I²MTC 2008 – IEEE International Instrumentation and Measurement Technology Conference
Victoria, Vancouver Island, Canada, May 12–15, 2008

Neural Networks for Environmental Recognition and Navigation of a Mobile Robot

Moufid Harb[1], Rami Abielmona[2], Kamal Naji[3], and Emil Petriu[1]
[1]School of Information Technology and Engineering (SITE) Departement, Engineering Faculty, University of Ottawa, Ottawa, ON, Canada, K1N 6N5
[2]Larus Technologies Corporation, Unit 1, 58 Antares Drive, Ottawa, ON, Canada, K2E 7W6,
[3]Department of Electrical Engineering, Faculty of Electrical Engineering, Damascus University, Damascus, Syria

Abstract *- Mobile robots could play a significant role in places where it is impossible for the human to work. In such environments, neural networks, instead of traditional methods, are suitable solutions to locally navigate and recognize the environment's subspaces. In order to learn and perform two important functions "environmental recognition" and "local navigation", multi-layered neural networks are trained to process distance measurements received from a laser range-finder. These neural networks are a major component of the control system of a mobile robot dedicated for industrial applications. This paper will focus on a computer based design and test of a neural system, that includes three neural controllers for local navigation, and two neural networks for environmental recognition, fed off-line by a simulated model of a laser range-finder.*

Keywords *– Environmental Recognition, Autonomous Robotic Navigation, Neural Networks*

I. INTRODUCTION

With the progress of technology and research in the field of intelligent systems and the development of the quality and quantity of industrial production based on total automation systems, industrial applications of mobile robotics has become one of the production line components. In addition, for some kinds of industries, there is an increased demand for robotics in complex circumstances and conditions of work, where it is very difficult and dangerous for humans to access, and often impossible to work in, such as chemical and medical industries, places of high temperatures, contaminated areas and purified laboratories.

Conventional methods for controlling mobile robots (MRs) show slow processing of data resulting from a sensory unit mounted on a MR, because of the serial processing of that data [1]. These methods need great efforts in programming to deal with nonlinear functions, as well as the difficulty of adapting to dynamic environments. Additionally, traditional approaches for robotic navigation are based on a detailed, accurate metric description of the environment [2], such as potential fields [3] and graph search methods [4]. Where it is difficult to perform in real-time applications [5], the prompt and high-speed processes for the input data are very important, especially if the kinematic constraints of the robot are taken into account [2]. Moreover, since the conventional methods represent the path in terms of artificial coordinate systems, they are highly vulnerable to spatial inaccuracies due to the sensory devices and movement actuators.

II. MODELING APPROACH

In this research, the modeling of a MR is executed using a personal computer. All processes of mobile movement control, measurements, mapping of working places, preparation and processing of data base are required for neural network (NN) training, and comparison of results are performed by simulation using a high level programming language. The Matlab tool was used to train the NNs.

A. The Robot Model

As shown in Fig. 1, a differential drive vehicle was selected, where steering is affected by the difference in velocity of the tracked wheels. These wheels can be driven by two low-level DC motors located on either side of the center of the robot [6].

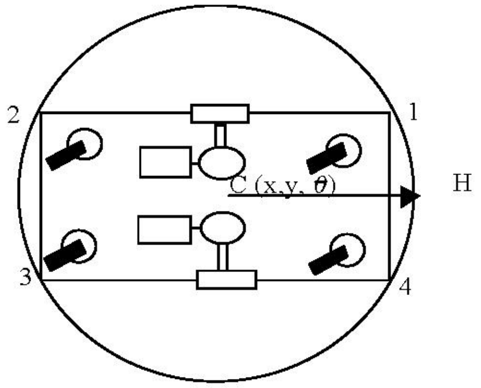

Fig. 1. MR model of differential drive vehicle

This type of MR has the ability to turn in place. The maximum dimension of the rectangular robot is 1 meter, which equals the diameter of the circle that surrounds the robot. Forward orientation angle θ, and Cartesian coordinates X_C and Y_C determine the vector that represents the location of the robot. For distance measurements and direction of obstacles, a laser range-finder [7] for the robot sensory system has been suggested. To simulate the measurements of the proposed laser range finder, an algorithm [6] is used where the measured distances are relative to the MR dimensions (i.e. diameter of the circle, which surrounds the robot).

The correct modeling of the mobile robot was proved by simulation tests which were conducted using the Simulink tool. The test proved that the model of the mobile robot is able to respond to the required changes in terms of both speed and direction [6].

III. ENVIRONMENTAL RECOGNITION

Fig. 2 shows the distribution of the selected measurements on the robot body. 19 directions regularly distributed with a separation angle of 15° surround the robot. That aims to cover most of the environment around the robot.

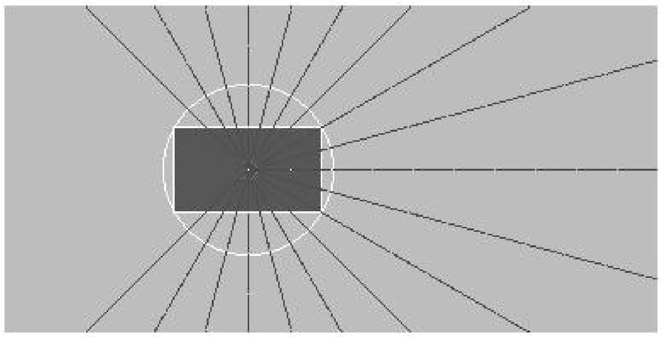
Fig. 2. Laser range-finder distance measurements for environmental recognition

In order to plan a mission path that a MR should follow, the construction map of the MR working place is considered available. The path tracking algorithm can then be run while recognizing indoor subspaces whether the drive system is on-line or off-line. This path can be classified according to a symbolic description of indoor subspaces, which can be divided into a number of places, such as corridors and cross-roads.

A. Environmental Classification NNs

A multi-layer NN model was chosen for this research [8]. It consists of three layers: the input, hidden and output layers, with a log-sigmoid function to activate the neurons. Two NNs were designed for environmental recognition: NN1 and NN2. Fig. 3 shows the input layer of NN1 that consists of 19 neurons, which associate with the output signals of the laser range-finder of the MR's measuring system. The hidden layer consists of 19 neurons; this was found, through experimentation, to be the most suitable number for that layer.

Note that the number of neurons in the output layer represents the number of subspaces to be classified. These subspaces can be considered to be seven subspaces [9], or eleven subspaces [10].

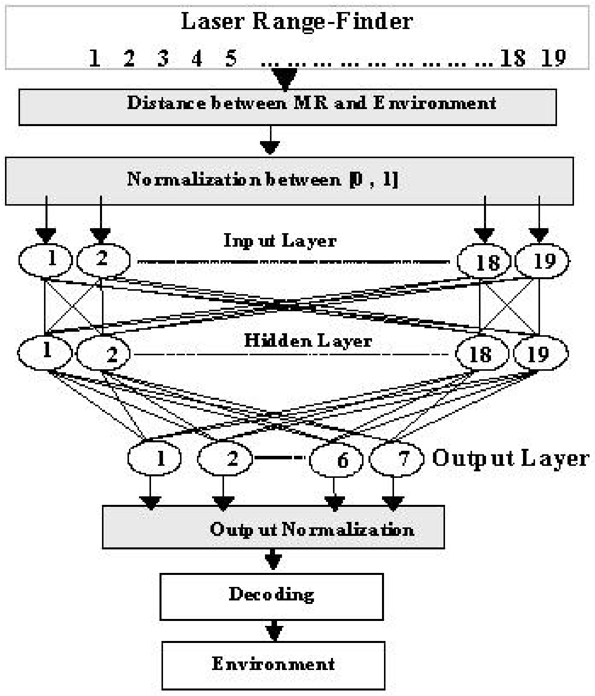

Fig. 3. NN1 structure diagram

Fig. 4 illustrates seven standard subspaces that are recognized by NN1, which are: corridor (CO), cross roads (CR), frontal T shaped cross-road (TF), left junction T shaped cross-road (TL), right junction T shaped cross-road (TR), left corner (LC), and right corner (RC).

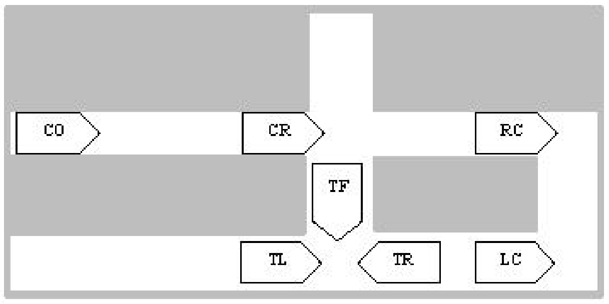

Fig. 4. NN1 classification subspaces

NN2 was designed and trained for environmental recognition to cover the other five subspaces. NN2 has the same number of layers as NN1; however differs in the number of the neurons in the output layer (five). These represent five subspaces that could face the MR during its movement, and are defined as follows: dead end (DE), exit (EX), entrance (En), right angle wall (AW), and room (RM). Fig. 5 shows these subspaces. In this paper we will concentrate on the training and testing of NN1. The reader is referred to [6] for a more detailed explanation of NN2.

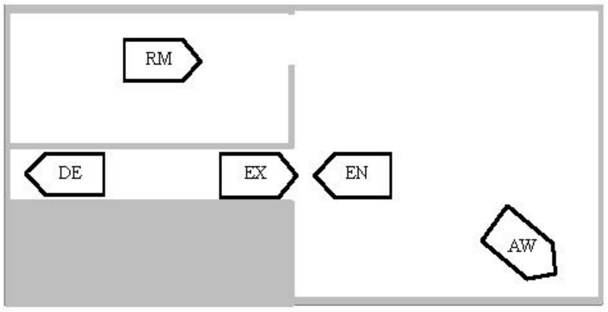

Fig. 5. NN2 classification subspaces

B. Data Base for Network Training

To train the NN, one must provide a suitable quantity of input vectors with the desired output vectors. This allows the NN to generalize previously unseen cases, as well as make correct approximations.

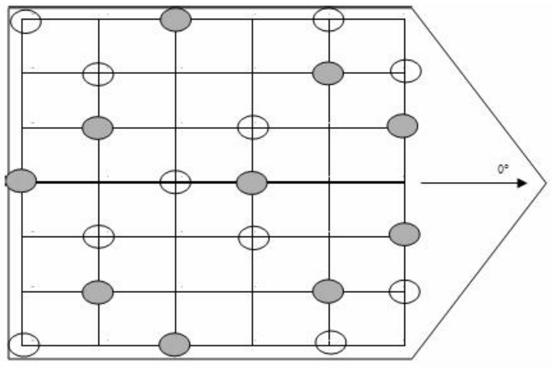

Fig. 6. Training and testing positions of NNs

Training areas were divided into a regular locations array. Eleven locations were selected for training in every subspace of the seven subspaces shown in Fig. 4. The orientation of the robot was varied by 15 ° within an angular sector of 90° (i. e. ± 45° at the front direction of the subspace as shown in Fig. 6).

C. NN Learning

To train the designed NN, the learning algorithm of back propagation was used because it is suitable for multi-layer NN learning. To avoid the problem of local minima, the technique of back propagation with momentum was used. This is due to its better and faster performance when compared to the standard back propagation algorithm [11]. A random function was selected to generate initial small weights suitable for the NN.
A learning rate of 0.001 was introduced to the algorithm to avoid losing the main learning direction when introducing unusual training vectors to the NN [12]. After 30000 training epochs, NN1 converged with Sum Squared Errors (SSE) of 15.17.

D. NN Performance Test

To check the capability of the NN for generalization, the robot was placed in differing locations from those used for training, as shown in Fig. 6. Ten locations were selected in every subspace. The robot was directed into three frontal directions in every location, -45° right, 0° front, and 45° left, assuming 0° is the front direction of the subspace as shown if Fig. 1.

Table 1. Critical cases of NN1

Actual Place	**NN1's actual output**							**NN1's desired output**						
	7	**6**	**5**	**4**	**3**	**2**	**1**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
CO	0	0	0	0	0	0	0.25	0	0	0	0	0	0	1
CR	0	1	0	0	0	0.01	0	0	0	0	0	0	1	0
CR	0	0.08	0.01	0	0	0.29	0	0	0	0	0	0	1	0
CR	0	0	0.13	0	0	0.43	0	0	0	0	0	0	1	0
CR	0	1	0	0	0	0	0	0	0	0	0	0	1	0
CR	0	0.99	0	0	0	0.05	0	0	0	0	0	0	1	0
RC	0	0	0	0.15	0.28	0	0	0	0	0	0	1	0	0
TR	0	0	0.58	0	0	0.14	0.01	0	0	1	0	0	0	0
TR	0	0.75	0	0	0	0	0	0	0	1	0	0	0	0
TR	0	0	0.05	0	0	0.01	0	0	0	1	0	0	0	0
TL	0	0.09	0	0	0	0	0	0	1	0	0	0	0	0
TL	0	0.02	0	0	0	0	0	0	1	0	0	0	0	0
TF	0.96	0	0	0.46	0	0	0	1	0	0	0	0	0	0

The performance of NN1 was tested, with Table 1 showing the critical locations for the NN. This table provided insightful information concerning the locations that NN1 shall be trained at. The poorest result was for the CR, due to the MR's proximity to the wall where the scanning sensory sector of the robot was quite small. Therefore, additional training measurement vectors of the critical locations were introduced to NN1 to allow for a more complete learning cycle.

IV. LOCAL NAVIGATION

For local navigation of a MR, three neural controllers were designed and trained. These controllers are termed: "Keep Right", "Keep Left", and "Pass a Cross-Road", where:

1) The "Keep Right" controller is to keep the robot at the right side of the surroundings,
2) The "Keep Left" controller is to keep the robot at the left side of the surroundings, and
3) The"Pass a Cross-Road" controller is to pass a cross-road.

Furthermore, these neural controllers can avoid obstacles that could be found in a MR's way. They also receive commands from a decision maker that plays the role of global navigator, which will be discussed in a future paper.

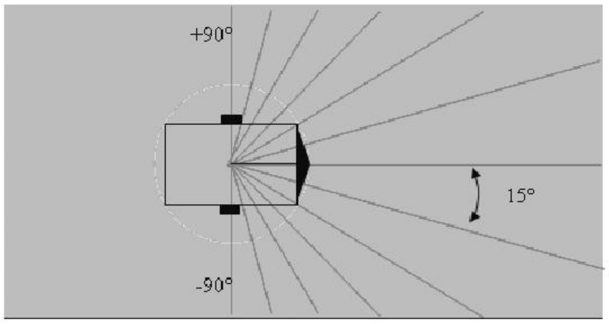

Fig. 7. Laser range-finder distance measurements for local navigation

For the purpose of local navigation and to achieve a higher resolution in distance measurements, the measuring range of the laser range-finder was limited to 3.5 meters. This helped in distinguishing the nearby obstacles, when the measurement vector was normalized to a value between 0 and 1. As shown in Fig. 7, thirteen measurements were taken within a measuring sector of –90° to 90° to the frontal direction of the robot, by intervals of 15°. These measurements were fed to the NN that has the structure shown in Fig. 8.

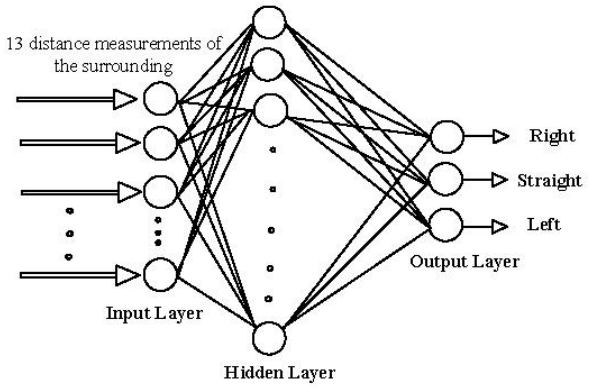

Fig. 8. Local navigation NN's structure

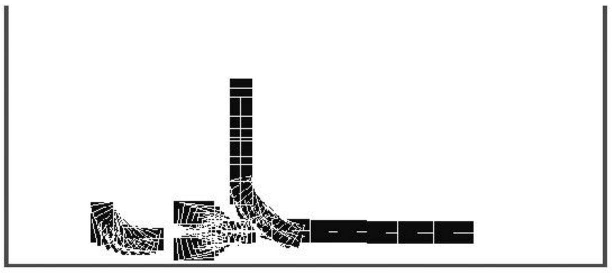

Fig. 9. Follow wall training locations and directions

A. Design of the "Keep Right" Neural Controller

To train the "Keep Right" neural controller, a data base was collected by locating the MR on 111 positions as shown in Fig. 9 and Fig. 10. The required steering signal was considered to move the robot a step, while trying to keep it at a safe position from the obstacles. The data base contained the training vectors, where every vector had two parts. The first part represented the sensory input of the NN shown in Fig. 8. These are the thirteen distance measurements of the laser range-finder. The second part represented the desired output indicating the steering signal in binary code, for example (1 0 0: is Turn Left), (0 0 1: is Turn Right), and (0 1 0: is Go Straight).

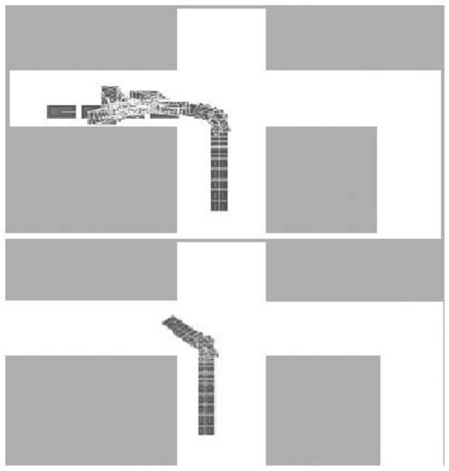

Fig. 10. Turn right at cross-road training locations

Furthermore, to train the "Keep Right" neural controller, the Matlab Neural Networks Tool Box was used to run the algorithm of back propagation with momentum and adaptive learning rate [8]. Fig. 11 shows the strategy of the training technique.

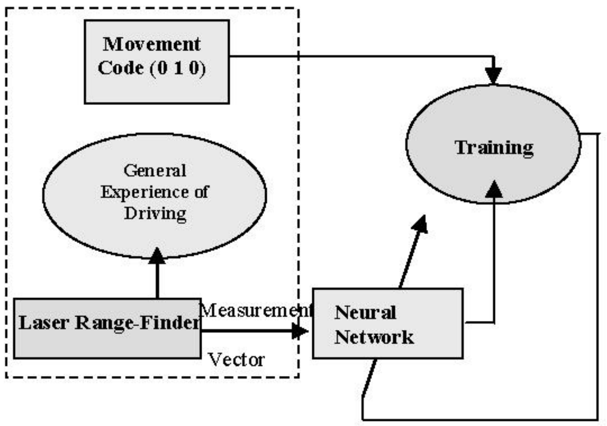

Fig. 11. Training of neural networks for local navigation

After 92000 training epochs, the weights converged when the total SSE was equal to 2.000011. The test was done for the "Keep Right" neural controller in an unknown environment, and it proved to exhibit a good performance as shown in Fig. 12.

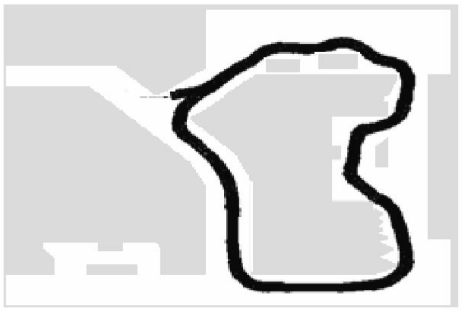

Fig. 12. "Keep Right" neural controller test in an unknown environment

B. Design of the "Keep Left" Neural Controller

To design the "Keep Left" neural controller, it was easier to use the "Keep Right" neural controller to generate the appropriate steering signals by spatially mirroring the sensory inputs of the "Keep Right" neural controller, and changing the sign of the steering signal [2].

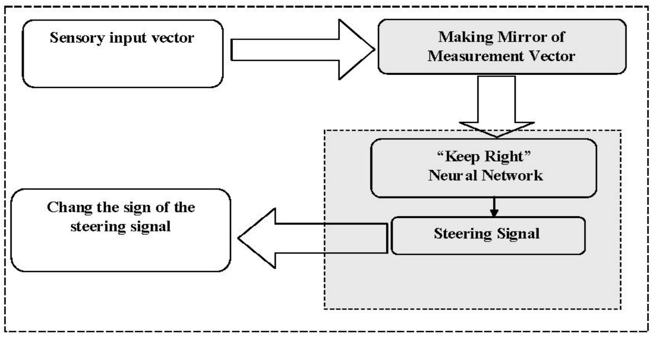
Sensory input vector
Making Mirror of Measurement Vector
"Keep Right" Neural Network
Chang the sign of the steering signal
Steering Signal

Fig. 13. Algorithm to convert to "Keep Left" controller

Fig. 13 shows the algorithm used to convert to the "Keep Left" controller.

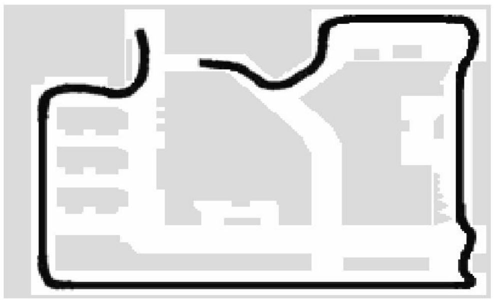

Fig. 14. "Keep Left" neural controller test in an unknown environment

As shown in Fig. 14, the test was conducted for the "Keep Left" controller in a different environment and it proved to exhibit a satisfactory performance.

C. Design of the "Pass a Cross-Road" Neural Controller

The design structure of this controller is similar to the design of the "Keep Right" neural controller. The main difference is the training data base introduced to the neural network to learn its task. Similar to the above-mentioned technique, data base training vectors, were collected by locating the MR at 47 positions as shown in Fig. 15.

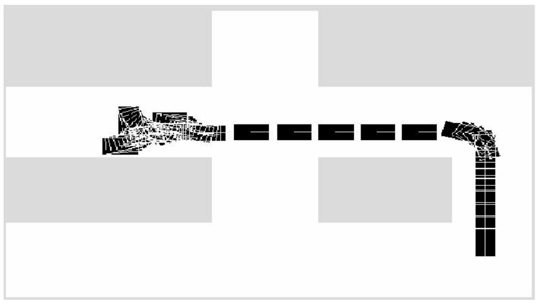

Fig. 15. Pass a cross-road and then keep right training locations and directions

The training vectors were enough to train the NN for passing a cross. The network weights converged after 43564 epochs. The SSE for the training vectors was equal to 1.10^{-6}, which was the targeted error. Fig. 16 shows the test of the "Pass a Cross-Road" neural controller. It can be noticed that the MR can pass the cross-road when it recognizes the cross.

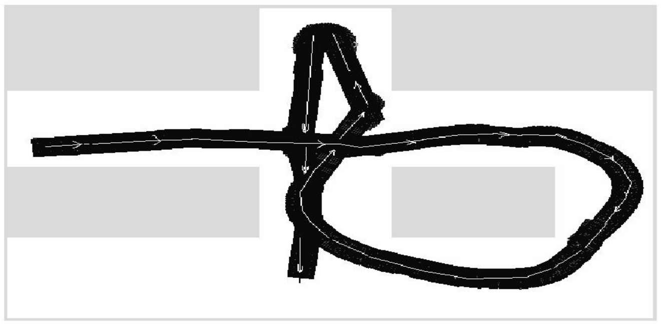

Fig. 16. "Pass a Cross-Road" test

V. CONCLUSION

Most mobile robotic systems are non-linear systems with kinematics that are very difficult to control by traditional methods, while, multi-layered neural networks have proved their competence to control mobile robotic systems, particularly when faced with real-time obstacle detection and avoidance. This paper discusses the local navigation of a mobile robot performing environmental recognition in industrial applications. For environment recognition two neural networks, NN1and NN2, were designed, trained and tested. In this paper, we only focused on the design of NN1 that can recognize seven standard subspaces: corridor (CO), cross roads (CR), frontal T shaped cross-road (TF), left junction T shaped cross-road (TL), right junction T shaped cross-road (TR), left corner (LC), and right corner (RC). For local navigation, three neural controllers, "Keep Right", "Keep Left" and "Pass a Cross-Road" were designed and trained to perform real-time direction control and obstacle avoidance.

The paper's main contributions are a novel training technique based on the method of creating a flat files data base, and the number of utilized neural networks that were used in the control system of the MR. By comparison, the number of neural networks in [10] is equal to the number of subspaces to be classified, while in [9], one neural network to recognize only a total of 7 subspaces, was designed and trained by locating the mobile robot randomly in the training subspaces using the uniform distribution function. In this paper, the training was guided; the locations were previously determined, which differs from the technique used in [10] and [9]. The use of two neural networks, NN1 and NN2, allows for a successful recognition of 12 subspaces.

Future improvements include performing the global navigation function that is concerned with predefined path tracking and autonomous path planning, using the neural networks and neural controllers developed in this paper.

Future research includes the integration of the developed autonomous navigation scheme to the intelligent sensor agents (ISAs) presented in [13]. These agents are intended for the autonomous investigation of relevant parameters in natural living environments, industrial or laboratory hazardous environments, polluted environments, water treatment plants, nuclear stations, war zones, or remote difficult to reach environments such as mining and deep-sea exploration. As individual ISAs only offer local and limited information about the environment, multiple and diverse ISAs are needed to cover the large area and multi-parameter natural environments that are monitored in real-life situations.

A robust local navigation and environmental recognition scheme is required for the resolution of the simultaneous localization and mapping (SLAM) problem that exists within intelligent robotic agents traversing unstructured environments. The research presented in this paper provides a proven and deployable methodology for the control system of each of the ISAs.

REFERENCES

[1] Sklarenko, E. G., " Control system for robotic electric drive using Neural models", Conference of problems of automotive electrical drive, Yalta, Ukraine, pp. 385-387, 1998.

[2] Zalzala, A. M. S. and A. S. Morris, "Neural Networks for Robotic Control: Theory and applications", Ellis Horwood, 1996.

[3] Badcock, J.M. and Dun, J. A., "An autonomous robot navigation system-integrating environmental mapping, path planning, localization and motion control", Robotica Journal, volume 11, pp. 97-103, 1993.

[4] Antonio, J. , Madrigal, F., and González, J., "Multihierarchical Graph Search", IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 24 Issue 1, IEEE Computer Society, 2002.

[5] Habib, Maki K., "Real Time Mapping and Dynamic Navigation for Mobile Robots", International Journal of Advanced Robotic Systems, ISSN 1729-8806, pp. 323-338 323, Vol. 4, No. 3, 2007.

[6] Harb, M., "Computer modelling of industrial mobile robot and control by incorporated neural networks and fuzzy logic", University of Damascus, Syria, PhD Thesis, 2000.

[7] Nof, S. Y., "Handbook of Industrial robotics, 2nd edition", John Willy & Sons, Inc., ISBN 0471177830, 1999.

[8] The Math Works, "Matlab User's Guide", Math Works, Inc.

[9] R. Biewald, "A neural network controller for the navigation and obstacle avoidance of a mobile robot", In Neural Network for Robotic Control, A. M. Zalzala and A. S. Morris, Eds., Ellis Horwood, 1996.

[10] Al-Allan, S., "Environment recognition and reactive navigation of an autonomous mobile robot using neural networks", University of EVRY, France, Ph.D. Thesis, 1996.

[11] Kartalopoulos, S. V., "Understanding of Neural networks & Fuzzy logic, basic concepts and applications", IEEE Press, 1996.

[12] Fausett, L., "Fundamentals of Neural networks", Prentice Hall, 1994.

[13] Abielmona, R., Petriu, E.M., and Whalen T., "Multi-Agent System Information Fusion for Environment Monitoring", Instrumentation and Measurement Technology Conference (IMTC), ISBN 0-7803-9359-7, pp. 1774-1779, April, 2006.