Control of a Muscle Actuated Manipulator using the NeuraBASE Network Model

Robert Hercus, Kit-Yee Wong, and Kim-Fong Ho
Neuramatix Sdn Bhd, Kuala Lumpur, Malaysia
Email: {hercus, kityee, kfho}@neuramatix.com

***Abstract*— This paper presents an alternative approach for the control of an antagonistic muscle actuated manipulator. The proposed method uses a neuronal network called NeuraBase to learn the sensor events obtained via a rotary encoder and to control the motor events of two DC motors, to rotate the manipulator. A neuron layer called the controller network links the sensor neuron events to the motor neurons. The proposed NeuraBase network model (NNM) has demonstrated its ability to successfully control the antagonistic muscle manipulator, in the absence of a dynamic model and theoretical control methods. The controller also demonstrated its robustness in the adaptive learning of control with imposed system changes.**

***Index* Terms—neural network, antagonistic muscle, muscle actuator, control.**

I. Introduction

To date, various types of artificial muscles are available in the market. The most noticeable muscle type is the pneumatic artificial muscle (PAM) – a contractile device operated by pressurized air or hydraulic material. The muscle serves as an actuator which transfers the pressure exerted on the inner surface of the bladder into the muscle length compression. The working operation of the pneumatic artificial muscle is well documented in [1]-[5].

There are generally three categories of controllers namely dynamic modelling and control, machine learning and hybrid method. In dynamic modelling and control, controllers using the methods of Proportional-Integral-Derivative (PID), H-infinity control method, sliding mode control and Linear-Quadratic-Gaussian (LQG), have been previously applied to the control of artificial muscle actuated robots [6]-[11]. Although most of these approaches achieved good performance, an accurate mathematical representation for the nonlinear muscle robot is not easily formulated as extensive knowledge of system dynamics is required. Machine learning approaches such as fuzzy logic, artificial neural networks, neuro-fuzzy, recurrent neural networks, cerebellar model articulation controller (CMAC) and echo state networks have also been studied extensively [12]-[18]. The capabilities of these machine learning methods in mapping non-linearity and in dealing with uncertainties in system parameters, eliminates the need for exact mathematical models. The third category consists of hybrids of the previously described two methods [19]-[23]. They combine the essence of both methods hence the hybrid controller has good stability control as well as non-linearity mapping feature of the system input and output.

In this paper, the trajectory control of a muscle actuated robotic manipulator using a temporal-based neural network model named NeuraBase [24] is introduced. Our objective is to be able to activate the muscles such that the manipulator/joint moves to a desired position. The proposed NeuraBase controller does not require any mathematical model and its online training attribute allows it to continuously learn to adapt in a changing environment. The successive interval halving method was used to define the rotational movement to the target position of the antagonistic muscle manipulator and the NeuraBase controller learns to follow the desired trajectory with least errors. Furthermore, the NeuraBase controller has to learn to respond according to various rotational movements, whereby longer movements are executed when the muscle manipulator is far away from the desired target and shorter movements are taken as the muscle manipulator draws near to its desired target.

In this paper, the implementation is not intended to make direct comparisons with other muscle robot control methods, but to introduce an alternative and novel approach. The NeuraBase generic toolbox can be downloaded at [25].

This paper is organized as follows. Section 2 of this paper describes the usage of the NeuraBase Network Model (NNM) as a controller. In section 3, the experimental test setup is described. Section 4 explains the online learning logic used with the NNM, and in section 5 experimental results and a discussion are presented to evaluate the performance of the proposed NNM controller in terms of trajectory control.

II. Neura Base Network Model

The NNM is a network data structure that can store sequences of events. As shown in Fig. 1a, the neurons in a NNM can be associated temporally or spatially. The basic unit is an elementary neuron. Each neuron represents an event. Two neurons can be joined to represent a sequence of sensor or motor events. The way events are constructed

Manuscript received August 15, 2013 ; revised November 20, 2013.

doi: 10.12720/joace.2.3.302-309

in the NNM provides for fast searching and matching. For instance, as shown in Fig. 1b, if each character is represented by a sensor neuron (Level 1), the association of overlapping sensory events can represent words.

The proposed NNM controller for the muscle manipulator consists of three distinct and fundamental networks as shown in Fig. 2. The same NNM controller has been applied in the balancing control of inverted pendulum [26] as well as the navigation control of UAV [27].

These three networks store different types of events, namely a) sensor neurons and events - input to the system (the readout from the encoder); b) motor neurons - outputs from the system (the motor voltage for driving the DC motor); c) interneurons - association between two sensor events (to form a linked sensor neuron), or between a sensor event and a motor action (to form a controller neuron). Each type of event builds up an association of events in their respective network. The sensor network, motor network and the interneuron network store sensor neurons and events, motor neurons and events, and interneurons associations respectively. A simplified data structure of the neurons used in NNM is described in Table I. More detailed descriptions of the sensor, motor and controller neurons are provided in Section 3.

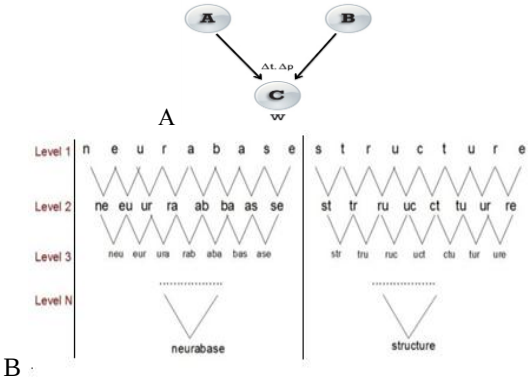

Figure 1. a) The NeuraBase Network Model where *t* denotes time proximity and *p* denotes spatial proximity; b) Words as sequences of events constructed using characters.

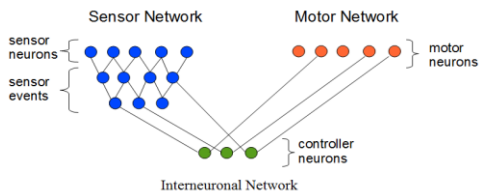

Figure 2. The fundamental architecture of the network of NeuraBase used in the trajectory control of the muscle manipulator.

TABLE I. DATA STRUCTURE OF A NEURON (BASIC)

Field	Data Type
Head	unsigned int
Tail	unsigned int
Successor	unsigned int
Frequency/ Weight*	signed int
Next	unsigned int
Overshoot/Undershoot Flags*	unsigned short

* Denotes Fields only applicable to the Controller Neuron

III. EXPERIMENTAL SETUP

The muscle manipulator is an antagonistic muscle mechanism emulating a simplified version of the human limb. With one of the links permanently attached to the base, the muscle mechanism can be considered a planar R-manipulator. The joint displacement of the muscle manipulator is controlled by two separate DC geared motors connected antagonistically to the two links via belts or "tendons". Each of these tendons has one end fixed to the DC motor in one link of the manipulator, and the other permanently attached to a specific point on the manipulator's adjacent link. Fig. 3 depicts the system components of the experimental setup and Fig. 4 illustrates the CAD drawing of the muscle robot.

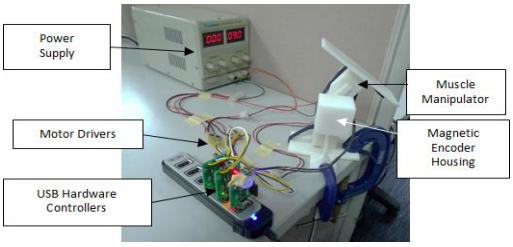

Figure 3. System components of the experimental setup.

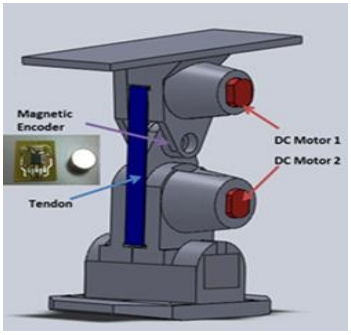

Figure. 4. The CAD drawing of the muscle robot.

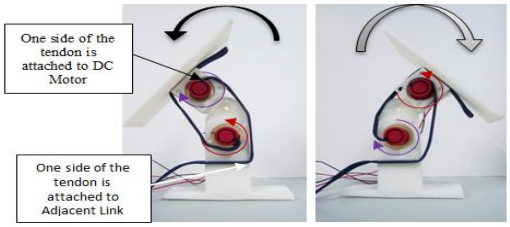

Figure. 5. The muscle manipulator's joint rotation mechanism for clockwise and counter-clockwise motions.

Fig. 5 shows the muscle manipulator's joint rotation mechanism for clockwise and counter-clockwise motions. The counter-clockwise rotation of the muscle manipulator requires the bottom DC motor to rotate in a counter-clockwise direction (see Fig. 5a), winding the belt and therefore contracting the left-side muscle. The top DC motor needs to rotate in a clockwise direction, unwinding the attached belt and therefore relaxing the left-side muscle. The opposite logic applies for rotating the muscle manipulator in the clockwise direction (see Fig. 5b). One assumption made in this system is that the elasticity of the

belts are negligible, hence any torque applied by the DC motor will be transferred directly to the adjacent links. The absolute angle of the joint is obtained through a rotary encoder attached to the muscle joint.

The muscle manipulator outputs its joint angular position, and accepts motor voltages as commands to rotate the joint. The joint angular position output has the following features:

- Measured via a magnetic encoder with a resolution of up to ≈1.40625°
- Sampling time as fast as 15ms
- The bottom position is 0°
- The upright position is 180°
- Angle wraps around 360 °counter-clockwise
- Movement limited to [130 °230 °] (±50 °away from the upright position)

The motor's commands include the following features:

- Motor voltage (-9V to 9V)
- From 0 to 1024 for rotation to the left (counter-clockwise)
- From 0 to -1024 for rotation to the right (clockwise)
- Each command unit corresponds to (9/1024)V

A 5-Layer NNM architecture was formulated and depicted in Fig. 6 as the controller for the muscle manipulator. It is derived based on the 3-layers architecture in Fig. 2. The 5-layer NNM architecture consists of the following layers:

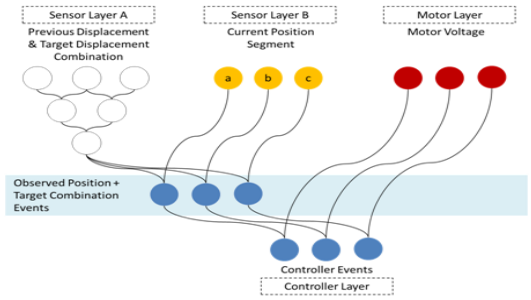

Figure. 6. The Architecture of the 5-Layer NeuraBase used in the application of Muscle Control.

A. Sensor Layer A

This layer contains events that represent the combination of observed variables of the past displacements and the target displacement of the muscle. The displacement is first translated into segments of 1.40625°/segment, and then further rounded to reduce neuron consumption:

-80	-40	-20	-10	-5	-2	-1	0	1	2	5	10	20	40	80

For instance, segment numbers between ±3 and ±5 are rounded up to segment number ±5. Segment numbers ±41 to ±80 are rounded up to the segment number ±80. Consequently, there are 15 sensor neurons in this layer. This layer builds patterns up to a maximum depth of 3; in other words, a maximum of two past displacements can be kept as a sequence while the last element is occupied by the target displacement. The purpose of using displacements instead of absolute angular positions to encode sensor event history is to maintain the relativity of all sensor elements in the sensor layer.

B. Sensor Layer B

This layer contains events that represent the muscle joint's current angular position. This variable serves to represent an awareness of gravity affecting the muscle at different joint positions. As with sensor layer A, the joint position is initially translated into segments of 1.40625 °segment. However, a different rounding adjustment is applied to the segment number – obtained values are rounded to the nearest 3 segments.

...	114	117	120	123	126	129	132	135	138	141	144	...

For instance, segment numbers 137 to 139 will be rounded to 138, segment numbers 149 to 151 will be rounded to 150, thereby giving an effective segment resolution of 4.21875°. Again, the purpose for this proposed rounding scheme is to reduce neuron consumption, which in this case results in the sensor layer having less than 40 sensor neurons. This layer only consists of patterns of length 1 as the history of motion has already been encoded in sensor layer A.

C. Target State Interneuron Layer

The target state event represents the link between sensor layers A and B (joint's current angular position linked to the sequence of its angular displacements). The nodes in this layer are linked nodes, representing nodes with heads and tails from two different layers (Sensor Layer A – Sensor Layer B).

D. Motor Layer

Each motor event represents a motor voltage, which corresponds to the resultant torque exerted on the muscle. This layer consists of patterns of length 1; representing the single motor action associated with each controller event. Each motor segment corresponds to an output voltage of $(9 \times n/32)$ V to the motor, therefore there are 65 motor neurons in this layer. Due to friction induced by the drive belts which hampers the muscle's ability to contract in the opposite direction, the theoretically non-actuating motor has to be given a constant counter-friction voltage of -1.35V.

E. Controller Layer

The controller event represents the link between the sensor combination (target state) layer and motor layer (the voltage required of the motor to move the muscle joint to the target angular displacement given its current angular position sequence). The nodes in this layer are linked nodes, representing nodes with heads and tails from two different layers (Target State Inter-neuronal Layer - Motor Layer).

Fig. 7A depicts the neuron structure of a sample sensor layer A neuron (named *C*) within the sensor network representing the sequence of sensor values $C = \{20, 10, 2\}$. The neuron *A* (head of *C*) represents the sensor sequence of {20, 10} and the neuron *B* (tail of *C*) represents the sensor sequence of {10, 2}. Neuron *D* represents a set of adjacent

neurons of *C* which can be a sequence of sensor segments {20, 10, 5} and *E* represents the target state interneuron for *C* linking to a specific sensor layer B neuron. Fig. 7B depicts the neuron structure of a sample sensor layer B neuron (*F*) within the sensor B network. The neuron *G* (next of *F*) contains the next sensor neuron within the same network.

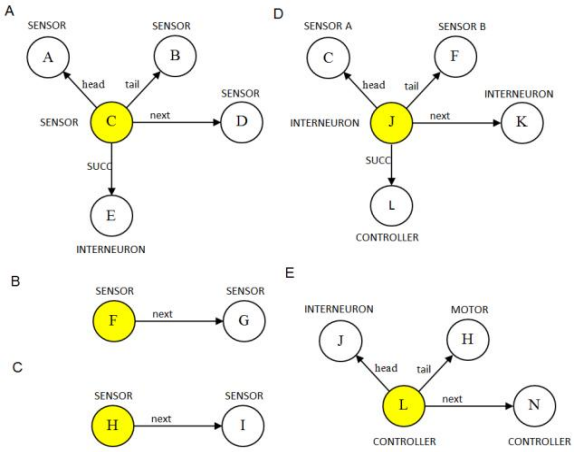

Figure 7. A) The structure of a sensor A event; B) The structure of a sensor B event; C) The structure of a motor event; D) The structure of a target state interneuron event; E) The structure of a controller neuron.

Each motor neuron represents an output voltage of $(9 \times n/32)$ V to the motor. Fig. 7C depicts the neuron structure of a sample motor neuron (*H*) within the motor network. The neuron *I* (next of *H*) contains the next motor neuron within the same network. Fig. 7D depicts the neuron structure of a sample target state interneuron (named *J*) within the target state interneuronal network and it represents the interneuron *J* = [{20, 10, 2}, 129]. The head of *J* is the neuron *C* (of sensor layer A), and its tail is the neuron *F* (of sensor layer B). Neuron *L* represents a specific controller neuron. Fig. 7E depicts the neuron structure of a sample controller neuron (*L*) within the controller network with the neuron *J* (target state interneuron) as its head and the neuron *H* (motor neuron) as its tail.

IV. Learning Logic

The long-term goals for the joint are set to be anywhere between ±50° from the upright position. For each long-term goal, there will be a set of short-term goals defining the target displacements for the muscle controller during each time interval. The short-term targets are set according to Table II (note that each displacement segment corresponds to a multiple of 1.40625°).

TABLE II. Data Structure of a Neuron (basic)

Remaining Distance to Goal (segments)	Short-term Target Distance (segments)
≥ 40	20
[20 40)	10
[10 20)	5
[5 10)	2
[2 5)	1
[1 2)	1

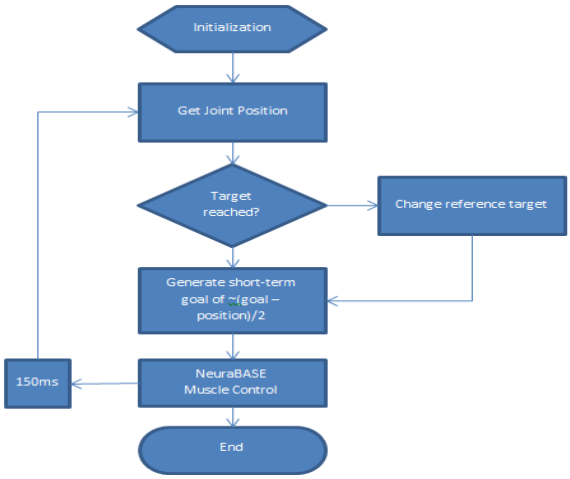

Figure. 8. The overall process flow of the NNM controller.

The overall process flow of the NNM controller is depicted in Fig. 8. Given a long-tern goal, the NNM first acquires the muscle manipulator's current joint position and validates whether the muscle manipulator has reached the target, if not, a new short term target will be computed using the interval halving method with the equation *newtarget* = (*oldtarget* – *currentposition*)/2. The control sampling time was set at 150ms. Referring to the Fig. 8, the NeuraBase muscle control block contains three main functions: predict, generate educated guess and feedback functions. In the predict function, the NNM controller will search for the matching motor patterns in the controller layer given the interneuron of the sequence of past displacements and the current short-term target (interneuron *J* in Fig. 7E). If a controller neuron exists in the controller layer which has a head pointed the aforementioned given *targetstate interneuron* (interneuron *J*), the NNM controller will retrieve the list of linked motor actions (the tail of the controller neuron *L*). Subsequently, the list of linked motor actions is sorted according to the linked frequency, and the motor action which has the highest frequency as well as greater than the preset threshold would be selected as the recommended motor action. If none of the linked motor actions passes the preset threshold, the generate educated guess function will be called upon to recommend a motor action interpolated from those found suitable for its neighboring positions. If all else fails, a random motor value bounded by the system's past experience of known overshooting and undershooting motor actions will be suggested.

Once the recommended force has been determined, the NNM controller will execute the motor action. The feedback function will then monitor the muscle manipulator's position to evaluate whether it has achieved the given target displacement after 150ms. If the objective has been met, the controller neuron linking the *targetstate interneuron* with motor action is given a reward of +3. On the contrary, if the actual displacement is less than the target displacement, the NNM controller penalizes the controller neuron by -2, as well as creating a new controller neuron which links the *targetstate interneuron* with [motor

action + 1] in the controller network and gives it an incentive of +4. Also, the NNM controller sets the undershoot flag of the controller neuron to *true*. On the other hand, if actual displacement is greater than the target displacement, the NNM controller penalizes the controller neuron by -2 and creates a new controller neuron which links the targetstate interneuron with [motor action - 1] in the controller network and gives it an incentive of +4. Also, the NNM controller sets the overshoot flag of the controller neuron to *true*.

The feedback function is a mechanism that strengthens controller actions that have successfully achieve the objectives set during a control cycle. Good controller actions that help the muscle controller achieve its objectives are rewarded. Bad controller actions that have failed to help the muscle controller achieve its objectives are penalised. The strength of each controller neuron is represented by its trained weight; hence, the controller neuron with the highest weight will be the most reliable prediction because it is the accumulated result of learning, using both positive and negative feedbacks from past trials. The association of the interneuron and motor neuron in the controller network is a reinforced learning process, whereby positive and negative feedbacks dictate how the learning takes place by tuning the weight of the controller neurons. A positive feedback is given if the muscle manipulator managed to reach within the target displacement, upon which, the weight of the controller neuron is incremented by 3. The more the controller neuron experiences positive feedbacks for its motor predictions, the stronger the link coupling will be, thereby resulting in a stronger positive memory of the respective motor action. A negative feedback is evoked if the muscle manipulator fails to reach the target displacement, upon which, the weight of the controller neuron is decremented by 2, thus reducing the coupling strength of that link. Alternatively, the NNM controller will create a new controller neuron linking the sensor event to the motor neuron. Eventually, a network of almost all possible sensor events associated with motor actions will be stored within NeuraBase, and the controller neurons linking sensor events to the right motor actions will have higher weights compared to those linking sensor events to incorrect motor actions.

This motor range is bounded by motor neurons linked to previously used controller neurons which have either been flagged as overshooting and/or undershooting controller neurons. Each time the overshoot/undershoot flag of a controller neuron is set, the range of potential solutions becomes smaller, with the controller neuron's linked motor neuron as the new upper or lower boundary of the set. This method helps the controller narrow down its choices of approximately correct motor actions for the sensor event more quickly, compared to the controller having to attempt all motor neurons before finding a suitable one.

V. Experimental Results

There are three performance metrics used in this experiment:

- Overshoot: intended to identify the muscle's ability to reach the goal segment without overshooting. The overshoot/undershoot is computed one time-step after the muscle enters or overshoots the coarse goal region (resolution of 4.2 °) for the very first time.
- Excess Motion: Intended to identify the controller's ability to bring the muscle joint to the middle of a goal segment and maintain its position for one second. Performance is gauged based the amount of excess distance travelled before completing a trial.
- Average Tracking Speed: intended to identify the NNM controller's ability to bring the muscle joint to its coarse goal segment while obeying the smooth trajectory short-term goal rules. This average speed is computed from the absolute angular distance between the initial and goal segments divided by the time taken to achieve this motion.

A. Training Without External Load

As shown in Fig. 9, it has been observed that with training, the muscle manipulator was able to reach the goal segment without overshooting much, eventually reaching an average overshoot of approximately 0.8° after 25,000 trials. However, it is worth noting that the curve had actually begun to saturate around 15,000 trials, wherein it recorded an average overshoot of 1.0°. The performance improvement can also be attributed to the NNM controller in managing to achieve the short-term goals defined with training, which had been set to ensure the muscle joint reaches the middle of its goal segment at low speeds. In the histogram presentation of the overshoot distribution (see Fig. 10), it has been observed that the overshoots ranged between ±1.40625° almost 80% of the time after training compared to a ~20% at the beginning of training. Overshooting between ±1.40625° is reasonable considering that position segments are stored at a resolution of 4.21875° in NNM.

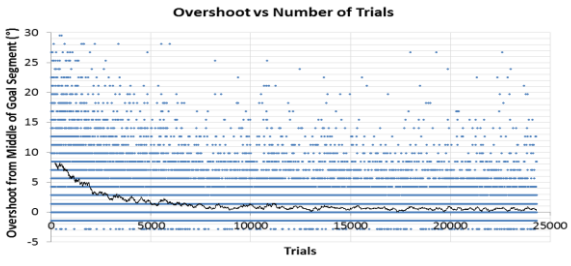

Figure 9. Plot of the muscle control system's excess motion for each trial. The blue dots represent the overshoots recorded in each trial, while the black line denotes a moving average curve of 255 for these points.

It has also been observed that the average excess motion decreased with training (see Fig. 12), which suggests that the amount of oscillations had been reduced as the muscle controller learns to effectively maintain the muscle within the goal region for 1 second. As with the overshoot plot, the excess motion performance appears to have begun saturating at the 15,000 trial mark with an average value of ~4.5° and finally recording an average of ~4° after almost

25,000 trials. The amount of neurons consumed during the training process has been observed to be high, at around 105,000. However, based on the plot in Fig. 11, the overall neuron growth appears to have slowed down considerably. The controller network takes up majority of the resources in NNM as its neuron consumption currently stands at about 70,000, approximately 6 times its distant second, which is the Target State Inter-neuronal Network with ~13,000 neurons.

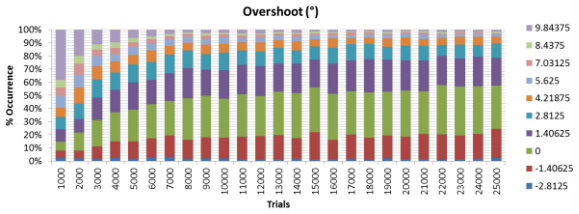

Figure 10. Histogram of the muscle control system's overshoot distribution every 1,000 trials. Each coloured bar represents the percentage of occurrence of its corresponding overshoot value. The maximum overshoot value is saturated to 9.84° in this chart.

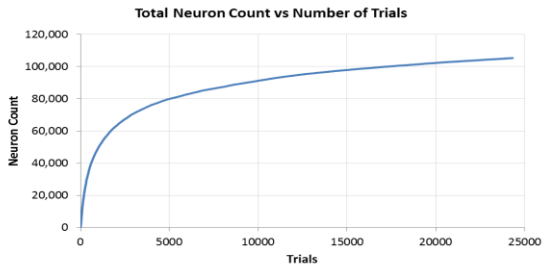

Figure 11. Total neuron usage vs. number of trials.

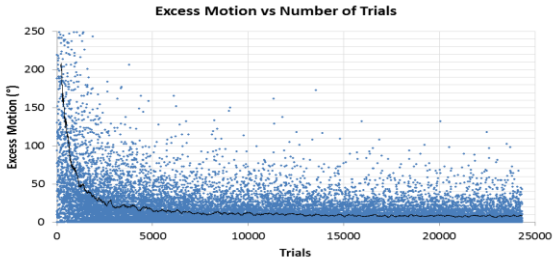

Figure 12. Plot of the muscle control system's excess motion for each trial. The blue dots represent the excess motions recorded in each trial, while the black line denotes a moving average curve of 255 for these points.

Based on the trajectory targets set, the muscle should exhibit higher average speeds when it is targeting goals far away compared to nearer ones. To illustrate this, results of average tracking speeds have been split into four categories, as shown in Fig. 13. Results suggest that the muscle controller managed to move the joint to its target segment in smoother motions with training. Compared to the beginning of training when the average muscle speeds were mostly around 70 °/s, it has been observed that the trained average muscle speeds are approximately ~30 °/s, ~40 °/s and ~50 °/s for targeting short, moderate and long distances respectively. As shown in Fig. 13, during the beginning of training, the motor actions used to bring the muscle to its target segment have not been trained yet. Therefore, especially for the short distance category, even though there already existed speed-distance patterns whereby the average speed appeared to be proportional to the target's distance, for most parts the muscle was moving at high speeds, introducing many overshoots in the process, which would be explained further in the next section. Towards the end of training, it has been observed that the average muscle speed for targeting short distance goals have drastically lowered to match the intended trajectory, whilst maintaining the speed-distance relationship. For one of the medium distances category (Fig. 13c), there was a shift in the average speed distribution's slope. This can be attributed to the muscle initially overshooting by large margins whenever it crossed the top position (segment 129 or 180 °), therefore the average muscle speeds for the shorter medium distances were higher. The controller later learned to reduce these occurences by significantly slowing down the muscle when it closes in on the target, which resulted in a flatter distribution slope, better resembling the one in Fig. 13a. A similar observation could be made for the other two categories, except that the changes in speed distribution as well as decreases in average speed do not appear as obvious. Overall, the positive relationship between the average muscle speed and the distance between the initial and target segments was maintained at the end of the training duration, however at much lower speeds compared to the first thousand trials. This suggests that the muscle controller has learned to better correspond to the short-term goals, where fast motions are executed when the muscle is far away from the target and slower ones are executed as the muscle draws near its target.

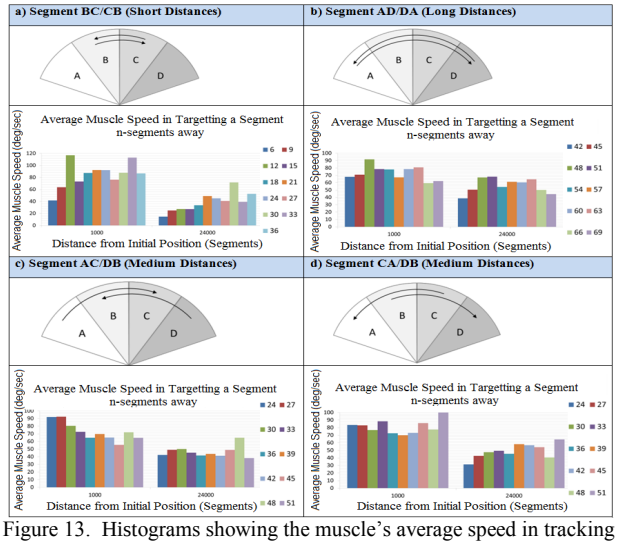

Figure 13. Histograms showing the muscle's average speed in tracking its target between 3 and 69 segments away regardless of its rotational direction for the first and final 1,000 trials of each training phase. Each coloured bar represents the average muscle speed recorded for the muscle to reach a goal located *n*-segments away from its initial position.

B. *Training with External Load*

Another experiment was performed to further demonstrate the effectiveness and adaptive behaviour of the NNM controller when changes were introduced to the muscle manipulator. We tested the NNM controller by attaching a 50gram weight to the centre of an 18gram rod at a distance of 13cm away from the joint. In a separate

experiment, a 50gram weight was attached to centre of a 35gram rod at a distance of 19cm away from the muscle joint. In this experiment, the muscle manipulator was executed at a slower speed compared to the former because sufficient torque is required to lift the attached weight. The experimental results are presented in Fig. 14-17. As shown in Fig. 14, due to the manipulator's generally slow speeds, it is not easy for the muscle to overshoot, though it has been observed that the occurrence was notably higher during the initial stages of training. After training though, additions of weights did not seem to have much effect, only recording negligibly small bumps in the averages which were then reduced once again after training. Therefore, the average overshoot stands at 0° after 15,000 trials. This improvement in performance, though slight, can also be attributed to NeuraBase managing to achieve the short-term goals defined with training, which have been set to ensure the muscle joint reaches the middle of its goal segment at low speeds. In the histogram presentation of the overshoot distribution as shown in Fig. 15, it has been observed that the overshoots ranged between ±1.40625° almost 100% of the time after training compared to a ~75% at the beginning of training. Overshooting between ±1.40625° is reasonable considering that position segments are stored at a resolution of 4.21875° in NeuraBase. Also, it can be seen that the addition of weights increased the amount of overshoots even though they are mostly only 1.40625 ° ones. However, with training the distribution returned to approximately its original distribution before the weights were added, which can be an evidence of the effectiveness of the training. The amount of neurons consumed during the training process has been observed to experience a slight bump with each training phase, as shown in Fig. 16. This situation can be explained by the change in the muscle's inertia when the weights were placed or shifted along the muscle link, which resulted in different sequences of the muscle joint positions, eventually encoded within the Sensor Layer A and propagated all the way down to the Controller Network. As shown in Fig. 17, the average excess motion did not appear to be significantly affected by the addition of weights partway through training, though it has been observed that the average did experience slight jumps when the weights were first placed or shifted on the hardware.

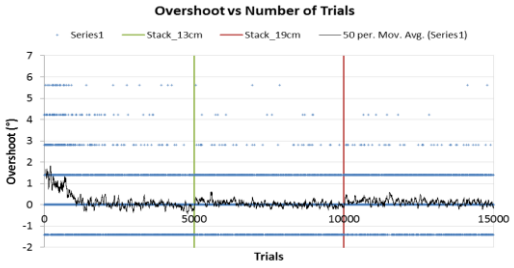

Figure 14. Plot of the muscle control system's excess motion for each trial. The blue dots represent the overshoots recorded in each trial, while the black line denotes a moving average curve of 50 for these points. The green line denotes the point when the weight is attached 13cm from the pivot whereas the red line denotes the point when the weight is attached 19cm from the pivot.

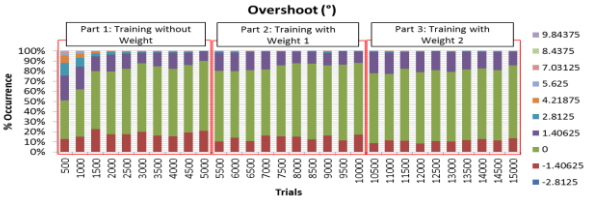

Figure 15. Histogram of the muscle control system's overshoot distribution every 1,000 trials. Each coloured bar represents the percentage of occurrence of its corresponding overshoot value. The maximum overshoot value is saturated to 9.84 °in this chart.

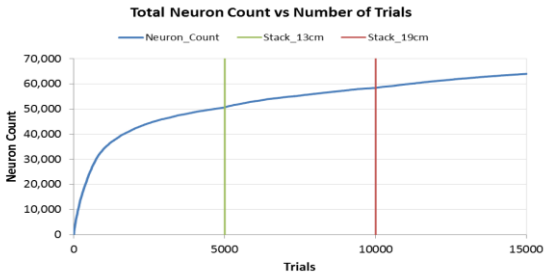

Figure 16. Total neuron usage vs. number of trials. The green line denotes the point when the weight is attached 13cm from the pivot whereas the red line denotes the point when the weight is attached 19cm from the pivot.

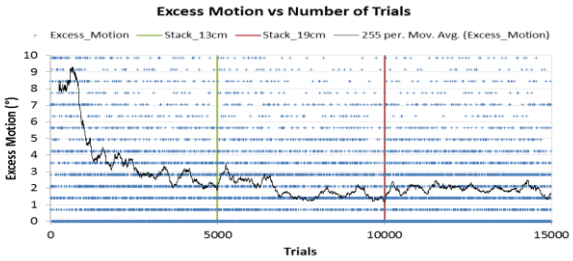

Figure. 17. Plot of the muscle control system's excess motion for each trial. The blue dots represent the excess motions recorded in each trial, while the black line denotes a moving average curve of 255 for these points.

REFERENCES

[1] C. P. Chou and B. Hannaford, "Measurement and modelling of McKibben pneumatic artificial muscles," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, pp. 90-102, 1996.

[2] F. Daerden and D. Lefeber, "Pneumatic artificial muscles: actuator for robotics and automation," *European Journal of Mechanical and Environmental Engineering*, vol. 47, pp. 10-21, 2002.

[3] D. G. Caldwell, A. Razak, and M. J. Goodwin, "Braided pneumatic muscle actuators," in *Proc. Conf. Rec. IFAC Conference on Intelligent Autonomous Vehicles*, 1993, pp. 507-512.

[4] B. Tondu and P. Lopez, "Modelling and control of McKibben artificial muscle robot actuator," *IEEE Control System Magazine*, vol. 20, pp. 15-38, 2000.

[5] M. Balara and A. Petik, "The properties of the actuators with pneumatic artificial muscles," *Journal of Cybernetics and Informatics*, vol. 4, pp. 1-15, 2004.

[6] A. Pujana-Arrese, A. Mendizabal, J. Arenas, R. Prestamero, and J. Landaluze, "Modelling in modelica and position control of a 1-DoF set-up powered by pneumatic muscles," *Mechatronics*, vol. 20, no. 5, pp. 535-552, 2010.

[7] H. P. H. Anh, "Online tuning gain scheduling MIMO neural PID control of the 2-axes pneumatic artificial muscle (PAM) robot arm," *Expert Systems with Applications*, vol. 37, no. 9, pp. 6547-6560, 2010.
[8] S. Oh, V. Salvucci, and Y. Hori, "Development of simplified statics of robot manipulator and optimized muscle torque distribution based on the statics," in *Proc. Conf. American Control Conference*, 2011, pp. 4099-4104.
[9] D. Mitrovic, S. Klanke, and S. Vijayakumar, "Adaptive optimal control for redundantly actuated arms," in *Proc. Conf. International Conference on Simulation of Adaptive Behavior*, 2008, pp. 93-102.
[10] M. Van Damme, B. Vanderborght, R. Van Damme, *et. al.*, "Proxy-Based sliding mode control of a manipulator actuated by Pleated Pneumatic Artificial Muscles," in *Proc .Conf. IEEE Internal Conference on Robotics and Automation*, 2007, pp. 4355-4360.
[11] G. Tao, X. Zhu, and J. C, "Modeling and controlling of parallel manipulator joint driven by pneumatic muscles," *Journal of Mechanical Engineering*, vol. 18, no. 4, pp. 537-541, 2005.
[12] H. Zhao and M. Sugisaka, "Simulation study of CMAC control for the robot joint actuated by McKibben muscles," *Applied Mathematics and Computation*, vol. 203, pp. 457-462, 2008.
[13] K. K. Ahn and H. P. H. Anh, "Design and implementation of an adaptive recurrent neural networks (ARNN) controller of the pneumatic artificial muscle (PAM) manipulator," *Mechatronics*, vol. 19, pp. 816-828, 2009.
[14] T. Leephakpreeda, "Fuzzy logic based PWM control and neural controlled-variable estimation of pneumatic artificial muscle actuators," *Expert Systems with Applications*, vol. 38, no. 6, pp. 7837-7850, 2011.
[15] K. Xing, Y. Wang, and Q. Zhu, "Modeling and control of McKibben artificial muscle enhanced with echo state networks," *Control Engineering Practice*, vol. 20, no. 5, pp. 477-488, 2012.
[16] L. D. Khoa and K. K. Ahn, "Synchronization algorithm for controlling 3-R planar parallel pneumatic artificial muscle robot," in *Proc. Conf. International Conference on Control, Automation and Systems*, 2011, pp. 1588-1593.
[17] T. Ozaki, T. Suzuki, T. Furuhashi, and S. Okuma, "Trajectory control of robotic manipulators using neural networks," *IEEE Transactions on Industrial Eletronics*, vol. 38, no. 3, pp. 195-202, 1911.
[18] R. Shadmehr, "Learning virtual equilibrium trajectories for control of a robot arm," *Neural Computation*, vol. 2, no. 4, pp. 436-446, 1990.
[19] T. D. C. Thanh and K. K. Ahn, "Nonlinear PID control to improve the control performance of 2 axes pneumatic artificial muscle manipulator using neural network," *Mechatronics*, vol. 16, no. 9, pp. 577-587, 2006.
[20] L. D. Khoa and K. K. Ahn, "Synchronization algorithm for controlling 3-R planar parallel pneumatic artificial muscle robot," in *Proc. Conf. Rec. Internal Conference on Control, Automation and Systems*, 2011, pp. 1588-1593.
[21] A. Rezoug, M. Hamerlain, and M. Tadjine, "Decentralized RBFNN type-2 fuzzy sliding mode controller for robot manipulator driven by artificial muscles," *International Journal of Advanced Robotic Systems*, vol. 9, pp. 1-12, 2012.
[22] X. Chang and J. H. Lilly, "Fuzzy control for pneumatic muscle tracking via evolutionary tuning," *Intelligent Automation and Soft Computing*, vol. 9, no. 4, pp. 227-244, 2003.
[23] T. D. C. Thanh and K. K. Ahn, "Nonlinear PID control to improve the control performance of 2 axes pneumatic artificial muscle manipulator using neural network," *Mechatronics*, vol. 16, no. 9, pp. 577-587, 2006.
[24] R. G. Hercus, "Neural networks with learning and expression capability," U.S. Patent 7412426 B2, 2008.
[25] NeuraBase Generic Toolbox. [Online]. Available: http://www.neuramatix.com
[26] R. Hercus, K.-Y. Wong, and K.-F. Ho, "Balancing of a simulated inverted pendulum using the neurabase network model," *LNCS*, vol. 8131, pp. 527-536, 2013.
[27] R. Hercus, H.-S. Kong, and K.-F. Ho, "Control of an unmanned aerial vehicle using a neuronal network," *LNCS*, ICONIP, Part II, vol. 8227, pp. 605–615, 2013.

Robert Hercus received the B.S. degree in information science from Monash University, Australia. He is an adjunct professor in the Department of Computer Science at the University of Malaysia Terengganu, Malaysia.
He has more than 40 years of experience in the field of information science and technology. He is the founder of Neuramatix Group of Companies and the inventor of NeuraBase. Robert is also the founder and Managing Director of Malaysian Genomics Resource Centre Berhad (MGRC).

Kit-Yee Wong received the B.E. degree in mechatronics engineering from Monash University Malaysia in 2011.
She joined the robotics department of Neuramatix Sdn. Bhd. as a mechatronics engineer and has been with them since 2012. Her current research interests include robotic manipulators, biped robots and intelligent systems.

Kim-Fong Ho received the B.S. degree in mechanical engineering from Utah State University, Logan, in 1998, and the M.E. degree in mechanical engineering in 2000 from the same university.
He is presently the Neural System Manager at the Neuramatix Sdn Bhd, Kuala Lumpur, Malaysia. His research interests include intelligent control of robotics, text mining and contextual mining.