(distance\ N_1\ to\ N_2)^2}{k}} \tag{10}$$

$$C_{np} = Level_{2s} - Level_{2d} \tag{11}$$

The algorithm is then

$$\text{If } C_{nd} \cdot C_{np} > T_j \text{ then}$$
$$\text{join the two neurons with an inhibitory neuron}$$

$$\text{If } C_{nd} \cdot C_{np} < -T_j \text{ then}$$
$$\text{join the two neurons with an excitory neuron}$$

In equations 10 and 11, the contribution from the distance between the two neurons and the supply value minus the demand value are calculated. These two values are multiplied together and if they are larger or smaller than a 'joining threshold' then the axon will be created either as an inhibitory or excitatory input respectively.

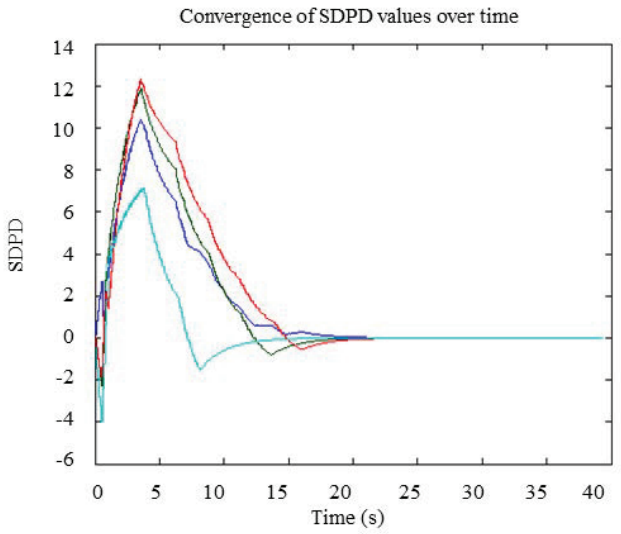

Figure 6. *Convergence of SDPD to 0 of 4 neurons producing the frequencies 8.33 Hz (light blue), 10 Hz (green), 16.6 Hz (red), and average SDPD (red).*

4.2. Axon plasticity

After the axon genesis algorithm has connected two neurons, the strength of that axon α then needs to be adjusted in order to equalize the supply and demand of its target neuron. If the supply to the target neuron is larger than the demand, then the axon's strength can be reduced. However if it is too little, and the axon's strength α is already larger than the firing threshold of the target neuron, then that neuron will require supply from another nearby neuron as the supplying neuron cannot provide any more stimulation.

$$\alpha_{fm} = \text{ axon fire magnitude}$$
$$\text{SDPD}_{\text{target}} = Level_{2s} - Level_{2d}$$
$$\alpha_{fm} \leftarrow \alpha_{fm}(1 - 0.01\alpha_{fm} \tanh(\text{SDPD}_{\text{target}}))$$

The axon plasticity algorithm tunes the weights of the axons generated in order to reduce SDPD to 0, or to reduce it enough to make it the change in axon magnitude negligible.

4.3. Applying the learning algorithms

The learning algorithms are able to solve basic frequency based problems such as producing 8.33 Hz, 10 Hz and 16.6 Hz on three different output neurons with an input neuron firing at 50 Hz. These only require single axons between the input and output layers, which makes the solutions very simple. This is shown in Figure 6.

Initially a large rise in average SDPD caused by demand values rising in the output neurons, and no output spikes to produce the supply to offset this occurs. As the axonal genesis algorithm creates axons, shown in Figure 6 at about 5 seconds, to spread the input spikes to the output layer, the SDPD starts to dramatically reduce and the axon plasticity algorithm tunes the newly created axons to produce the correct post-synaptic spiking.

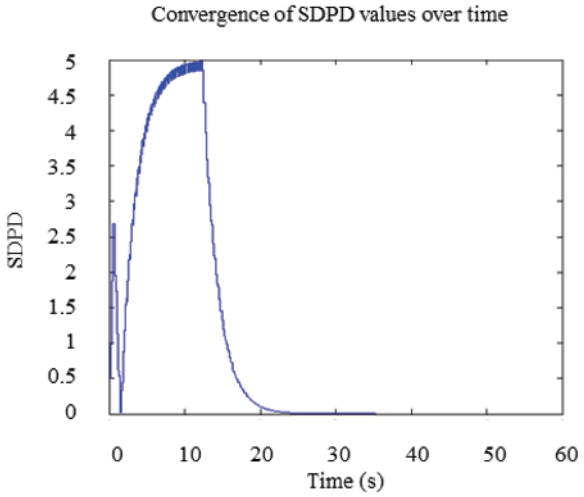

Figure 7. *CNN showing SDPD reducing to 0, (learning the XOR problem).*

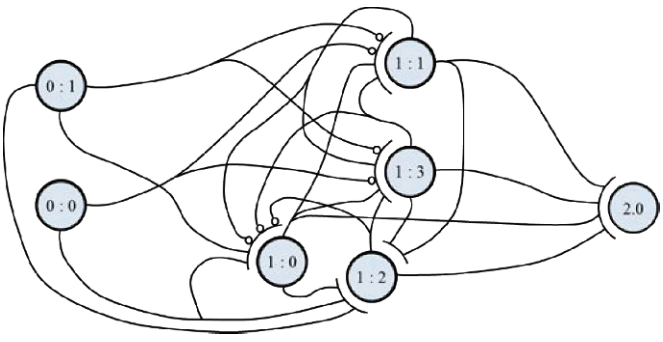

Figure 8. *Diagram showing the complex network generated using the two learning algorithms to solve the XOR Convergence of SDPD values can also be oscillatory, for example figure 10, is the SDPD plot of and attempt to solved the XOR problem, but some of the states of the XOR fail to be trained causing the oscillations in SDPD shown.*

The algorithms are also able to solve temporal and sequential based problems such as a finite state machine and a staggered XOR problem, where the XOR states were presented at specified intervals.
The initial SDPD rise and consequent SDPD fall can be seen in Figure 7, and is characteristic of the axon genesis and axon plasticity algorithms (a similar pattern is shown in Figure 6). Figure 8 shows the corresponding neural network generated by the axon genesis algorithm solving the XOR problem.
However, the axon plasticity and axon genesis algorithms rely on initial positioning of the neurons. This in turn means some generated axons can in fact be detrimental to minimising SDPD levels in the network. For example, Figure 9 shows the axon genesis and plasticity algorithms trying to solve the same 3 frequency problem described previously, however, 20 neurons are randomly placed between the input and output layers. After the intitial axons are generated by the genesis algorithm, the axon plasticity algorithm fails to succeed in causing the values of SDPD to converge to 0, leaving the network in an untrained state.

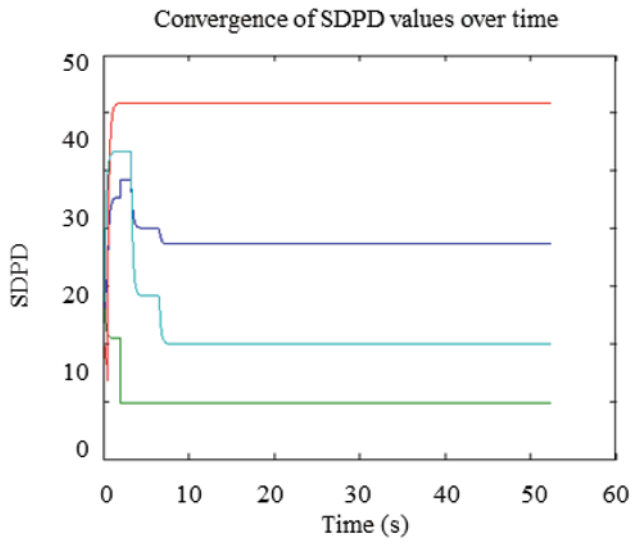

Figure 9. *Axon genesis and axon plasticity alorithm failing to learn the 3 frequency problem.*

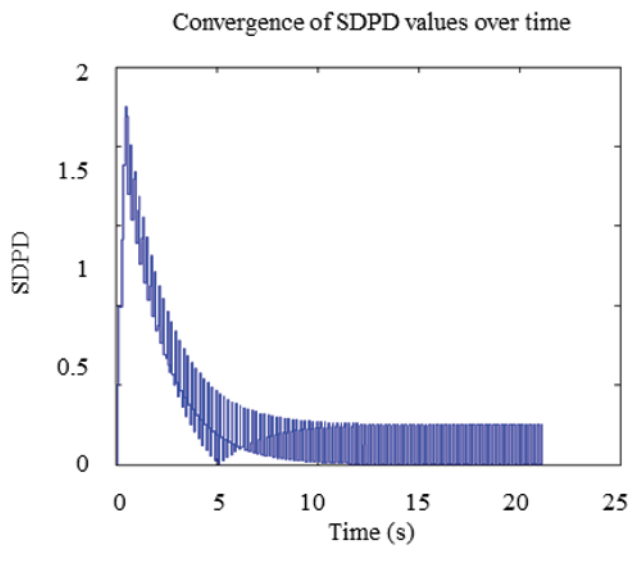

Figure 10. *Failure to fully converge all 4 Xor states, causing oscillations in SDPD.*

5. CNN on ROBOT

A CNN has been developed to control a robot. In the following it is shown how the network allows the robot to explore its environment, avoiding obstacles, are to control the speed of its motors. First, however, the robot used is briefly described.
Figure 11 shows a picture of the robot developed as part of the integrated masters project. The robot has caterpillar tracks as it was

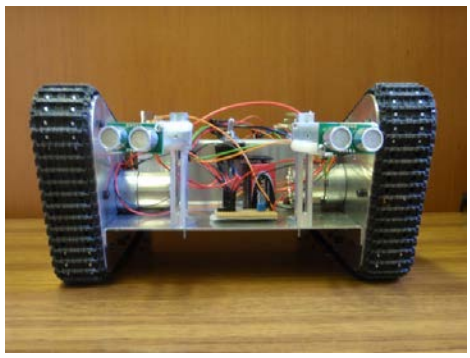

Figure 11. *Robot Used in Project.*

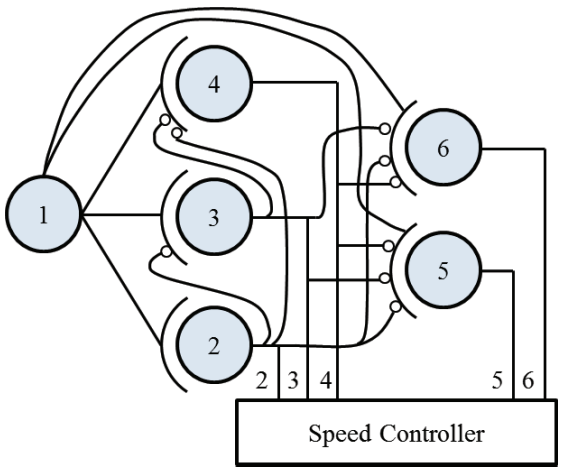

Figure 12. *CNN speed control network.*

designed to move over variable terrain. These are driven by wheels whose speed is set using pulse width modulation. The robot has an on board computer and is able to communicate with other machines using ZigBee networking technology. In addition it had ultrasonic sensors for detecting near by objects.

The robot's ultrasonic sensors are based on the SRF05 module, When triggered, the module sends an ultrasonic burst and waits until the burst is received back at the module's sensor. It then causes a pre-synaptic spike in an input neuron. This process continues, so the sensor is in fact sending a series of pulses whose frequency is roughly inversely proportional to the distance between an object and the robot. These pulses are then fed into a set of CNs as a series of impulses at fixed amplitudes and their parameters adjusted so they fire at different distances. These fires can then be used to feed directly back into the motor control's pulse width control neurons and therefore slow the Robot down when it is close to walls.

Figure 12 shows a CNN network set up for controlling one motor of a robot from a pulse source CN_1 firing at a rate inversely proportional to the distance of a sensor to a wall.

There are 5 CNs receiving these pulses, the network of axons creates a low level subsumptive architecture where only one neuron fires at a time, depending on the frequency of stimulation. At very low frequencies neuron 6 will be firing setting the motor speed on full. At closer distances, neuron 5 will fire slowing the motor speed slightly, at even lower frequencies, neuron 4 will fire slowing the motor still, but also stopping the motor from speeding up again by inhibiting 6 and 5, this process continues until 2 fires inhibiting all other neurons. CN_2 also causes the motor to change direction, thus causing the robot to move in the opposite direction of a wall. When the robot has moved a large enough distance from the wall, two will then stop firing and inhibiting the other neurons and they will start firing again in sequence, until 6 begins to fire, at which the motor will change direction back to forwards.

Full control of the robot is then achieved as follows. Two networks like those in Figure 12 are used, one for the left ultrasonic sensor, one for the right. In addition, the speed control values are used to generate the PWM signals for the robots left and right motors, based on the network in Figure 5. Finally, a forward/backward direction signal is required by each motor, and that is achieved by outputs from a CN setting or resetting a latch. The full system is in Figure 13.

The system has been implemented and during testing the robot was shown to show very effective wall avoiding capabilities. The robot also demonstrated its ability to successfully navigate closed environments without producing repetitive predictable behaviors.

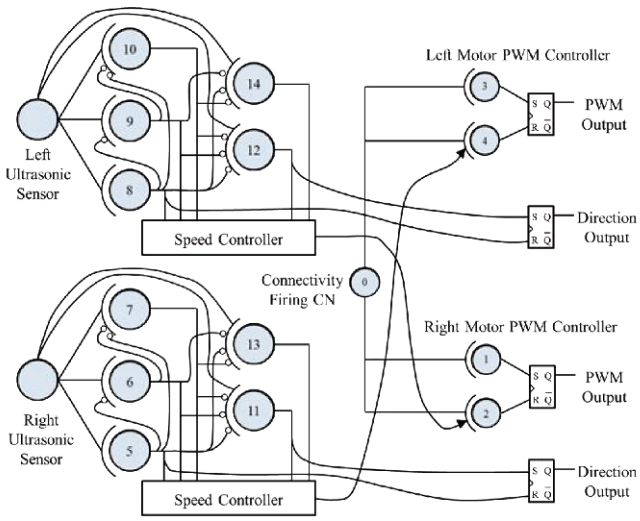

Figure 13. *Set of CNNs for controlling the robot.*

It was decided that due to the lack of on-board processing capabilities of the robot, that the two learning algorithms described in this paper would not be feasisble in real-time. If training were to be accomplished on the robot, a supervised learning technique would need to be implemented in order to apply the "demand" values to the outputs of the network. If a network such as the one described in Figure 13 were to be achieved, each module would need to be developed and trained separately in order to successfully generate the required axons and tune them to the frequencies needed (for pulse width modulation and speed controlling). It would be possible to do this in a simulated environment and then itertively build a train each module of the network. The networks could then be exported to the robot itself.

6. Conclusion

The use of a novel neural network in order to process sensory information in the form of pulses at different frequencies has been outlined. It has been shown that data from ultrasonic sensors and rotary encoders can be fed directly into the CNN with a very small amount of software pre-processing. It has also been shown that CNs can be used in order to produce useful outputs such as pulse width modulation.
The proposed axon genesis and axon plasticity algorithms worked on a range of problems as a supervised learning technique. However these algorithms relied heavily on initial conditions, making a successful outcome somewhat unreliable. Techniques could be employed in future research to improve this. Forgetting could be employed in these circcumstances in order to decrease the number of redundant or detrimental axon connections. Genetic algorithms could also be used to determine the structure of the networks instead of the real-time axon genesis.
Due to the limitation of the processor used on the robot chassis for control, the learning algorithms developed could not be implemented while the robot was navigating its environment. If more expensive processing hardware was implemented this could be possible and the robot could adapt its drive mechanisms, sensitivity to objects and speed control real-time. The adaptivity of the neural network due to the learning algorithms proposed could prove useful in navigating in unknown environments, future work will be needed to explore this possibility.
Applications for modular CNN systems are very wide ranging as complex modelling is not required. This research has shown that under certain circumstances the cavalcade neural network system and developed algorithms can solve various problems and provide a control system for a robot in an environment with very little computational power.

Acknowledgements

The authors wish to thank the rest of the Project group, Andrew Groom and Josh Homerston, for their contribution to the project.

References

[1] P. Tomei, Adaptive PD controller for robot manipulators, IEEE Transactions on Robotics and Automation, vol. 7, no. 4, (1991), 565–570
[2] Bing Hao and Xuefeng Dai, The collision-free motion of robot with Fuzzy neural network , in International Conference on Industrial and Information Systems., (2010), 219–222
[3] Dong-Sun Park, Sok Yoon, and YoungBu Kim, Robot end-effector recognition using modular neural network for autonomous control, in International Joint Conference on Neural Networks, (1999), 2032–2036
[4] E. T. Rolls, S. M. Stringer, Learning transform invariant object recognition in the visual system with multiple stimuli present during training, Neural Networks, vol. 21, no. 7, (2008), 888–903
[5] J. M. Tromans, S. M Stringer, E. T. Rolls, Spatial scene representations formed by self-organizing learning in a hippocampal extension of the ventral visual system, Eur. J. Neurosci., vol. 28, no. 10, (2008), 2116–27
[6] Y. S. Xia, Gang Feng, and Jun Wang, A primal-dual neural network for online resolving constrained kinematic redundancy in robot motion control, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 35, no. 1, (2005), 54–64
[7] Xiang Yu, S.X. Yang, and M. Ishikawa, Neural network robot controller based on structural learning with forgetting, in IEEE International Symposium on Intelligent Control., (2003), 264–268
[8] C. D. Bamford and R. J. Mitchell, Cavacade Neural Network For Mobile Robot, Cybernetic Intelligent Systems (CIS), 2010 IEEE 9th International Conference on, (2010), 1–6
[9] D. H., Macready, W. G. Wolpert, No Free Lunch Theorems for Optimization, IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, (1997)
[10] Xianyi Yang and M. Meng, A neural network approach to real-time motion planning and control of robot manipulators, in Systems, Man, and Cybernetics, (1999), 674–679
[11] S. X. Yang, A biological inspired neural network approach to real-time collision-free motion planning of a nonholonomic car-like robot, in International Conference on Intelligent Robots and Systems IROS, (2000), 239–244
[12] F. Alnajjar, I. Bin Mohd Zin, and K. Murase, A Spiking Neural Network with dynamic memory for a real autonomous mobile robot in dynamic environment, in IEEE International Joint Conference on Neural Networks, (2008), 2207–2213
[13] R. -J. Wai, Y. -C. Huang, Z. -W. Yang, and C. -Y. Shih, Adaptive fuzzy-neural-network velocity sensorless control for robot manipulator position tracking, Control Theory & Applications, IET, vol. 4, no. 6, (2010), 1079–1093
[14] Y. H. Kim and F. L. Lewis, Neural network output feedback control of robot manipulators, in IEEE Transactions on Robotics and Automation, (1999), 301–309
[15] M. D. Capuozzo and D. L. Livingston, A compact evolutionary algorithm for integer spiking neural network robot controllers, in Southeastcon, 2011 Proceedings of IEEE, Baltimore, (2011), 237–242
[16] F. Alnajjar and K. Murase, Sensor-fusion in spiking neural network that generates autonomous behavior in real mobile robot, in IEEE International Joint Conference on Neural Networks. , (2008), 2200–2206
[17] A. Schreibman,M. Häusser, M. E. Larkum,I. Segev, and M. London, The information efficacy of a Synapse, Nature Neuroscience, vol. 5, (2002), 332–340
[18] W. L. Kath, N. Spruston, Dendritic Arithmetic, Nature Neuroscience, vol. 7, no. 1, (2004),2004 567–569