A Pong Game Agent Using the Neuronal Network Model

Robert Hercus and Hong-Shim Kong
Neuramatix Sdn. Bhd., No. 27-9 Level 9, Signature Office Bandar Mid-Valley,
59200 Kuala Lumpur, Malaysia

Abstract - *This paper presents an alternative approach to intelligent agent development in video games through the use of the Neuronal Network Model (NNM). The NNM is an integrated network of multiple networks of neurons capable of learning from and adapting to its environment. The pong game is simulated to evaluate the feasibility of implementing the NNM as an agent controller. The goal of the controller is to place the agent which is the paddle, in the correct position to return the pong ball. The learning process of the model is based solely on its failures in predicting the correct motor actions to return the pong ball. Experimental results show that the NNM is capable of acting as an intelligent agent controller.*

Keywords: agent, control, game, neuronal network, pong

1 Introduction

Today, there is a huge market for the video games industry. In the United States alone, it was reported that consumers spent US$20.77 billion in 2012 [1] on video games, hardware and gaming accessories. Of these sales, 40 percent of them were purchases of digital content, which includes games, add-on content, mobile apps, subscriptions and social networking games. Video games are going through massive enhancements and changes driven by technology advancements. At present, games can be played on a variety of media such as dedicated consoles, personal computers, handheld devices and smartphones. At the same time, the demand for realistic gaming experiences has also increased. Players today expect games to be intelligent as well as engaging. It is only through such experience that a player's interest in the games can be continuously piqued and sustained.

In the pioneering stages of the video games industry, the agent or opponent in a game is often a preset computer program with finite strategies. Once these strategies have been discovered, the players can respond to them. As such, games begin to lose their appeal. Furthermore, if the difficulty in games is increased, the number and complexity of strategies may increase as well. Encoding strategies into these games can thus prove to be a daunting task.

On the other hand, when a game is played against another person, the games are more interactive and entertaining. Therefore, efforts have been made to develop believable, intelligent games. The advent of artificial intelligence (AI) technology provides solutions to this challenge. AI methods such as neural networks, Bayesian models and behaviour trees have been used in many ways to enhance a player's satisfaction in terms of gaming experience and entertainment value.

One such usage is the simulation of human-like intelligence and behaviour in an agent or non-player character (NPC). An NPC is an entity or character that is not controlled by a player in a game, such as the hostile entities in Far Cry 3 [2]. Far Cry 3 is a first-person shooter (FPS) game, wherein a player needs to combat numerous mercenaries. In this game, AI is programmed into the enemies so that they can react in many situations adaptively. For example, if the enemy is injured, he can shout for help or release emotional distress. Another example is the Drivatar opponent drivers in Forza Motorsport [3]. Drivatar employs neural networks to control drivers. Each driver is assigned a different neural network. As such, each driver has different skill levels and driving tendencies, such as how they approach a particular corner or how aggressively they try to overtake.

Matchmaking in online gaming is another example of the usage of AI in games. In Ghost Recon Online [4], an FPS game, a neural network has been used to select players from a pool of players with matching skills in a multiplayer online match. TrueSkill [5], which is based on Bayesian networks, has been developed by Microsoft for the Xbox Live online gaming for the same purpose as well. The matching is done by considering each player's profile, which includes information about the player's behaviour and personal preference. The information on players is derived from historical data, taking into account both previous match results and player attributes collected by tracking the players' behaviour over time. Neural networks make it easy to include additional parameters, and allow for continuous updating of the model in real time as new data is collected. A good matchmaking process can increase the chance of players having fun together, thus improving player retention.

AI can also be used to plan the path of agents in a game environment, such as in real time strategy (RTS) games [6], [7]. RTS games usually require positioning and movement of units and structures to secure game areas. When deciding the movement of units, the complexity of the terrain and obstacles may be taken into account by AI to decide future courses of action.

In recent years, game-AI contests such as Unreal Tournament [8] and StarCraft [9], have been held to apply artificial intelligence algorithms on commercial gaming platforms. Such contests offer researchers and developers a platform to evaluate their AI algorithms in robust game environments. For example, in the AIIDE (AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment) StarCraft AI tournament, AI approaches such as finite state machines (FSM), decision trees and probabilistic inferences have been employed for micromanagement and/ or macromanagement of agents in the game [10], [11].

Numerous researches on the development and application of artificial intelligence for games can be found in many publications. For example, Kotrajaras and Kumnoonsate presented a tool that uses genetic algorithms and steepest ascent hill-climbing to learn and adjust map properties [12]. In [13], Recio *et al.* have developed a generic framework based on swarm intelligence, specifically ant colony optimisation for controlling agents in a dynamic environment. These show that research on artificial intelligence in games is gaining momentum. Moreover, game environments provide a range of excellent platforms for fundamental AI research.

In this paper, the Neuronal Network Model (NNM) is introduced as an intelligent agent's controller in games. The NNM is a data structure consisting of interconnected nodes in a multi-level network. A group of networks in the model can be interconnected to perform a task. In this work, the feasibility of the NNM is demonstrated by its application in a simulated pong game. The objective of the NNM in the game is to predict the trajectory of the pong ball, and to actuate the paddle to the predicted final position that will result in the ball being returned successfully. The NNM is implemented in the NeuraBase toolbox, which can be downloaded at [14].

The remainder of the paper is organised as follows: Section 2 describes the fundamental design of NNM, followed by the description of the pong game and its NNM controller designs in Section 3. In Section 4, the learning algorithm of the model is presented. Section 5 analyses and discusses the experimental results. Finally, Section 6 presents conclusions and future work to be done.

2 Neuronal network model

The NNM is a multi-node, multi-level, and multi-network neuronal network which is based on a concept of how the human brain may work. The basic node in the network is termed an elementary neuron, which represents an event. A couple of elementary neurons can be associated temporally or spatially to form a sequence of events, such as shown in Fig. 1. A detailed description of the NNM is available in [15].

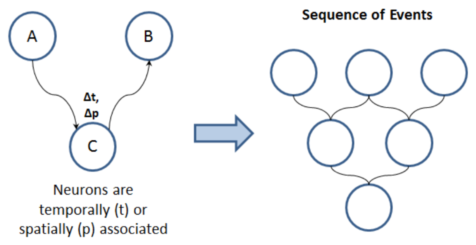

Figure 1. The NNM fundamental design

Generally, neurons can be classified into three categories: sensory neurons, motor neurons and controller neurons. Sensory neurons represent the data acquired by sensors, whereas motor neurons represent the motor actions that are performed in response to the sensory stimuli. Controller neurons associate a sensory neuron sequence to a motor neuron action.

A group of sensory or motor neurons can be associated in a multi-level network, and form a network of sensory or motor event sequences respectively. These networks represent sequences of events. In other words, a history of events is stored in the NNM. As for the controller neurons, they are grouped in a network known as an interneuronal network. The relationship between all the networks is illustrated in Fig. 2, which shows the general architecture of NNM. This general architecture has been applied in the balancing of an inverted pendulum [16], navigational control of an unmanned aerial vehicle [17] and control of an antagonistic muscle actuated manipulator [18].

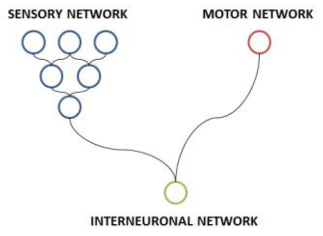

Figure 2. The NNM architecture

A detailed description of how the general NNM architecture applied in the pong game is presented in Section 3.

3 Pong game and NNM controller model

This section describes the design of the pong game, and more essentially the design of the NNM controller.

3.1 Pong game design

A pong game environment is simulated with the 2D robotics simulation platform, Stage [19], [20], as shown in Fig. 3. The blue rectangular object is the simulated paddle, and the red spherical object is the pong ball. The green area is the playable area, which is the table for the game. The paddle is only allowed to move horizontally, while the ball is allowed to move anywhere within the table area, in a linear fashion.

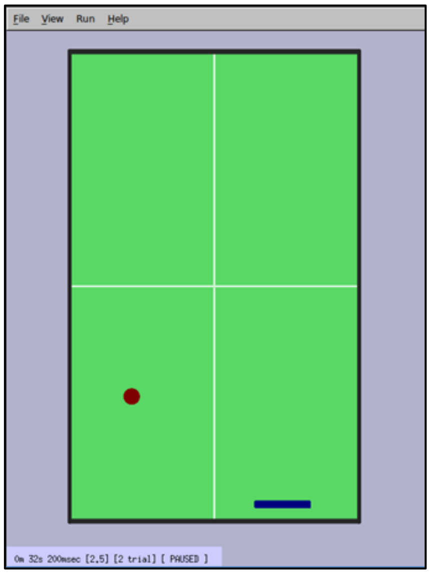

Figure 3. Simulated pong game in Stage

In this application, the blue paddle is controlled by the NNM. Its opponent is a preset computer program, which returns the ball in random directions from the other end of the table. In this simulation, the ball can start from any of 15 given positions at the top of the table. The ball can move at any angle, between 21 to 159 degrees. The NNM will learn to position the paddle at the correct position at the bottom of the table in order to return the ball successfully. The paddle can be placed at 14 possible horizontal positions. The discrepancy between the ranges of the paddle (14 discrete values) and pong (15 discrete values) positions is due to the size and shape of the objects. These attributes determine the ranges of the objects. The ranges should confine the movement of the objects within the table area.

The deflection of the ball at the sides of the table is simulated as a perfect-reflection. However, the deflection of the ball at the paddle and at the top of the table need not be of perfect-reflection, but can be anywhere between 21 to 159 degrees.

3.2 Pong game NNM architecture

The structure of the NNM in this pong game is shown in Fig. 4. It is a 4-network-NNM architecture (denoted by 4 different colours), which is based on the general 3-network-NNM architecture in Fig. 2. The sensory events of the pong game are the ball positions on the pong table, in terms of chessboard-based coordinate system (*x* and *y* axis, as column and row respectively). These events are captured in a network known as a sensory network. On the other hand, the positions of the NNM controlled paddle are the motor events in the model. These paddle positions are stored in another network, known as motor network. A sensory event and a motor event are associated through a controller event in another network defined as an interneuronal network.

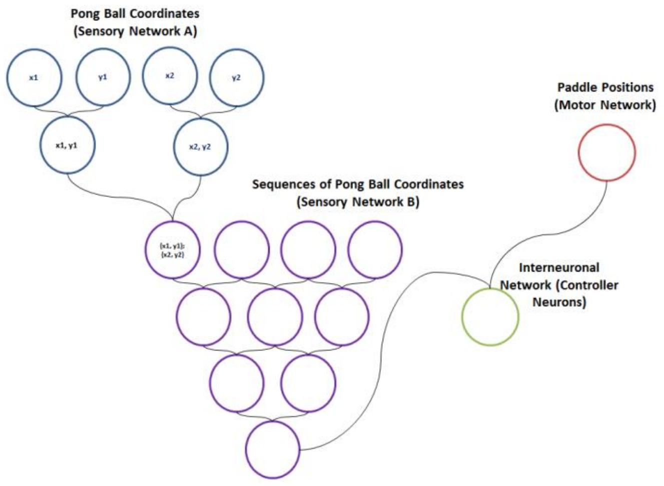
Pong Ball Coordinates (Sensory Network A)
x1
y1
x2
y2
x1, y1
x2, y2
Sequences of Pong Ball Coordinates (Sensory Network B)
(x1, y1); (x2, y2)
Paddle Positions (Motor Network)
Interneuronal Network (Controller Neurons)

Figure 4. The NNM architecture of pong controller

Fig. 5 shows the coordinate system of the pong table. The range of *x* axis is of lower case alphabets: *{a, b, c, …, n, o, p}*, while the *y* axis is of upper case alphabets: *{A, B, C, …, W, X, Y}*. So, the positions of the ball and paddle in the figure are *(b, B)* and *(h, Y)* respectively.

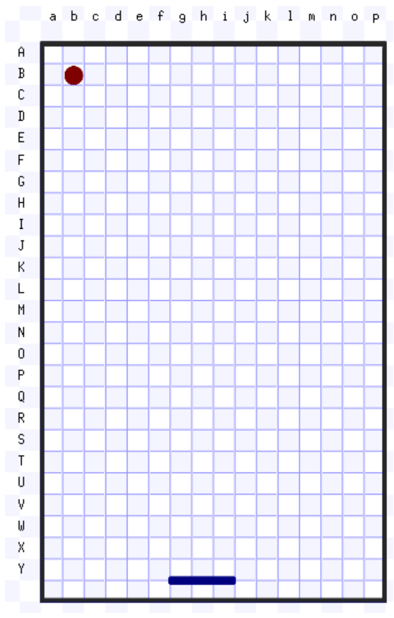
a b c d e f g h i j k l m n o p
A B C D E F G H I J K L M N O P Q R S T U V W X Y

Figure 5. Grid map of simulated pong game

3.2.1 Sensory network

The coordinates of the pong ball are sampled at 5 pre-defined positions on the *y* axis. The five *y* positions are *{D, F, I, N, V}*, wherein each position comes after another position in a sequence heading towards the NNM-controlled paddle's end. When the ball has reached the defined *y* positions, the coordinate is taken and stored in the NNM.

The first two coordinates acquired by the sensor provide the information on the initial trajectory of the ball. The next three coordinates refine the trajectory of the initial information, as the ball trajectory will be changed if it hits the sides of the table.

The sensory events are defined by two stacked sensory networks, such as shown in Fig. 4. One of the sensory networks, Sensory Network A builds the coordinates from a set of pre-defined elementary sensory neurons. The elementary sensory neurons contain the basic representations to define the coordinates. Based on the possible squares on the *x* and *y* axes, there are only 21 elementary sensory neurons (16 *x* squares + 5 *y* squares).

Sensory Network A captures the coordinates at interval *y* axis squares that the ball has "glided over" when the game is running. Therefore, there are only 2 levels in this network. The first level contains the elementary neurons, and the second level stores the coordinates. This network will serve as the basis to build the sequences of ball positions in the other sensory network, Sensory Network B.

Sensory Network B has only 4 levels of neurons. This is because there are only 5 coordinates from the start of the ball at the opponent's end to the NNM's end. Two coordinates are required to form a sequence in the first level neuron of this network. Hence, only 4 levels of neurons exist.

3.2.2 Motor network

In the motor network, there are 14 basic motor events: *{b, c, d, ..., m, n, o}*. Hence, there are only 14 elementary motor neurons. The motor event, which is also known as motor action, is the position of the paddle. The position corresponds to the *x* axis squares where the paddle can be placed, while being fixed at the bottom of $y = Y$ (*y* axis square). The range is set between *b* and *o* because the length of the paddle is of 3 units, and the need of keeping the paddle within the table.

The motor actions are stored in a single level in the network. Each motor action is associated with at least a controller event.

3.2.3 Interneuronal network

The relationship between a motor action and a sensory event is defined by a controller event in the controller interneuronal network, through a controller neuron. A sensory event can have multiple controller events, wherein each controller event is associated with a different possible motor action.

The strength of the relationship between a sensory event sequence and a motor action is weighted in terms of being the correct action. By using the frequency attribute to define this, correct motor actions can be predicted for a sensory event.

4 Learning algorithm

The overall learning model of the NNM controller is depicted in Fig. 6. The learning starts when the pong ball has been hit by the opponent, and on its way towards the NNM's end of the table. The coordinate of the ball is acquired sequentially at a series of *y* axis squares.

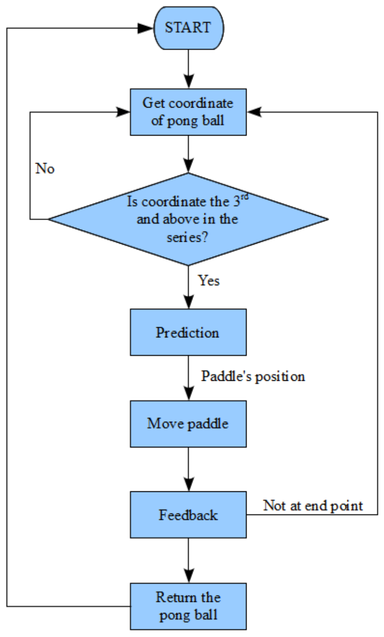

Figure 6. The overall learning process flow

4.1 Prediction

The prediction starts when the third coordinate in the *y* axis series is acquired. The prediction does not begin from the first two coordinates because at least two coordinates are required to gauge the direction of the pong ball. Therefore, it is reasonable to begin the prediction at the third coordinate. Furthermore, at the third coordinate, the ball is still within the opponent's area (about a quarter way of the pong table).

The NNM controller will attempt to return the pong ball based on the sequence of sensory events received. A matching sensory event is searched within its network. If a match is not found, it means that the event is a new sensory event. The new event will be encoded into the NeuraBase sensory networks; and a random motor action to position the paddle is executed.

On the other hand, if a match is found, a number of potential motor actions will be given through the related controller neurons. The controller neuron has to meet a certain prediction threshold before the corresponding motor action is considered for execution. In this model, the prediction threshold, which is the frequency of the controller neuron, has to be equal or greater than five *(≥ 5)*.

The controller neurons which have met the threshold will be processed with the weighted average method to predict the most sensible motor action. If none of the controller neurons related to the sensory event has met the threshold condition, a random motor action will be executed.

4.2 Feedback

Once the recommended motor action is given, the paddle is moved accordingly to position. When the paddle is in position, feedback is provided to the controller for learning. Fig. 7 shows the feedback learning process of the model. If the pong ball has not reached the paddle *y* axis position, which is also the end point, learning is not required on the feedback since the pong ball is still moving forward.

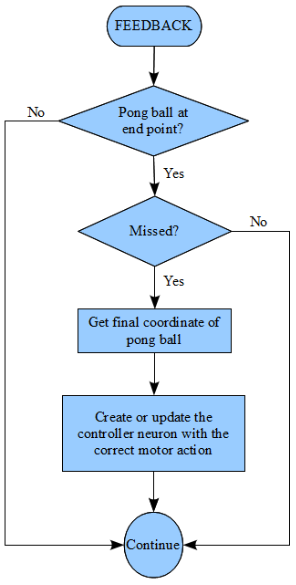
FEEDBACK
Pong ball at end point?
No
Yes
Missed?
No
Yes
Get final coordinate of pong ball
Create or update the controller neuron with the correct motor action
Continue

Figure 7. The feedback process flow

When the pong ball has reached the end point, the feedback is analysed. In this model, the NNM is learning from its failures to predict correctly. If the ball is returned successfully, nothing is updated in the NeuraBase and the game continues. However, if the paddle misses the ball, the final position of the ball is taken into consideration for learning. The final position is treated as the expected position of the paddle if given the same sensory sequence in future, subject to meeting threshold requirements.

NeuraBase registers this knowledge by creating or updating a controller neuron. If the controller neuron has already been registered, the corresponding controller neuron is updated instead. With this, knowledge is reinforced. A reward is given to the associated controller neuron through the increment of the frequency attribute. In this case, the reward is 5.

When learning is based on failed predictions, the previous failures in prediction are corrected by creating or updating the correct controller neuron. In this case, the frequency of the erroneous controller neurons is not penalised. Instead, the correct controller neuron's frequency is increased.

As such, the problem of overfitting is minimised or avoided altogether. Overfitting may occur in the pong game when the high number of successful returns may be misconstrued as good prediction, even though it is actually not.

5 Experimental results

The pong game was started with only the elementary sensory and motor neurons defined in NeuraBase, and run for 10000 trials. Each trial begins with the ball at a random position from the opponent's end, with a random angle as the trajectory. The trial ends when the NNM-controlled paddle misses the pong ball. Fig. 8 shows the number of times the ball was returned by the NeuraBase-controlled paddle in each trial.

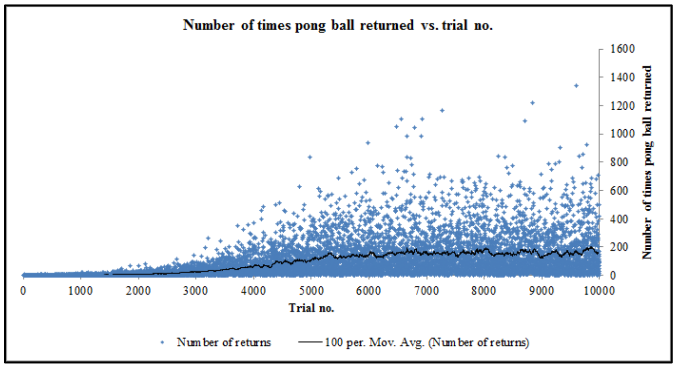

Figure 8. Frequency of pong ball returns per trial

A 100-trial moving average trend line of the frequency of ball returns per trial is shown in Fig. 9. As observed in the figure, the learning process is on an uptrend, and begins to show saturation at around 6000 trials, between 125 and 200 returns.

Fig. 10 shows the total number of neurons used in NeuraBase after 10000 trials. Altogether, 30341 neurons were created in the NeuraBase. This requires approximately 1.3 Mbytes of physical memory as each neuron uses 40 bytes of memory. Out of these neurons, 16401 neurons are built in the extended sensory network, 14 neurons in the motor network and 13926 neurons in the controller interneuronal network.

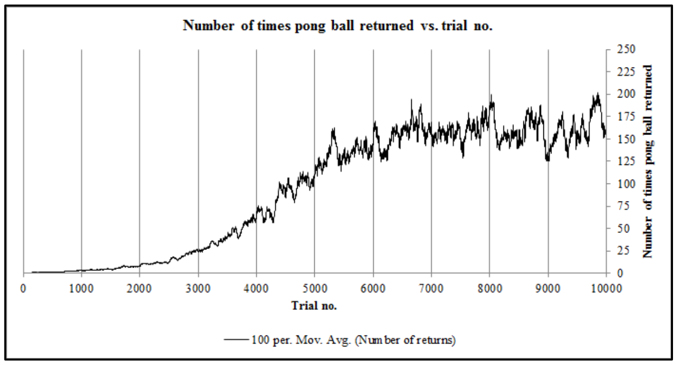

Figure 9. Moving average of frequency of pong ball returns per trial

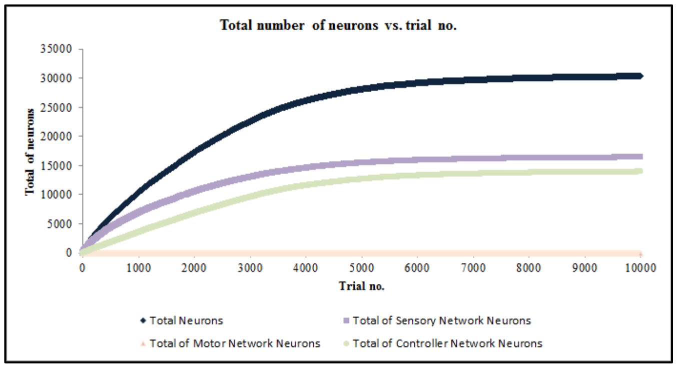

Figure 10. Neuron statistics in NeuraBase

Initially, the number of neurons in the sensory and controller networks is growing in a steep slope because of the high occurrence of new sensory events or experience. This growth eventually approaches saturation as the number of new sensory events decreases. The number of controller network neurons is lower than sensory network neurons all the time; as the links between sensory network neurons and motor network neurons are only created by learning the final pong ball position when it is missed. Therefore, only the correct motor actions are associated with sensory neurons. However, each sensory neuron can possibly have a few correct motor actions due to the variation of pong ball trajectories in a square.

Another experiment was the study of the failure of prediction from NeuraBase. In the failure study, a hundred trials were run with the NeuraBase that has been trained earlier with 10000 trials. As each trial ends with a miss, hence, there are 100 failures. Suppose that an instance is a series of predictions from NeuraBase to prevent an opponent's attempt to score. Overall, 15584 such instances of prediction were performed. So, the failures occurred at 0.6417% (99.3583% successes) of the time. Out of the 100 trials, 45 trials failed because the predictions were still random, indicating that there is still room for NeuraBase to learn. Therefore, only 0.3529% of instances are actually failures from NeuraBase's predictions.

One reason found for the failures in NeuraBase's predictions is the approximation of the ball position to a square on the pong table. The ball can be in any position in a square. As such, a square may consist of a number of different paths, even if the trajectory angles may be the same. This may result in a miss because the predicted motor action may only tolerate a certain variance in the path. Even if another motor action is considered, it may fail again as the variance may have just shifted.

Table I shows how the path can be different for the same sensory event, *(b, V)* square. In the *x* axis, any value between *-6* and *-7* is considered as *b*. As for the *y* axis, any value between *-8* and *-9* is mapped to *V*. As depicted in the table, there are multiple ball coordinates corresponding to the same square. The predicted motor action, which is *d*, is positive for most coordinates, except for an anomaly in the first row (negative feedback).

TABLE I
PATH OF PONG BALL AND SENSORY DATA REPRESENTATION

x Axis	Sensory X	y Axis	Sensory Y	Trajectory Angle	Feedback
-6.09	b	-8.15	V	-45	NEG-
-6.62	b	-8.31	V	-129	POS+
-6.66	b	-8.31	V	-129	POS+
-6.64	b	-8.29	V	-129	POS+
-6.22	b	-8.18	V	-45	POS+
-6.16	b	-8.21	V	-45	POS+
-6.18	b	-8.23	V	-45	POS+

In addition, failure may also be caused by the fact that the paddle is sometimes slow in moving towards the predicted position. The pong ball may have a shorter distance towards the end of the table; and the paddle may coincidently be further away from the point that it is supposed to be. So, the ball might reach the end before the paddle is in its final position. However, the chances of this happening are small, and are based on the Stage simulation model specified for velocity control of the paddle.

The causes of failures described above may be solved by having more sensory information, for example by having extra angles allowed to be travelled by the ball. Or, an extra coordinate in addition to the five already in the series. Speed may be also factored into the NeuraBase design so that the paddle can move with variable speeds.

6 Conclusions

The research described in this paper has demonstrated that the Neuronal Network Model performs well as a controller in a game environment. The high percentage of successful predictions is a testimony to this. The experimental results also show that the total amount of required memory is small.

As future work, more parameters such as velocity and acceleration can be incorporated into the controller of the pong ball. Besides that, a 3-dimensional model of the pong game could be developed to examine how the model performs in a more complex environment. Two independent NeuraBases could be created for a two-player game as well.

References

[1] Entertainment Software Association. (2013). Essential facts about the computer and video game industry [Online]. Available: http://www.theesa.com/facts/pdfs/esa_ef_2013.pdf

[2] Wikipedia. (2014). Far Cry 3 [Online]. Available: http://en.wikipedia.org/wiki/Far_Cry_3

[3] Microsoft Research. (2014). Drivatar in Forza Motorsport [Online]. Available: http://research.microsoft.com/en-us/projects/drivatar/forza.aspx

[4] Olivier Delalleau, Emile Contal, Eric Thibodeau-Laufer, Raul Chandias Ferrari, Yoshua Bengio, and Frank Zhang. "Beyond skill rating: advanced matchmaking in Ghost Recon Online" ; IEEE Trans. Computational Intelligence and AI in Games, vol. 4, no. 3, pp.167-177, Sept 2012.

[5] Micrsoft Research. (2014). TrueSkill Ranking System [Online]. Available: http://research.microsoft.com/en-us/projects/trueskill

[6] Renato Luiz de Freitas Cunha and Luiz Chaimowicz. "An artificial intelligence system to help the player of real-time strategy games," in 2010 Brazilian Symp. Games and Digital Entertainment (SBGAMES), pp. 71-81.

[7] Sindre Berg Stene. "Artificial intelligence techniques in real-time strategy games – architecture and combat bahavior," M. S. thesis, Dept. Comput. and Inform. Sci., Norwegian Univ. of Sci. and Technology, Norway, 2006.

[8] Wikipedia. (2014). Unreal Tournament [Online]. Available: http://en.wikipedia.org/wiki/Unreal_Tournament

[9] 2014 StarCraft AI Competition [Online]. Available: http://webdocs.cs.ualberta.ca/~cdavid/starcraftaicomp/

[10] Santiago Ontañón, Gabriel Synnaeve, Alberto Uriarte, Florian Richoux, David Churchill, and Mike Preuss. "A survey of real-time strategy game AI research and competition in StarCraft" ; IEEE Trans. Computational Intelligence and AI in Games, vol. 5, no. 4, pp. 293-311, Dec 2013.

[11] David Churchill. (2013). 2013 AIIDE StarCraft AI Competition Report [Online]. Available: http://webdocs.cs.ualberta.ca/~cdavid/starcraftaicomp/report2013.shtml

[12] Vishnu Kotrajaras and Tanawat Kumnoonsate. "Fine-tuning parameters for emergent environments in games using artificial intelligence" ; Int. J. of Comput. Games Technology, Jan 2009.

[13] Gustavo Recio, Emilio Martin, César Estébanez, and Yago Sáez. "AntBot: Ant colonies for video games" ; IEEE Trans. Computational Intelligence and AI in Games, vol. 4, no. 4, pp. 295-308, Dec 2012.

[14] NeuraBase Generic Toolbox [Online]. Available: http://www.neuramatix.com

[15] Robert George Hercus. "Neural networks with learning and expression capability". U.S. Patent 7412426 B2, 2008.

[16] Robert Hercus, Kit-Yee Wong, and Kim-Fong Ho. "Balancing of a simulated inverted pendulum using the NeuraBase network model" ; Lecture Notes in Comput. Sci., vol. 8131, pp. 527-536, 2013.

[17] Robert Hercus, Hong-Shim Kong, and Kim-Fong Ho. "Control of an unmanned aerial vehicle using a neuronal network," in 2013 IEEE Symp. Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), Singapore, 2013, pp. 73-79.

[18] Robert Hercus, Kit-Yee Wong, and Kim-Fong Ho. "Control of a muscle actuated manipulator using the NeuraBase Network Model" ; J.Automation and Control Eng, vol. 2, no. 3, pp. 302-309, 2014.

[19] The Player Project [Online]. Available: http://playerstage.sourceforge.net/

[20] Brian Gerkey, Richard T. Vaughan, and Andrew Howard. "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems", in Proc. 11th Int. Conf. Advanced Robotics, Coimbra, Portugal, June 2003, pp. 317-323.