Int. J. Advance. Soft Comput. Appl., Vol. 1, No. 1, July 2009
ISSN 2074-8523; Copyright © ICSRS Publication, 2009
www.i-csrs.org

Modular Weightless Neural Network Architecture for Intelligent Navigation

Siti Nurmaini, Siti Zaiton Mohd Hashim, Dayang Norhayati Abang Jawawi

Faculty of Computer Science
University of Sriwijaya
Indonesia
e-mail: siti_nurmaini@.unsri.ac.id
Faculty of Computer Science and Information System
Universiti Teknologi Malaysia
Skudai, Johor Bahru
e-mail: sitizaiton@utm.my, dayang@utm.my

Abstract

The standard multi layer perceptron neural network (MLPNN) type has various drawbacks, one of which is training requires repeated presentation of training data, which often results in very long learning time. An alternative type of network, almost unique, is the Weightless Neural Network (WNNs) this is also called n-tuple networks or RAM based networks. In contrast to the weighted neural models, there are several one-shot learning algorithms for WNNs where training takes only one epoch. This paper describes WNNs for recognizes and classifies the environment in mobile robot using a simple microprocessor system. We use a look-up table to minimize the execution time, and that output stored into the robot RAM memory and becomes the current controller that drives the robot. This functionality is demonstrated on a mobile robot using a simple, 8 bit microcontroller with 512 bytes of RAM. The WNNs approach is code efficient only 500 bytes of source code, works well, and the robot was able to successfully recognize the obstacle in real time.

Keywords*: Weightless neural network, environmental recognition, microprocessor system, embedded application.*

1 Introduction

The neuron model used in the great majority of work involving neural networks is related to variations of the McCulloch-Pitts (1943) neuron, which will be called the *weighted-sum-and-threshold neuron,* or *weighted neuron* for short. The implementation of the network operation requires the use of multiplier unit, which can be problematic for embedded application due to the large size of resource because the training the generalized data required both the forward propagation phase and backward (back error) propagation phase. A typical weighted neuron specifies a linear weighted sum of the inputs, followed by some non-linear transfer function (I. Aleksander & H. Morton, 1995; T. B. Ludermir et al. 1999). The complexity of the training algorithm has limited its implementation in dedicated hardware. In contrast, the ANNs investigated are based on artificial neurons which often have binary inputs and outputs, and no adjustable weights between nodes. Neuron functions are stored in look-up tables that can be implemented using commercially available Random Access Memories (RAMs). Learning on these systems generally consists of changing the contents of look-up table entries, which results in highly flexible and fast learning algorithms. These systems and the nodes that they are composed of will be described respectively as *Weightless Neural Networks* (WNNs) and *weightless nodes* (I. Aleksander & H. Morton, 1995; T. B. Ludermir et al. 1999). They differ from other models, such as the weighted neural networks, whose training is accomplished by means of adjustments of weights. In the literature, the terms, “RAM-based” and “N-tuple based” have been used to refer to WNNs (R. Rohwer and M. Morciniec 1996; T. M. Jorgensen 1997; R. Al-Alawi 2007; P. Coraggio and M. De Gregorio 2007; C. Linneberg and T. M. Jrgensen 2006). In that approach supervised (Austin & Stonham, 1987; Filho et al. 1991; Gorse & Taylor 1991; Howells et al. 1995; Jorgersen et al. 1995) and unsupervised (Allison & Johnson 1989; Clarkson et al. 1991; de Carvalho et al. 1992; Ntourntoufis et al. 1990) learning techniques have been used with WNNs.

WNNs have received extensive research attention and regarded as powerful learning machines, in particular, as excellent pattern classifier (L. Teresa et al. 1999; Ludermir et al. 1999). WNNs possess many prominent features such as their simple one-shot learning scheme (Simoes, 1996), fast execution time and readiness to hardware implementation (Simoes 1996; Austin 1994; I. Alexander 1995) also well suited to microcontroller-based-real-time systems. In contrast to the weighted neural models, there are several one-shot learning algorithms for WNNs where training takes only one epoch. Although the basic WNNs approach was powerful, in terms of is learning speed and simple implementation. This method has a major limitation. This relates to the learning capacity of a given network. By inspection it may be obvious that network cannot implement all possible function of the data inputs, this is called the generalization problem [Austin, 1999]. To increase the robustness of the method the functional capacity needed to be raised whilst maintaining the generalization ability, along with the training speed and simple hardware implementation.

This work concerns development of efficient WNNs for classifies environment and recognize an obstacles in a mobile robot based on a simple microprocessor system. Some work done by others indicates this approach may be successful and effectiveness using WNNs in a number experimental studies in mobile robot (Mitchell et al. 1994; Silvia et al. 1996; Zhou et al. 1999; B. Osterloh, 2003; Yao et al. 2003; Zhang et al. 2005; Paolo C and M.D. Gregorio 2007; Simoes, E.D.V, 2008). They studies base on software implementation. However, that approach is limited by the amount of variable memory since RAM neurons are assembled as binary vector, addressed by the inputs and stored into system memory (L. Teresa 1999). If the number of inputs is high, what is very common in complex problems, each neuron will need a great amount of memory. This fact can make intractable the application of the RAM model in some cases with less available memory like microcontrollers. But the advantage of WNNs is their modularity. Modularity can be viewed as a manifestation of the principle of divide and conquer, which allows us to solve complex problems, by dividing them into smaller sub problems, easier to conquer, combining their individual solutions to achieve the final solution (Haussler, 1989). One of greater improvement achieved by this architecture is better generalization; the convention in neural networks is to use architecture as small as possible to obtain better generalization (A. Schmidt 1996).

One more important problem to solve in mobile robot research is self localization. It is often addressed as the first property to endow a robotic system to have high level reasoning capacity (Cox, I. J. 1991). The global position problem is the ability, for the robot, to autonomously recognize its pose starting from an unknown position on a given map (Borenstein, J. et al. 1996). It is a very difficult problem needing a lot of computational power, and giving not so much accurate results in terms of robot pose estimation (Thrun, S. 2005).

To overcome the above problem, in this paper we proposed combination of software and hardware implementation in modular architecture to enhance generalization and for solving a robot problem in estimation of the robot position This characteristic simplifies the modification the architecture of the neuron. The number of neuron inputs can be modified by rearranging the connectivity to the sensors. This work demonstrates the potential of proposed architecture in embedded applications on mobile robot where powerful computers may not be available and mobile robot capable of detecting and classifying the environment. In the paragraphs that follow, the robot's architecture, sensor system and the WNNs structure will be presented. Results of experiments that measure the robot's recognition environment ability will also be given.

2 System Overview

This section describes the theoretical basis of the paper as well as the problem definition. Some basics about WNN systems are first presented.

2.1 The RAM Node

The basic architecture of RAM model is as follows, the input vector is divided into parts; each part is connected to the address inputs of a 1 bit-RAM unit. In a typical RAM network, that is single-layer architecture. This depicted in Figure 2 is a device which can store one bit of information for each input address. A control input is available to switch the mode of the RAM between 'Write' and 'Read' for learning and recall. Initially all memory units are set to '0'. During the learn (Write) mode the memory is set to '1' for each supplied address; in the recall (Read) mode the output is returned for each supplied address, either '1' (if the pattern was learned) or '0' (if the pattern was not learned). RAMs are taught to respond with a 1 for those patterns in the training set and only for those patterns. An unseen pattern is classified in the same class of the training set if all RAMs output 1.

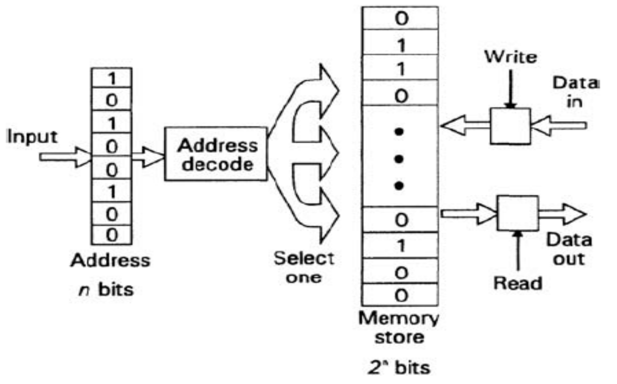

Fig. 2 RAM as Boolean function

In contrast to biologically motivated nodes, RAM nodes were initially designed as engineering tools to solve pattern recognition problems. An N input RAM node has 2^N memory location, addressed by the N-bit vector a = {a_1, a_2, ….,a_N}. A binary signal I = {I_1, I_2, ….,I_N} on the input lines will access only one of these locations, that is the one for which a = I. The bit C [I], stored at this activated memory represent the output of the node, that is r = C[I].

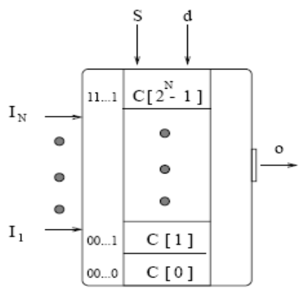

Fig. 1 Typical RAM node

Where,

N = connectivity; I_j = input terminal

C [I] = memory content; o = output terminal to transmit

d = input terminal to receive; s = learning strategy

2.2 The Discriminator

The basic element of discriminator is a bit addressable RAM discriminator that is formed by a layer of K n-input RAMs, each one storing 2^N one bit words. Although the input mapping is chosen at random, such a mapping is often fixed parameter of the network [L. Teresa et al. 1999; C. Linneberg and T. M. Jrgensen 2006]. This response vector can be regarded as a feature vector that measures the similarity of an input pattern to all classes.

In order to solve the problem of RAM-nodes having no generalization, several of these RAM-nodes can be combined to a so called discriminator. It consists of a layer of RAM-nodes organized as illustrated in Figure 2. The most important feature of a discriminator is that it splits the input pattern into K n-sized tuples, each of which are used as the input for a RAM-node. In this way the input pattern of the discriminator is distributed over the memory of several RAM-nodes, instead of being stored in only one location. The generalization is obtained from introducing a summing device, denoted by the $\sum$ symbol. The function of this device is to sum up the outputs of all the RAM-nodes of that discriminator and divide it by the number of RAM-nodes K. This gives the fraction of RAM-nodes that generated a "1" when input vector X was given to the discriminator:

$$r_j = \frac{1}{K} \sum_{i=1}^{K} f_{ji}(X) \tag{1}$$

Where

n= tuple size; K = number of RAM-nodes ; X = input vector

f_{ji} = output of RAM-node i belonging to the j^{th} discriminator

x_i = the i^{th} element of X; r_j = the output of the j^{th} discriminator

In this way the result *r* of the discriminator can be seen as the outcome of a membership function, it is a number between 0 and 1 which states how much pattern X resembles the pattern used in training set for discriminator. The set of feature vectors generated by the WNNs for a given training set will be used as input data to the fuzzy rule-based system. The normalized response of class j discriminator x_j is defined by:

$$x_j = \frac{r_j}{K} \tag{2}$$

The value of x_j can be used as a measure of the similarity of the input pattern to the j_{th} class training patterns. The normalized response vector generated by all discriminators for an input pattern is given by:

$$\mathbf{x} = [\, x_1, x_2, \ldots\ldots, x_N \,] \,, \quad 0 \le x_j \le 1 \tag{3}$$

This response vector can be regarded as a feature vector that measures the similarity of an input pattern to all classes. During the recalling, the system presents the input vector to all trained discriminator. The results r_j of the discriminators j_{th} are passed to a calculation unit. This unit determines which discriminator gives the highest output. This is the class to which the input pattern is classified. It also calculates a measure of relative confidence C:

$$C = \frac{r_{highest} - r_{\sec ondhighest}}{r_{highest}} \tag{4}$$

In the WNNs, Winner-Takes-All decisions can be attached to the adder outputs to choose the discriminator containing the greater number of active neurons and pointing to the winning class. Each pattern will produce a feature vector that describes its similarity to all classes.

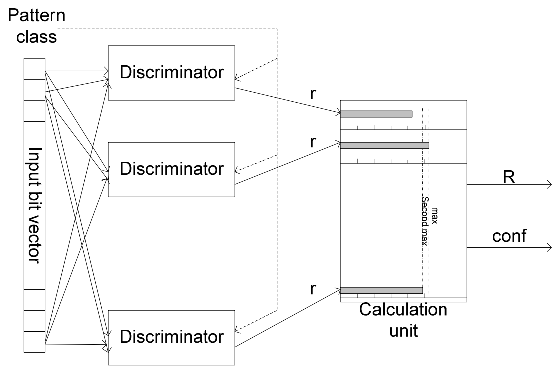

Fig. 2. Discriminator

3 The Robot Platform

The system of the robot in Figure 4 is a modular system and containing several microcontrollers implies the implementation of a robust communication mechanism between modules. For this platform, the architecture can conceptually be seen as the central module, the motor driver module and the sensor module are connected to each other via 8 bit data bus. Each microcontroller can be equipped with a central control module as the motion unit controller. We use microcontroller AT89x55 for central controller, attached with ROM 20 Kbytes, RAM 256 bytes and clock 24.3 MHz, operated in 0.5 micro-second for each process. A general purpose high performance controller for decision making is connected to the parallel bus. There are eight ultrasonic range finder sensors located at the front, left and right side of the robot. In sensor module, eight PIC16F84 chips are used for process signal detection from ultrasonic sensors, with ROM 1 Kbytes. Attached to the mobile robot, each separated at 30^o along the circumference. Furthermore, for real time control applications, PIC18F2550 is used exclusively to generate the Pulse Width Modulation (PWM) signals and to run the two dc motors, attached to each motor is an optical encoder which is used for distance and velocity calculation. Figure 4, illustrates the architecture that is available on the respective modules.

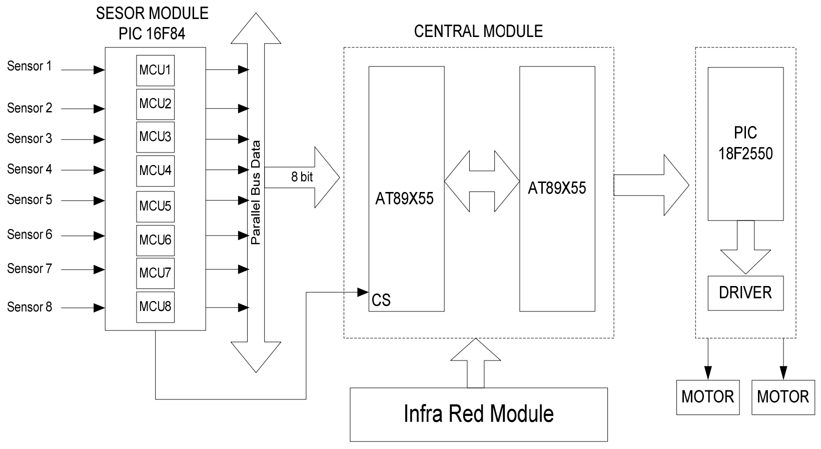

Fig. 4. Architecture of modular system

The robot behaviors tested by eight sensors, and divided into four behaviors namely obstacle avoidance, left and right wall following, and emergency condition. The sensors position in all array are fixed: one on back sides, and seven facing outwards at 30^o intervals starting from 1 -7. They report the distance between the robot and the closest obstacle. The placement of the sensors permits the detection of an obstacle in different positions. That configuration was very efficient during preliminary tests of robot navigation. Seven sensors positioned in the front of the robot, considering that the robot will be moving when it is maneuvering, there is no situation where a robot can approach an obstacle without seeing it. One sensor is placed in the back to allow the robot to detect emergency condition and the robot must be move backward or stop.

4 Modular WNNs Classifier Design

4.1 Sensor Classification

In order to generate the training set, it is necessary to create for each class of discriminator. We considered the common target that there exist in real environment of mobile robot applications such as plane, edge, corner with angle 90 degree, acute corner with angle 60 degree. Length of plane is about 45 cm and other objects are of similar size. These objects at four distances: 10, 20, 30, 40 cm. Also angle between the head of mobile robot and these objects is assumed to be -30, -20, -10, 0, 10, 20, 30 degree. Figure 5 show the sensor classification at these nine classes environment. Here we use patterns with a single far, single medium, or single near obstacle to train the neural network. The threshold values for training are, 00100010 (30 cm) that mean far, 0001 0111 (20 cm) that mean medium, 00101111 (10 cm) that mean near and 0100 1011(75 cm) indicating that no obstacle is detected.

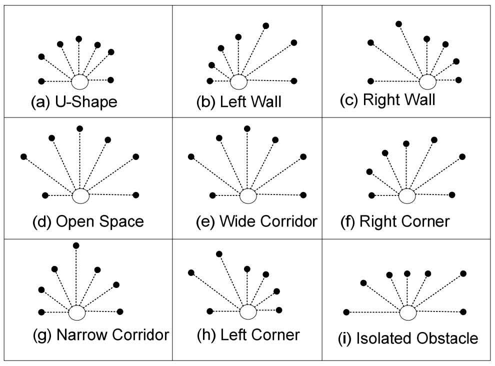

Fig. 5. Sensor classification

4.2 Structure of Modular WNNs

The WNNs is designed to identify the current environment by recognizing typical patterns. To implement this network, the 8 bit data from eight ultrasonic sensors is used to determine the direction of the obstacle, in each direction an obstacle may or may not appear. The sensor module can enable or disable this 8-bit signal, preventing the sensor to be connected to the input lines. The combination of them appearance in the seven directions makes up different input pattern. In this experiment, a two- layer RAM base neural network was used, there are sensor layer and output layer, the structure is shown in Figure 6. The neural network used eight neurons for identifies where obstacle may lie in seven direction through ultrasonic sensor and seven output commands means that the neural network needs eight bits to encode them. Its configuration has seven groups of eight neurons (m=8 and n=7), so the network has 56 neurons.

The neurons s_1 to s_n are the binary sensor readings are connected in groups (discriminators) that correspond to one of the possible classes of commands ($c_1, c_2, \ldots, c_n$) the neural network can choose. The groups are connected to an output adder ($o_1, o_2, \ldots, o_n$) that counts the number of active neurons in the group, there are seven position of obstacle in the robot environment such as, front, right, and left side of the mobile robot. A winner take all decision can be attached to the adder outputs to choose the discriminator containing the greater number of active neurons, pointing to the winning class.

The RAM neuron shown in Figure 6 has four behavior classes namely, obstacle avoidance, right wall follow, left wall follow and goal seeking behavior with eight neuron each. The RAM neural network simplicity and its implementation as elementary logic functions are responsible for its fast performance. Basically, the neural network must have enough inputs to cover all the sensors, although some of the sensors may be connected to more than one input line. To avoid saturation, enough neurons must be placed in the groups so that the network can learn all the different input configurations, which correspond to the correct output commands

[Filho, 1992]. If the network is having difficulty learning a different configuration, more neurons should be added. Different architecture was implemented until the developed solution was obtained.

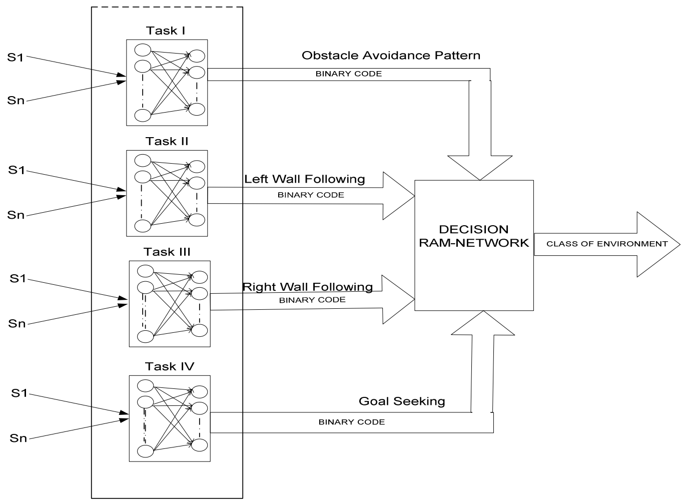

Fig.6. Structure of WNNs

4.3 Generating Learning Strategy

In contrast to the weighted neural models, there are several one shot learning algorithms for WNNs, where training takes only one epoch. The learning of WNNs has no weight matrix, it directly changing the neuron contents in the look-up tables [5]. The procedures of learning/recalling process takes places by writing/reading into corresponding look up table entries. Initially all the contents are set to 0 and the learning of an input pattern is through the writing of the value 1 in the address content of the RAM neuron. During the recalling phase an input pattern is clamped to the network and all the neurons produce an output. The algorithm used to train is as follow.

- Initialize all location of the memory to a random binary [1,0] values and created the discriminator
- Defined recognition interval by the parameter $0 \leq r_{\min} \leq r_{\max} \leq 1$, $r_{\max}$ being the maximum recognition
- Select an input pattern of the environment from sensor array
- Access the RAM and generate an output,
- If the value at the output of the network is occurred and correct, r set to 1,
- If the value at the output of the network is occurred and incorrect, r set to 0

- Check value of recognition interval r, if $r_h \le r_{\min}$, a new discriminator is created
- Check value of recognition interval r, if $r_{\min} \le r_h \le r_{\max}$, it is assumed that the pattern to be performed is probabilistic selected
- Check value of recognition interval r, if $r_h \le r_{\max}$, it is assumed that the pattern is already well represented the winning class of discriminator
- For all nodes, if r =1 then set input pattern to be learned, if r = 0, then clear all the nodes and reenter input pattern.
- Go to access the RAM

The algorithm requires repeated application of this process, until all nodes have learned the patterns.

5. Experimental Result

5.1 Training with threshold

The weightless neural networks, mainly the RAM-network model, tent to occupy a great portion of memory. Therefore, memory allocation can be a significant problem in some hardware applications, limiting the control system choice. The WNNs mapping into logic function and their direct execution in the microcontroller ALU have also allowed a reduction of the memory required by the control algorithm. A group of test was performed according to the processing time, number of collisions and required memory. The WNNs great simplicity ad their implementation as elementary logic functions is responsible for their greater performance. Experiment is conducted to demonstrate the ability of a mobile robot to recognize to various unknown environment, particularly, the ability of a robot to escape from the trapping situation, this depicted in Figure 8. The base address 30H in hexadecimal is the address in memory of the first byte of the 256 locations of the memory. The result is based on the environment classification can show in Table 1 and Table 2. Using 10 experiment data, winner take all learning algorithm has achieved 95 % recognition for obstacle and 94 % classification. How ever the poorest result was if the robot closes the object, where the scanning sensory sector of the robot was quite high and some noise has still interfered in echo signal.

Table 1: Environmental Recognition

Actual Place	Distance (cm)	Refe-Rence (hex)	Result (hex)
Convex (90^{o})	20	12h	12h
	30	0bh	0bh
	40	05h	0dh
Concave (270^{o})	10	06h	00h
	20	0eh	00h
	30	00h	00h
Plane (180^{o})	10	03h	07h
	20	0ch	0ch
	30	15h	15h
	40	1eh	1eh
Left-corner (180^{o})	10	03h	07h
	20	0ch	0ch
	30	15h	15h
	40	1eh	1eh
Right-corner (180^{o})	10	03h	07h
	20	0ch	0ch
	30	15h	15h
	40	1eh	1eh
Corridor (0^{o})	10	03h	07h
	20	0ch	0ch
	30	15h	15h
	40	1eh	1eh
U-shape (180^{o})	10	03h	03h
	20	0ch	0ch
	30	15h	15h
	40	1eh	1eh

Table 2: Obstacle Recognition

Target angle	Obstacle angle	Obstacle distance (cm)	Plane	Rectangular object	Circular object
30	15	10	0eh=14d	0eh=14d	0eh=14d
		20	12h=18d	12h=18d	12h=18d
		30	17h=23d	17h=23d	17h=23d
		40	1dh=29d	1dh=29d	1dh=29d
40	20	10	0ch=12d	0bh=11d	0ch=12d
		20	16h=22d	17h=23d	16h=22d
		30	20h=32d	22h=34d	20h=32d
		40	2dh=45d	2eh=46d	2dh=45d
50	25	10	0	0	0
		20	0	0	0
		30	1eh=30d	0	0
		40	2ah=42d	0	0
60	30	10	0	0	0
		20	0	0	0
		30	0	0	0
		40	0	0	0

5.2 Generalization

During the training phase the memories of the RAM-nodes are filled with ones. The growth if the number of ones in the memory location of the RAM-nodes of a certain discriminator should converge to a certain number. When the number is reached new patterns fed into the network are already recognized by the discriminator and it produces a high output. In order to investigate how well the network is trained, one might look at the performance of the discriminator just before training. When the discriminators are trained on the whole pattern, this gives the result shown in Table. 3. The obtained result allow the identification of the best neural network configuration for enhance generalization to be implemented in hardware. A reasonable solution for this problem is the 9-input neuron because the larger ones make difficult hardware implementation. This configuration has shown the best relationship between cost and recognition level.

Table 3. WNNs Recognition Level

Parameter				
Number neuron input	Total Neuron	Number Train. Pattern	Average Recog.	Average error
3	80	50	82	2%
4	80	50	85	1.8%
5	80	50	90	1.5%
6	70	50	92	1.4%
7	60	60	94	1.3%
8	50	65	97	1.25%
9	30	70	98	1.1%
10	30	75	98	1%

5.3 Comparison

For some applications, where software approach is more flexible because it permit better integrations with other routines of the system. Then, the description of the RAM base neural network can be automatically translated into C language code. Consequently, the network can be directly executed into microprocessor ALU (Arithmetic Logic Unit). This fact decreases the execution time of the algorithm. The advantage of this technique is the employment of the microprocessor ALU to execute the logic operation set of the RAM algorithm. Simple logic functions are faster to execute than complex floating point operation, used by greater number of neural models. Table 4, show comparison the total processing time of software using C code and hardware implementation.

Type of Implementation in low cost microcontroller	Source code	Execution time
MLPNN in C language	30 Kbytes	5 ms
WNN in C language	5 Kbytes	1 μs
WNN (combination in software and hardware)	500 bytes	0.25 μ s

From the result in table 4, the WNNs software solution many times faster than other continues neural network like MLPNN implementation. Exactly, software implementation is flexible and portable comparison that hardware implementation but software speed is not so high. In this work combination of hardware and software solution is more flexible because minimized the resource and fast execution time.

6 Conclusion

Strategy has been developed which allows mobile robot to learn and move around an environment avoiding obstacles and this has been implemented in hardware. The operation of the complete controller that would take hundreds of lines of code to be implemented can be executed with one single instruction that reads the command byte in the memory. It can make the controller work hundreds of times faster. The controller is stimulated with every possible input and a corresponding output table is written. This output table is then stored into the robot RAM memory and becomes the current controller that drives the robot in the current environment. The network was implemented with very modes microcontroller system 20 Kbytes ROM, small amount of data memory about 256 bytes and 500 bytes source code program using combination in the microcontroller assembler and C code. The algorithm only 750 lines of program because . The result shows the mobile robot was able to detect and avoid obstacles in real time.

This work will continue with the development of an embedded real time learning system, implemented according to the same techniques that will make use of the WNNs one shot learning to quickly map the environment where the robot is moving. The application of the presented technique is not restricted to mobile robot application. It can be applied in industrial process, control process, and complex pattern recognition, that demands a high processing capacity.

The application of the presented technique is not restricted to mobile robot applications. It can be applied in complex pattern recognition problem that demands a high processing capacity.

7 Open Problem

The WNNs while fast learning algorithm, lacks generalization power. It is only recognizes previously learned tuple. To overcome this limitation, a set of neurons can be organized, where each neurons is responsible for the learning or recognition of a subset of an input pattern. The sub pattern assigned to each neuron is defined in a randomly created input-neuron mapping, which is used in both learning and recognition phases. Most of the WNNs configurations have the problem of poor growth of storage capacity with the size of the system. To solve this problem, ways to decompose system need to be considered.

The possible motivations for adopting a modular approach, because potential advantage such as reducing model complexity, making the overall system easier to understand and training times can be reduced and a priori knowledge task at hand can be incorporated in terms of devising an appropriate task decomposition strategy. This is required the development of a flexible implementation for investigate the generalization performance on large input space.

ACKNOWLEDGEMENTS.

This work was supported by The Faculty of Computer Science, Sriwijaya University, Indonesia and Softcomputing Research Group, Faculty of Computer Science and Information System Universiti Teknologi Malaysia, Johor Bahru.

References

[1] A. de Carvalho, M.C. Fairhurst and D.L. Bisset. "SOFT – A boolean self organizing feature extractor". In I. Aleksander and J.G. Taylor , editors, Proceedings of the International Conference on Artificial Neural Network, pp 669-672, Brighton, UK, Sepetember, 1992

[2] Botelho, S. C., Simoes, E. D. V., Uebel, L. F., and Barone, D. A. C., "High Speed Neural Control for Robot Navigation". Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Beijing, China, pp. 421-429, 1996.

[3] Borenstein, J., Everett, H.R., Feng, L. "Navigating Mobile Robots: Systems and Techniques", A. K. Peters, Ltd., Natick, MA, USA, 1996

[4] C. Linneberg and T. M. Jrgensen. "N-tuple or RAM based neural network classification system and method". United States Patent 6999950, 2006

[5] Cox, I.J. "Blanche-an experiment in guidance and navigation of an autonomous robot vehicle". Robotics and Automation, IEEE Trans. on 7 pp. 193–204, 1991

[6] E.Filho, M.C.Fairhurst and D.L.Bisset "Adaptive pattern recognition using goal-seeking neurons", Pattern Recognition Letters, v.12, 131-138, march 1991.

[7] I. Aleksander, W. V. Thomas, P. A. Bowden, "Wisard: A radical step forward in image recognition." Sensor Review, pp 120-124, July 1984

[8] I. Aleksander and H. Morton. "An introduction to neural computing". Chapman and Hall, London, U.K., 2 edition, 1995

[9] I. Aleksander and H. Morton. "General neural unit: retrieval performance". IEE Electronics Letters, 27:1776-1778, 1991

[10] J, Austin, "A Review of RAM-Based Neural Networks". Proceedings of the Fourth International Conference on - MICRONEURO94, pp. 58-66,1994

[11] J.Austin and T. J. Stonham, "An associative memory for use in image recognition and occlusion analysis", Image and Vision Computing, v. 5, pp. 251-261, 1987

[12] L. Teresa et al, "Weightless Neural Models: A Review of Current and Past Works", Neural Computer Surveys v. 2,pp41-61, 1999

[13] M. Keat et al.. “Weightless neural network array for protein classification”. In PDCAT, volume 3320 of LNCS, pages 168–171. Springer, 2004.

[14] N.M. Allison and M.J. Jhonson, “ Realisation of Self Organizing Neural Maps in {0,1} space. In J.G. Taylor and C.L.T. Mannion, editors, New Development in Neural Computing.

[15] P. Coraggio and M. De Gregorio. “WiSARD and NSP for Robot Global Localization” In Proc. of the IWINAC, LNCS, volume 4528, pages 449–458, 2007. Springer-Verlag.

[16] Q. Yao, D. Beetner, D.C. Wunsch, and B. Osterloh., “A RAM-Based Neural Network for Collision Avoidance in a Mobile Robot”, IEEE, 2003

[17] R. Al-Alawi, “FPGA Implementation of a Pyramidal Weightless Neural Networks Learning System” International Journal of Neural Systems, Vol. 13, No. 4, pp.225-237, 2003

[18] R. Al-Alawi. “Performance evaluation of fuzzy single layer weightless neural network”. Int. J. of Uncertainty, Fuzziness and Knowlege-Based Systems, 15(3):381–393, 2007.

[19] R. J. Mitchell, D. A. Keating, and C. Kambhampati, “Neural network controller for mobile robot insect,” Internal report, Department of Cybemetics, University of Reading, April 1994

[20] R. Rohwer and M. Morciniec. “A theoretical and experimental account of n-tuple classifier performance”. Neural Computation, 8(3):629–642, 1996

[21] Simoes, E.D.V., “An Embedded Evolutionary Controller to Navigate a Population of Autonomous Robots”, Frontiers in Evolutionary Robotics, I-Tech Education and Publishing, Vienna, Austria Book edited , ISBN 978-3-902613-19-6, pp. 596, April 2008.

[22] Simoes, E. D. V., Uebel, L. F., and Barone, D. A. C., “Hardware Implementation of RAM Neural Networks”. In Pattern Recognition Letters, n. 17, pp. 421-429, 1996

[23] S. Ramanan, R.S. Petersen, T.G.Clarkson and J.G.Taylor “ pRAM nets for detection of small targets in sequence of infra-red images”, Neural Networks, pages 1227-1237, 1995

[24] T. B. Ludermir et al. “Weightless neural models: A review of current and past works”. Neural Computing Surveys, 2:41–61, 1999.

[25] T. M. Jorgensen. “Classification of handwritten digits using a RAM neural net architecture”. Int. J. of Neural Systems, 8(1):17–25, 1997.

[26] T. G. Clarkson, D. Gorse, and J.G. Taylor. “Application of the pRAM. In Proceedings of the Int. Joint Conference on Neural Network, pp. 2618-2623, Singapore, 1996.

[27] T. G. Clarkson et al.,“Speaker Identification for Security Systems Using Reinforcement-Trained pRAM Neural Network Architectures,” IEEE Transaction On Systems, Man and Cybernatics-Part C: Application and Reviews, Vol. 31, No. 1, pp. 65-76, Feb. 2001.

[28] Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press, 2005

[29] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. Bulletin of Math. Biophysics, 5:115–137, 1948

[30] Y. Zhou, D. Wilkins, R. P. Cook, “Neural network for a fire-fighting robot,” University of Mississippi, 1994