Hierarchical Architecture for Incremental Learning in Mobile Robotics

N. H. Siddique, J.V. Condell, T.M. McGinnity, Y. Gatsoulis and E. Kerr

Abstract— **Neural networks have been applied to many real world problems due to their capability of modelling and learning without much *a priori* information about the system. Scalability is a snagging characteristic of neural networks, i.e. it demands huge computational cost for retraining the network if any new feature is to be added to the model. In such architectures there is no stipulation for incremental learning. This paper presents a hierarchical architecture which supports incremental learning. The performance of the proposed architecture has been verified on a Khepera robot.**

I. Introduction

A central issue in current robotics is how to scale up to more complex cognitive abilities, such as learning in a changing environment. Some animals and insects are capable of learning, integrating multisensory cues, real-world navigation, and flexible behavioural choice. As they obtain such competences with relatively small resources, understanding these mechanisms should lead to efficient robot applications. Humans can accumulate skills from experience by going through a cumulative learning process. However, such ability has been largely overlooked in computational modelling research in higher-order human cognition, particularly in defining a hierarchical architecture for learning. Current research introduces frameworks of cognitive models of learning which take granules of information into account. This assumes that human learning processes are not strictly error minimization, but accumulation of learning. The accumulated skills and knowledge are to be represented in a granular and hierarchical form so that new skill and knowledge can be acquired with minimum effort. If a suitable computational model is found to represent knowledge in a granular and hierarchical form, new skills can be learnt using knowledge that was previously learnt.

Manuscript received May 30, 2010. This work was supported by EU FP7-ICT-2007-3: 2.2 Cognitive Systems, Interaction and Robotics PROJECT: IM-CLeVeR (2009-2013).

N.H. Siddique, J.V. Condell, T.M. McGinnity, Y. Gatsoulis and E. Kerr are with Intelligent Systems Research Centre, University of Ulster, Londonderry, BT48 7JL, NI, UK, phone: +44-28-71375340; fax: +44-28-71375470; e-mail: {nh.siddique, j.condell, tm.mcginnity, i.gatsoulis, ep.kerr}@ulster.ac.uk.

Neural structures have been used for such knowledge representation, and have found applications in robotics and other fields [1]. Most of these neural structures are unitary and monolithic. Scalability is the foremost snagging feature of monolithic neural networks. In such architectures there is no stipulation for incremental learning, i.e., if any additional information is to be stored in a neural network, it has to be retrained using the old data used initially along with the new data set. The retraining of the neural networks suffers from the phenomenon of interference or catastrophic forgetting (also known as crosstalk). The crosstalk phenomenon is attributed to the fact that the same set of weights of a neural network has to learn different mappings, whether simultaneously or consecutively. This interference can be of two types: temporal crosstalk - this phenomenon is basically the loss of already learned knowledge by a neural network about a task when it is retrained to perform another task of a different type, or when two or more tasks have to be learned by a single global neural network consecutively [2]. Spatial crosstalk - this phenomenon occurs when a neural network has to learn two or more different tasks simultaneously [2].

It will demand huge computational cost to support an incremental learning and dynamic structure. Therefore, flexibility in the architecture and hierarchy is much more demanding than a monolithic structure. The advantages of modular structures would be incremental learning trained with a small data set and reduced training time. Such modular neural networks present an architecture where an addition of a new module that stores the knowledge is permitted. Both temporal and spatial crosstalk can be avoided by the use of modular structures of neural networks in which tasks can be subdivided, each represented by a single module, and each module with a separate set of interconnecting weights, is responsible for its individual tasks. A module can be pre-trained individually for specific subtasks and then integrated via an integration mechanism or can be trained along with an integrating unit. If the processing can be divided into separate, smaller and possibly parallel subtasks, then the computational effort which will, in

general, be greatly reduced [4]. Such modules can learn sets of functional mappings faster. The modules operate on distinct inputs. The outputs of the modules are mediated by an integrating unit that is not permitted to feed information back to the modules. In particular, the integrating units decide on (1) how the modules are combined to form the final output of the system, and (2) which modules should learn which training patterns.

The objective of this paper is to present the decomposition of a complex network into simpler networks that generate simpler behaviours, and define an integrating function that combines the networks in a hierarchy capable of generating behaviours appropriate to stimuli.

The remainder of this paper is organised as follows: Section 2 describes the hierarchical structure; Section 3 describes the experimental setup using the Khepera robot. Section 4 demonstrates the experimentation on the Khepera robot and some concluding remarks are made in Section 5.

II. HIERARCHICAL ARCHITECTURE

The construction of a hierarchy in general is a difficult task. The difficulty arises from the complexity of decomposition (also known as modularisation) of an overall task into smaller suitable subtasks. Although detailed quantitative models are presumably not available, there is a lot of qualitative information available about the structure of the physical systems and potential relationships among the inputs and outputs. The internal structure proposed here is based on the decomposition of the set of inputs into subsets of related inputs. The decomposition can be done hierarchically, beginning with the smallest possible sets of related inputs, and then integrate using a similar hierarchical structure. Some examples of hierarchical structure are shown in Figure 1.

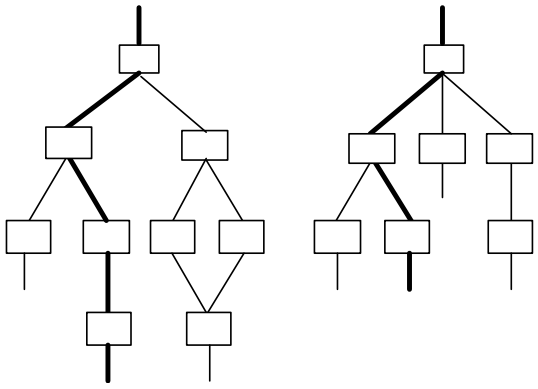

Figure 1: Hierarchical structure of subnets.

The hierarchy is reflected in the structure of the neural network, which starts with small subnets (also known as modules) that focus on the smallest subnets of inputs. These subnets feed their outputs into subnets that focus on larger subsets of the input set. These can be, for example, inputs that refer to a specific small portion of the physical system, a particular phenomenon or constraint, or a single time instant within the time interval. In other words, such a small subset would be localized spatially, temporally or conceptually [5]. A simple modularisation scheme and hierarchical structure is proposed in this paper for generation of behaviours of a mobile robot. The architecture is shown in Figure 2.

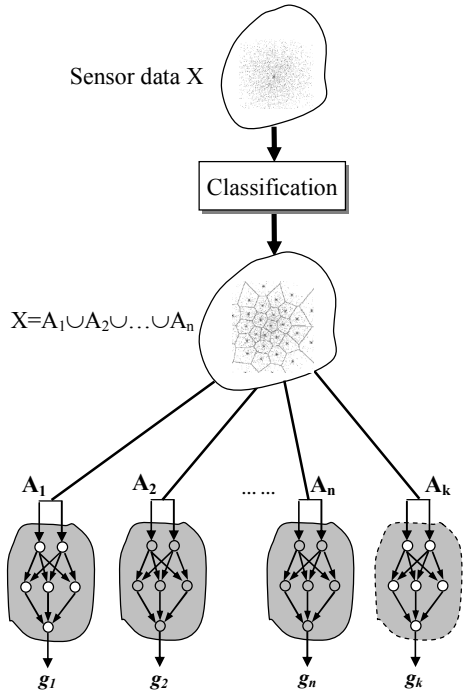

Figure 2: Two stage hierarchical architecture.

The architecture consists of two stage modules: a classification system and Multi-layer Perceptron (MLP) network. The classifier maps high dimensional input data into a low-dimensional matrix. The matrix is topologically arranged in an unsupervised manner and categorises stochastic data into groups by employing some metric measures. Similar input data map to the same class and very different inputs map to distant classes. These classes can be considered as sets of data points or feature data extracted from objects in an environment. The MLP, represents the known classes, it maps between the feature sets and output function g_i defined as $g_i = \Gamma\{[MLP]_i, A_i\}$. MLPs are responsible for individual actions to be performed. Using set

theoretic notations, the architecture can be further explained in the following way.

Let X be a set of features of objects consisting of subsets of features $\{A_1, A_2, \ldots, A_n\}$. A partition of a set X is a set of nonempty subsets of X such that $\forall x \in X$ is in exactly one of these subsets. Equivalently, a set A of nonempty sets is a partition of X if the union of the elements of A is equal to X, mathematically defined as

$$\bigcup_{i=1}^{n} A_i = X \tag{1}$$

A partition of a set X is any collection of non-empty subsets $\{A_i \mid i \in I\}$ of X such that $\{A_i\}$ are pairwise disjoint and

$$\bigcup_{i \in I} A_i = X \tag{2}$$

The elements of A are said to cover X. Each of $A_i\,(i = 1,2,...,n)$ can be thought of a set of elements of features or data points that represent an object or part of an object. Any new data points or set of elements that appear as a distant class will be a disjoint set A_k, mathematically defined as

$$A_k \cap A_i = \varnothing,\ i = 1,2,\ldots,n;\ k = 1,2,\ldots,m \tag{3}$$

If there exists a disjoint set A_k, it must be unknown data of a new situation, object or feature. A new action is to be taken in such situations. Due to the limited processing power of the system, an online learning mechanism is not possible; instead an offline training of a new MLP is to be done when the system is idle. Once the MLP is trained, it is added to the neural structure. This process is also shown in Figure 2. The classifier system can be a Bayesian network classifier [6], Parzen classifier [7], Kohonen SOM [8], Support Vector Machine (SVM) [9], or neural network classifier [10]. Neural networks have emerged as an important classifier tool. Recent research activities in neural classification have established that neural networks are a promising alternative to various conventional classification methods. A recent survey on the most important developments in the neural networks classification research is reported in [11].

III. CLASSIFIER AND MLP ARCHITECTURE

The problem being investigated in this research is to generate action for a mobile robot upon receiving perception signals through infra-red (IR) sensors. Based on the perceptions about the objects and environment, the robot has to produce action to avoid obstacles. There has been much research work carried out on this issue where a single neural net has been deployed. The disadvantages of a single monolithic network have been discussed in Section 1. The aim of this research is to modularize the single monolithic network into simple subnets called modules which will be able to generate simple behaviour or action. The hierarchical structure of the perception-action network along with the Khepera robot is shown in Figure 3.

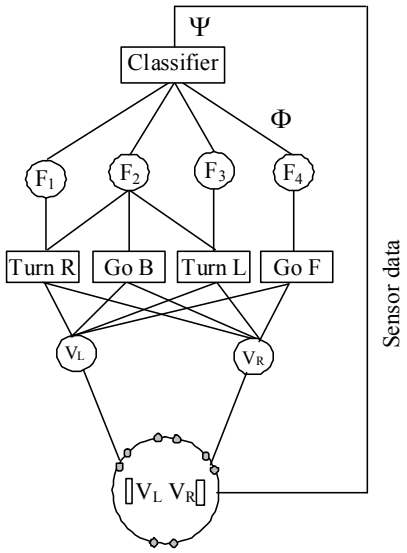

Figure 3: Hierarchical architecture for action generation.

A Khepera robot has been used in the experimentation of this research [12]. It has 8 infra-red sensors and ambient light sensors which can measure distances to obstacles. A functional diagram of a Khepera robot with two wheels and eight infra-red sensors is shown in Figure 4. The practical difficulty with the infra-red (proximity) sensors of the Khepera robot is that they can provide reasonable measurement values for object distances between 10mm and 60mm and measurements are inaccurate if the objects are more than 60mm away. Furthermore, the sensor readings are subject to several other environmental conditions such as material, colour, texture of the object, and the amount of ambient light. False or inaccurate detection are often caused by sensor noise and sporadic or inconsistent measurement. Most of these measurements correspond to the distance of the nearest object in the measurement cone and so occluded objects are not detected.

The eight infra-red sensors $\{s_1, s_2, \cdots, s_8\}$ are used to measure distances. As the action to be taken by the

Khepera robot is going to based on the direction to which an obstacle is approaching, say N actions here, the quantitative distance information measured by the eight sensors $\{s_1, s_2, \cdots, s_8\}$ are transformed into a N-dimensional distance vector Φ by using a classifier system defined by the following equation:

$$\Phi = \Gamma(W\Psi) \tag{4}$$

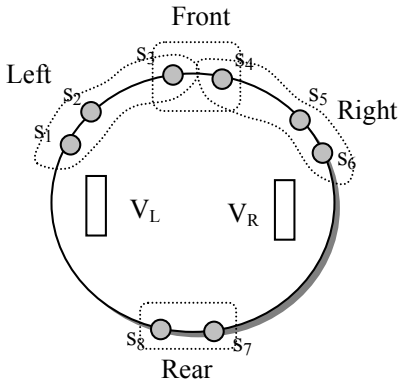

Figure 4: Functional diagram of Khepera robot.

where $\Phi = [F_1 \quad F_2 \quad \cdots \quad F_N]^T$ and $\Psi = [s_1 \quad s_2 \quad \cdots \quad s_8]^T$. W is the weighting matrix of $[N \times 8]$ dimension defined as

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{18} \\ w_{21} & w_{22} & \cdots & w_{28} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{N8} \end{bmatrix} \tag{5}$$

The distance of an obstacle to ith direction will be

$$F_i = w_i \Psi \tag{6}$$

where w_i is a row vector with $i = 1,2,\cdots,N$. The N-dimensional distance vector Φ has another advantage – measuring distance information to eight directions using all sensors is somewhat redundant. Measurement of distances to four directions, i.e., left, front, right and rear will be sufficient for the Khepera robot. These are labeled as F_1, F_2, F_3 and F_4 respectively in Figure 3. It means there are 4 actions to take by the robot: turn right if obstacle approaches from the left, turn left if obstacle approaches from the right, go back if obstacle approaches from the front and go forward if no obstacle is present in front. No action is required if any obstacle is at the rear as the Khepera always strives to move forward. Thus N becomes 4 and the weight matrix becomes a $[4 \times 8]$ matrix. A further reduction in the number of the elements of the weight matrix W $[4 \times 8]$ is possible by devising a simple mechanism, which is to divide the eight sensors $\{s_1, s_2, \cdots, s_8\}$ into four groups for four directions. Each group will measure the distances to obstacles on the left, front, right and rear. Thus the pairs of sensors $\Psi_1 = \{s_1, s_2, s_3\}$, $\Psi_2 = \{s_3, s_4\}$, $\Psi_3 = \{s_4, s_5, s_6\}$ and $\Psi_4 = \{s_7, s_8\}$ will provide distance information of obstacles to the left, front, right and rear as shown in the functional diagram in Figure 4. Most importantly, the grouping of sensors will thus simplify the weighting matrix in (5) which will now require determining only 8 weights instead of 32. Thus the matrix in (5) becomes

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_{23} & w_{24} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{34} & w_{35} & w_{36} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & w_{47} & w_{48} \end{bmatrix} \tag{7}$$

The grouping of the sensors will also counteract the detrimental effects such as aging, thermal and ambient light condition and sudden damage that affect the accuracy of the sensor readings. A mechanism of finding the appropriate weights incorporating all these detrimental effects and uncertainties into w_{ij} is essential at this stage. An analytical approach is not available to hand, so an empirical approach can be applied to determine the eight weight values. The 8 weights of $W = \{w_{ij}\}$ of the classifier system have to be learnt using the sensor data in an unsupervised manner. Implementation of unsupervised learning performs better if trained with redundant input data. Redundancy provides knowledge about statistical properties of input patterns. In fact, there are close relations between statistical approaches to pattern classification and neural network techniques [13]. Therefore, a neural network classifier is developed here. The notion is similar to the Hamming network. Classification is meant to distinguish between the directions of incoming data, which eventually identifies an approaching obstacle. The classifier network proposed here works in a similar principle but significantly different from a conventional Hamming net [14] in that integer input vectors were used instead of binary bipolar vectors. The expanded diagram of the network for classification of integer *n*-tuple input vectors is shown in Figure 5.

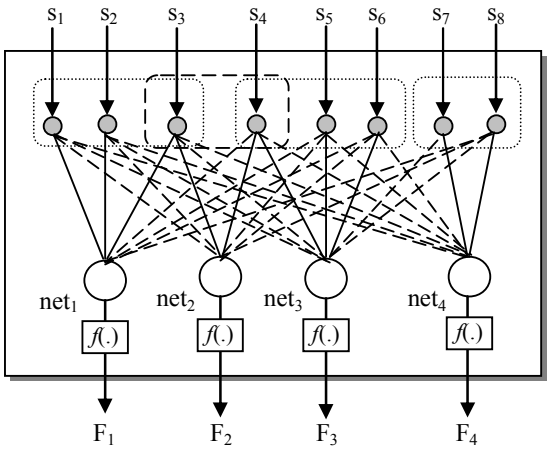

Figure 5: Classifier network for *n*-tuple integer vectors classifying *p* classes.

Assume that the *n*-tuple prototype vector of the *i*-th class is $x^{(i)}$, for $i = 1,2,\cdots,p$ and the *n*-tuple input vector s. The entries of the weight matrix are defined in (7), which connect inputs to the *i*-th neuron. The connectivity of *0*-weights is represented by dotted-lines according to weighting matrix in (7). The *i*-th output is expressed by

$$net_i = \frac{1}{2} w_i s + \frac{n}{2} \tag{8}$$

where the factor $\frac{1}{2}$ is convenient for scaling purposes. $\frac{n}{2}$ is a fixed bias value added to the input of each neuron. The activation function $f(\cdot)$ is chosen as a linear function. The value of the distance is expressed by means of the scalar product of s and $x^{(i)}$. The distance between the two vectors will be a minimum if and only if $s = x^{(i)}$. This would require the weights be $w_i = x^i$, $i = 1,2,\cdots,4$.

The proximity sensors (IR-sensors) can measure a value between 0 and 1023 due to the 10-bit nature of the A/D converter, so $s_j \in [0,1023]$. A sensor value of 1023 indicates that an object is very close and a sensor value of 0 indicates that the robot does not receive any reflection of the infra-red signal meaning an object is far from the robot. If the weights are set equal to the normalized sensor value, i.e., $w_i = \bar{s}$, a weighted sum of the inputs can be obtained. This ultimately justifies the grouping of the sensors. The normalization of the inputs of the sensor values is performed using equation (9).

$$\bar{s}_{j,norm} = \frac{1}{1023} \times s_j, \bar{s}_{j,norm} \in [0,1] \tag{9}$$

where $j = \{1,2,\cdots,8\}$. The classifier output F_i will thus be

$$F_i = f\left(\frac{1}{2}\bar{s}_{ij} s + \frac{n}{2}\right),\ i = 1,2,\cdots,4 \tag{10}$$

It is to be noted that $w_{ij} = \bar{s}_{ij}$ in (7) where $w_{ij} \neq 0$. F_i consequently provides the distance of an approaching object to *i*-th direction and acts as an input to MLP network of the next hierarchy as shown in Figure 3.

The structure of the MLP is shown in Figure 6. The MLPs take F_i as input and generate torques for the two outputs that drive the Khepera's wheels. Each of the MLP's two outputs $\{V_L, V_R\} \in [-10,+10]$ are the velocities of the left and right wheels of the Khepera robot.

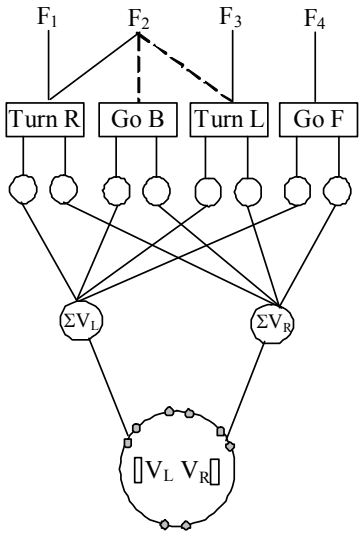

Figure 6: Structure of the MLP

A velocity of $\{\pm 10\}$ will be too fast to control the navigation in maze environment in Figure 7, so it was decided to keep the velocity within a slower range of $\{\pm 5\}$. The set of four different actions {Turn right, Go back, Turn left and Go forward}, as shown in Figure 6, are trained in a supervised manner using collected data for four possible situations. If an object is right in front of the robot, it has three choices turn right, go back or turn left. Rather than training it for three actions, only one fixed action was chosen, e.g. turn right, thus reducing training time. By default the robot is trained to go forward.

IV. EXPERIMENTAL RESUTS

The Khepera robot, has a diameter of 55mm, a height of 30mm, and a weight of 70g. Since the sensor data from the robot is imprecise and local, it cannot localise itself in a global map. The two wheels are controlled by two DC motors with incremental encoders that move in both directions. It has an on-board 68331 processor and can also be controlled by an off-board computer via a serial link. Its convenient size, ready availability and the fact that it is straightforward to program, has made it a very popular tool for simple autonomous robotics experiments. Two simple experiments are carried out with the Khepera robot to verify the performance of the developed methodology: straight-line movement, and maze following movement.

Figure 7 shows the experimental set up for a straight-line movement of the Khepera root. The robot has to move straight ahead up to the next wall and come back. If the Khepera robot moves in a straight line, the left and right wheel speeds will be the same or very close to each other. Figure 8 shows the distances of obstacles or walls to the left, front, right and to the rear represented by F_1, F_2, F_3 and F_4 respectively that are calculated using Equation (10). A high value indicates an obstacle is close and a low value indicates an obstacle is far from the robot. The Khepera robot then starts traveling from one wall to the other wall. Figure 9 shows the corresponding left and right wheel speed that makes the robot avoid obstacles and move straight ahead. On approaching the wall, the robot starts turning to the left indicated by decreasing left wheel speed and increasing right wheel speed.

Figure 7: Experimental set up – straight line movement.

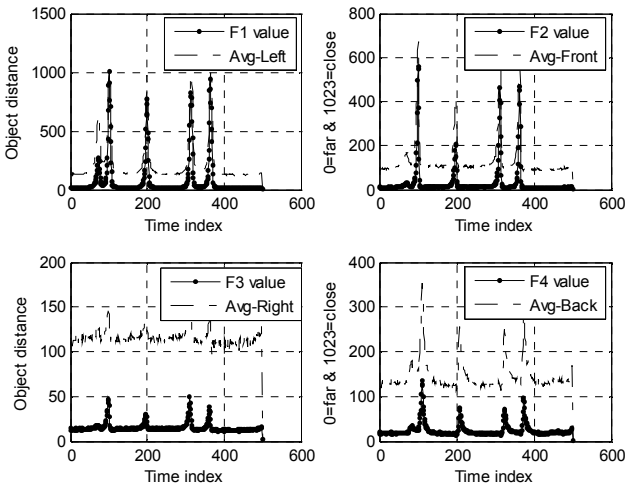

Figure 8: Obstacle distances $\{F_1,F_2,F_3,F_4\}$ vs avg of {Left, Front, Right, Rear} in the straight-line environment.

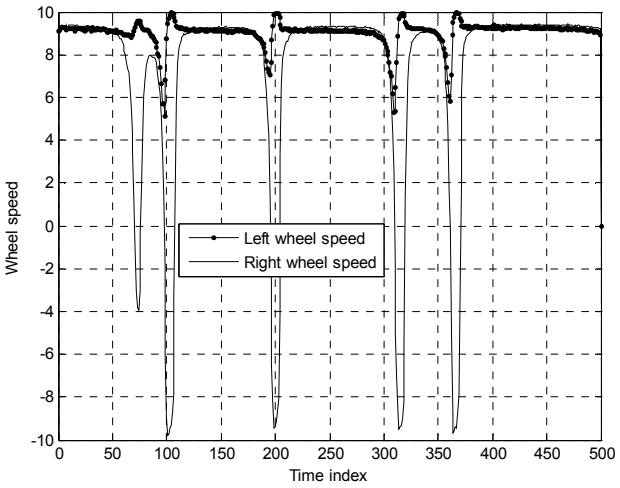

Figure 9: Left and right wheel speeds in the straight-line movement.

In a second investigation the maze following behaviour has been studied. The maze (environment) was created randomly as shown in the experimental setup in Figure 10 and no path planning algorithm was used. The Khepera robot successfully follows the maze in the test. High values indicate an approaching obstacle or wall. Figure 11 shows object distances $\{F_1,F_2,F_3,F_4\}$ calculated using Equation (10) indicating an approaching obstacle to different directions. Figure 12 shows the corresponding left and right wheel speed that makes the robot avoid obstacles and follow the maze. Figure 12 indicates that the robot maintained a safe distance from the walls within the maze and navigates correctly through the maze.

Figure 10: Experimental set up – maze following movement.

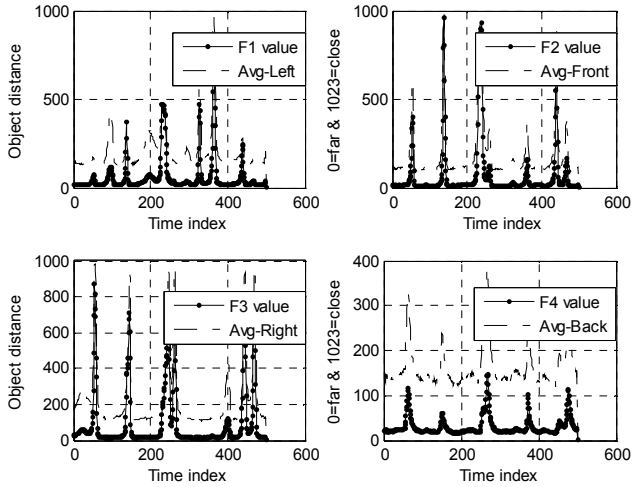

Figure 11: Approaching obstacle distances $\{F_1,F_2,F_3,F_4\}$ vs avg of {Left, Front, Right, Rear} in the maze following environment.

V. CONCLUSION

This paper investigated mechanisms of how new knowledge can be represented in a hierarchical form and how this learnt knowledge can be reused to reduce training time and maximise benefit.

The advantages of the two stage hierarchical structure are extendibility and incremental learning. Any new behaviour can be accommodated by training a new MLP in the second level without making any modification or retraining of any of the MLPs. In this paper classification was straightforward due to the nature of data collected from a homogeneous set of sensors. Classification will be much more complex and computationally intensive if data are collected from a set of heterogeneous sensors or from a stereo-vision system. In such cases, a further level of hierarchy may be required for the classification and even for the action MLPs. These issues will be investigated in future research.

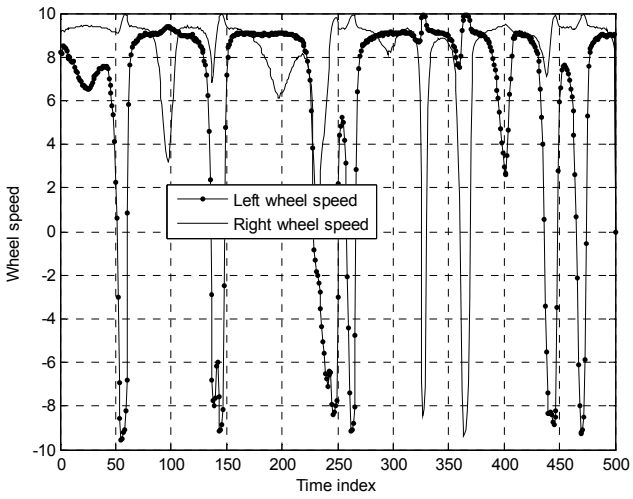

Figure 12: Left and right wheel speeds in the maze following environment.

REFERENCES

[1] M. Kawato, Computational Schemes and Neural Network Models for Formation and Control of Multi-joint Arm Trajectory, Chapter 9, *Neural Network for Control*, The MIT Press, Cambridge, Massachusetts, London, England, pp. 197-228, 1990.

[2] R. M. French, Catastrophic Forgetting in Connectionist Networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

[3] M. I. Jordan and R. A. Jacobs, A competitive modular connectionist architecture. In *Advances in Neural Information Processing Systems 3*, pages 767–773, San Maeto, CA, Morgan Kaufmann Publisher Inc, 1991.

[4] S. Elfwing, E. Uchibe, K. Doya, and H.I. Christensen, Evolutionary Development of Hierarchical Learning Structures, *IEEE Transaction on Evolutionary Computation*, Vol. 11, no. 2, pp. 249-264,1997.

[5] M. L. Mavrovouniotis and S. Chang, Hierarchical Neural Networks, *Computers in Chemical Eng.*, Vol. 16, No. 4, pp. 347-369, 1992.

[6] S. Monti, and G. F. Cooper, A Bayesian network classifier that combines a finite mixture model and a Naïve Bayes model. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, pp. 447–456, 1999.

[7] K. Fukunaga, R.R. Hayes, The Reduced Parzen Classifier, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 4, pp. 423-425, 1989.

[8] T. Kohonene, *Self-organising Maps*, Springer Verlag, 3rd Extended Version, 2001.

[9] Corinna Cortes and V. Vapnik, Support-Vector Networks, *Machine Learning*, Vol. 20, 1995.
[10] M. D. Richard and R. Lippmann, Neural network classifiers estimate Bayesian *a posteriori* probabilities, *Neural Computing*, vol. 3, pp. 461–483, 1991.
[11] G.P. Zhang, Neural Networks for Classification: A Survey, IEEE Transactions on Systems, Man and Cybernetics –Part C: Applications and Reviews, Vol. 30, No. 4, pp. 451-462, 2000.
[12] Khepera Robot, http://www.k-team.com/kteam/home.php, 26 May , 2010.
[13] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York, 1973.
[14] R. P. Lippmann, An Introduction to Computing with Neural Nets, *IEEE Magazine on Acoustics, Signal and Speech Processing*, pp. 4-22, 1987.