Robotica (2003) volume 21, pp. 79–93. © 2003 Cambridge University Press
DOI: 10.1017/S0263574704526 Printed in the United Kingdom

Navigation of multiple mobile robots using a neural network and a Petri Net model

D.T. Pham and Dayal R. Parhi

Intelligent Systems Laboratory, Manufacturing Engineering Centre, School of Engineering, Cardiff University, Cardiff CF24 3TE (UK)

(Received in Final Form: August 22, 2002)

SUMMARY

This paper describes the use of a hybrid system comprising a multi-layer perceptron and a Petri Net model to control the navigation of multiple mobile robots in an unknown and cluttered environment. The multi-layer perceptron is trained with examples representing the typical static obstacles to be encountered by the robots and the evasive actions to be taken. The Petri Net model embodies the rules describing how the robots should move in order to avoid colliding against one another. The hybrid system has been implemented in simulation software as well as in actual mobile robots. The paper presents the results of simulation and practical tests that demonstrate the ability of groups of robots incorporating the proposed navigation system to negotiate various types of obstacles and find targets successfully.

KEYWORDS: Multiple mobile robots; Navigation; Neural network; Petri Net

1. INTRODUCTION

Mobile robot research has spanned some four decades. Initially, efforts focussed on building individual units able to navigate autonomously.[1–3] These units grew in intelligence and capabilities as computer and sensor technologies developed.[4–6] A new direction of work started in the last decade with the design of systems involving multiple robots.[7] The aim of such systems is, through collaboration between different robots, to perform tasks that individual robots cannot.

In a multiple robot system, navigation control is clearly more difficult than in single mobile robots. This paper describes a system for controlling the navigation of several mobile robots in an unknown and cluttered environment containing many obstacles. The system employs a multi-layer perceptron (MLP) neural network and a Petri Net model.

The function of the MLP, which resides in the controller of each robot, is to compute the steering angle for the robot to enable it to avoid static obstacles. The advantage of adopting a neural network for this task is that it can readily be trained to handle different kinds of obstacles using a limited training set of typical scenarios. This advantage has been recognised by other researchers who have also experimented with neural-network-based navigation control, though not in the context of multiple robot navigation.[8–10]

The much simpler and better defined task of ensuring that the robots do not collide against one another is implemented with a Petri Net model. The latter is eminently suitable for representing the deterministic collision avoidance rules involved in this task. However, as far as the authors are aware, this method of achieving collision-free navigation in a multiple robot environment has not been attempted previously.

The remainder of this paper is organised as follows. Section 2 describes the MLP developed for static obstacle negotiation. Section 3 discusses the Petri Net model for inter-robot collision avoidance. Section 4 describes the testing, in simulation and with actual mobile robots, of the proposed hybrid navigation system. Section 5 concludes the paper.

2. ANALYSIS OF NAVIGATION METHOD

2.1. Obstacle avoidance and target seeking

The neural network used is a four-layer perceptron.[11] The chosen number of layers, was found empirically to facilitate training. The input layer has four neurons, three for receiving the values of the distances from obstacles in front and to the left and right of the robot and one for the target bearing. If no target is detected, the input to the fourth neuron is set to 0. The output layer has a single neuron, which produces the steering angle to control the direction of movement of the robot. The first hidden layer has 10 neurons and the second hidden layer has 3 neurons. These numbers of hidden neurons were also found empirically. Figure 1 depicts the neural network with its input and output signals.

The neural network is trained to navigate by presenting it with 200 patterns representing typical scenarios, some of which are depicted in Figure 2. For example, Figure 2a shows a robot advancing towards an obstacle, another obstacle being on its right hand side. There are no obstacles to the left of the robot and no target within sight. The neural network is trained to output a command to the robot to steer towards its left.

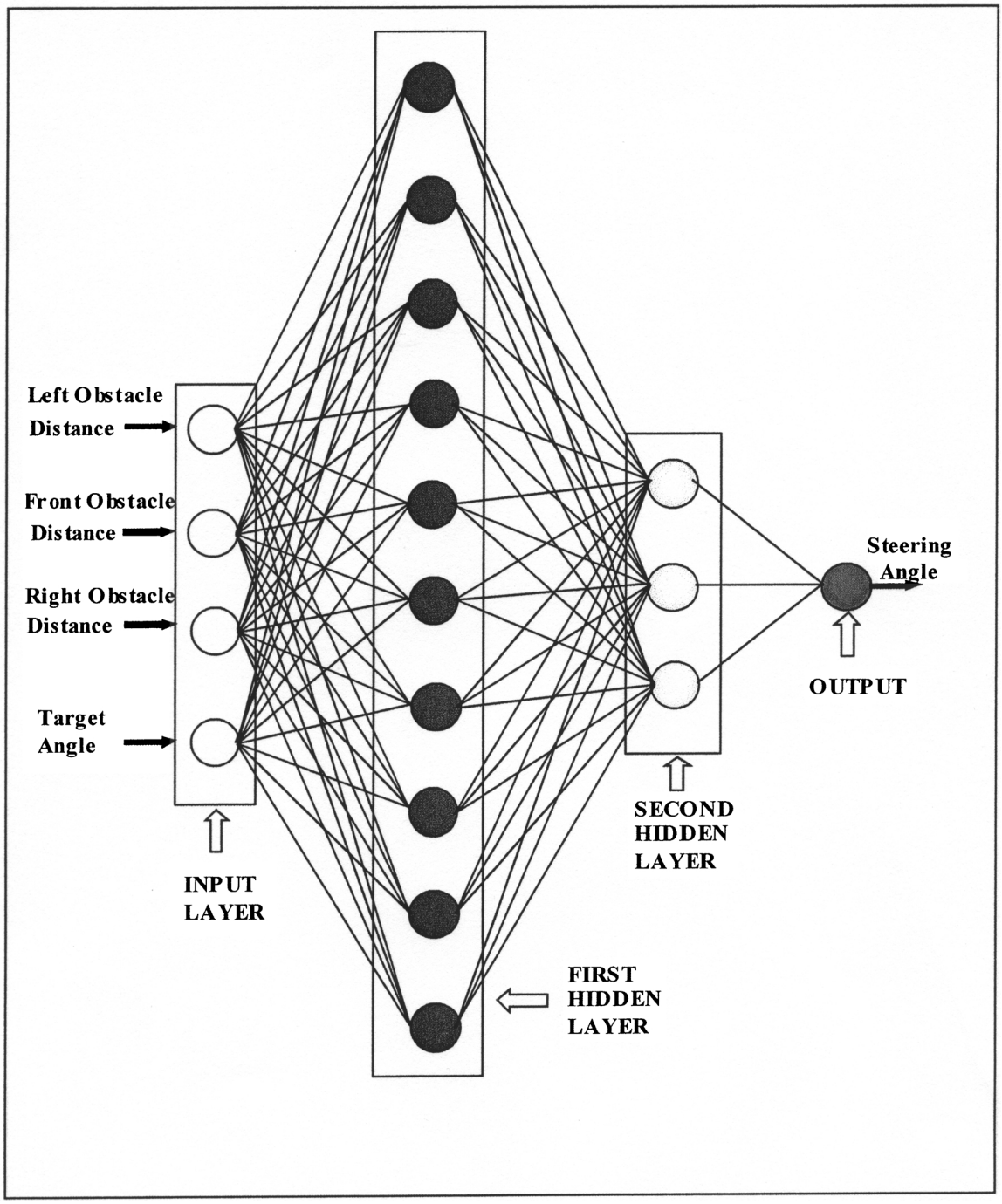


Fig. 1. Four-layer neural network for robot navigation.

During training and during normal operation, the input patterns fed to the neural network comprise the following components:

$y_1^{\{1\}}$ = Left obstacle distance from the robot (1a)

$y_2^{\{1\}}$ = Front obstacle distance from the robot (1b)

$y_3^{\{1\}}$ = Right obstacle distance from the robot (1c)

$y_4^{\{1\}}$ = Target bearing (1d)

These input values are distributed to the hidden neurons which generate outputs given by [11]:

$$y_j^{\{lay\}} = f(V_j^{\{lay\}}) \quad (2)$$

where

$$V_j^{\{lay\}} = \sum_i W_{ji}^{\{lay\}} \cdot y_i^{\{lay-1\}} \quad (3)$$

lay = layer number (2 or 3)

j = label for j^{th} neuron in hidden layer 'lay'

i = label for i^{th} neuron in hidden layer 'lay − 1'

$W_{ji}^{\{lay\}}$ = weight of the connection from neuron i in layer 'lay − 1' to neuron j in layer 'lay'

https://doi.org/10.1017/S0263574702004526 Published online by Cambridge University Press

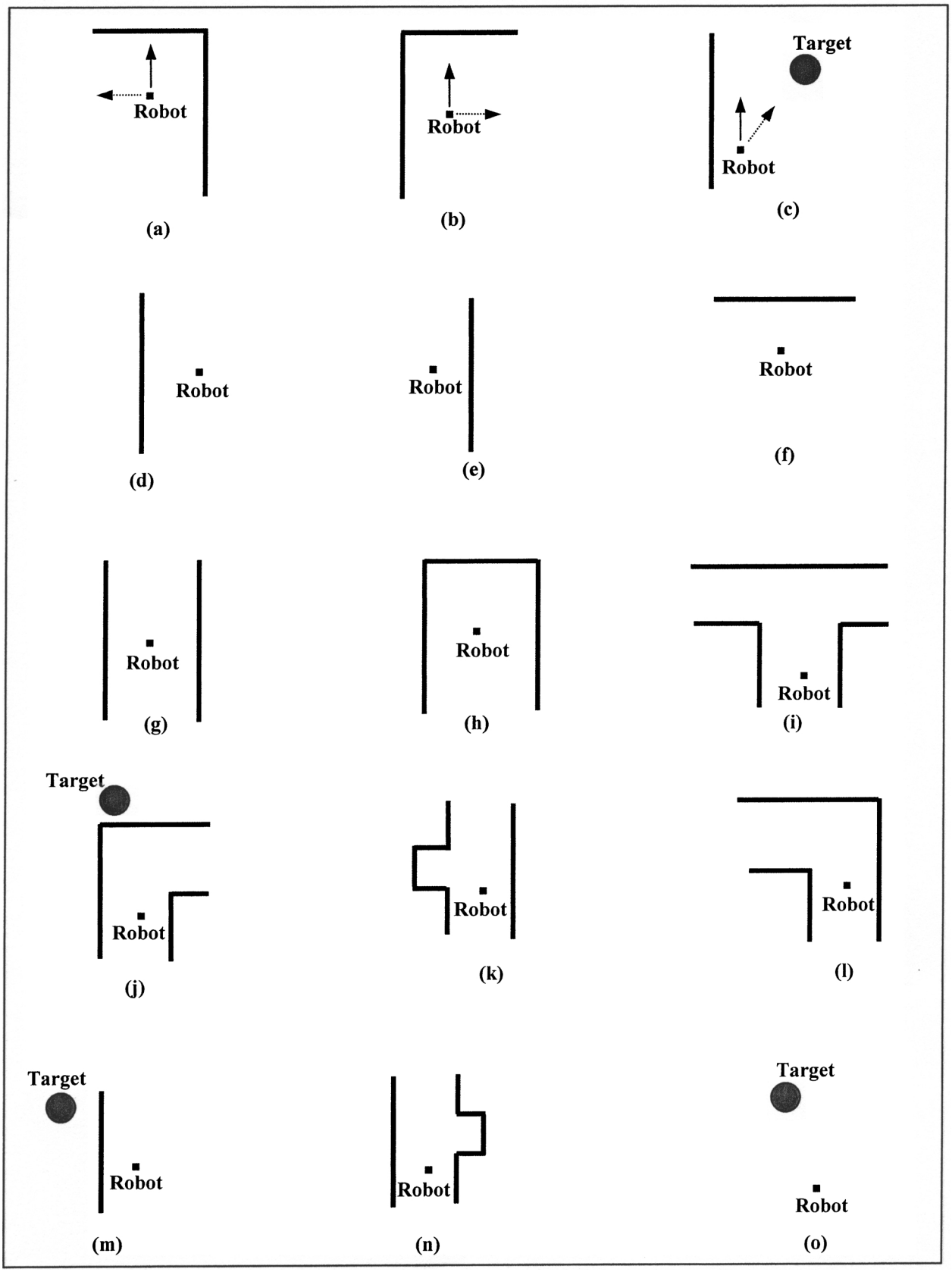


Fig. 2. Example training patterns.

f(.) = activation function, chosen in this work as the hyperbolic tangent function:

$$f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{4}$$

During training, the network output θ_{actual} may differ from the desired output $\theta_{desired}$ as specified in the training pattern presented to the network. A measure of the performance of the network is the instantaneous sum-squared difference between $\theta_{desired}$ and θ_{actual} for the set of presented training

https://doi.org/10.1017/S0263574702004526 Published online by Cambridge University Press

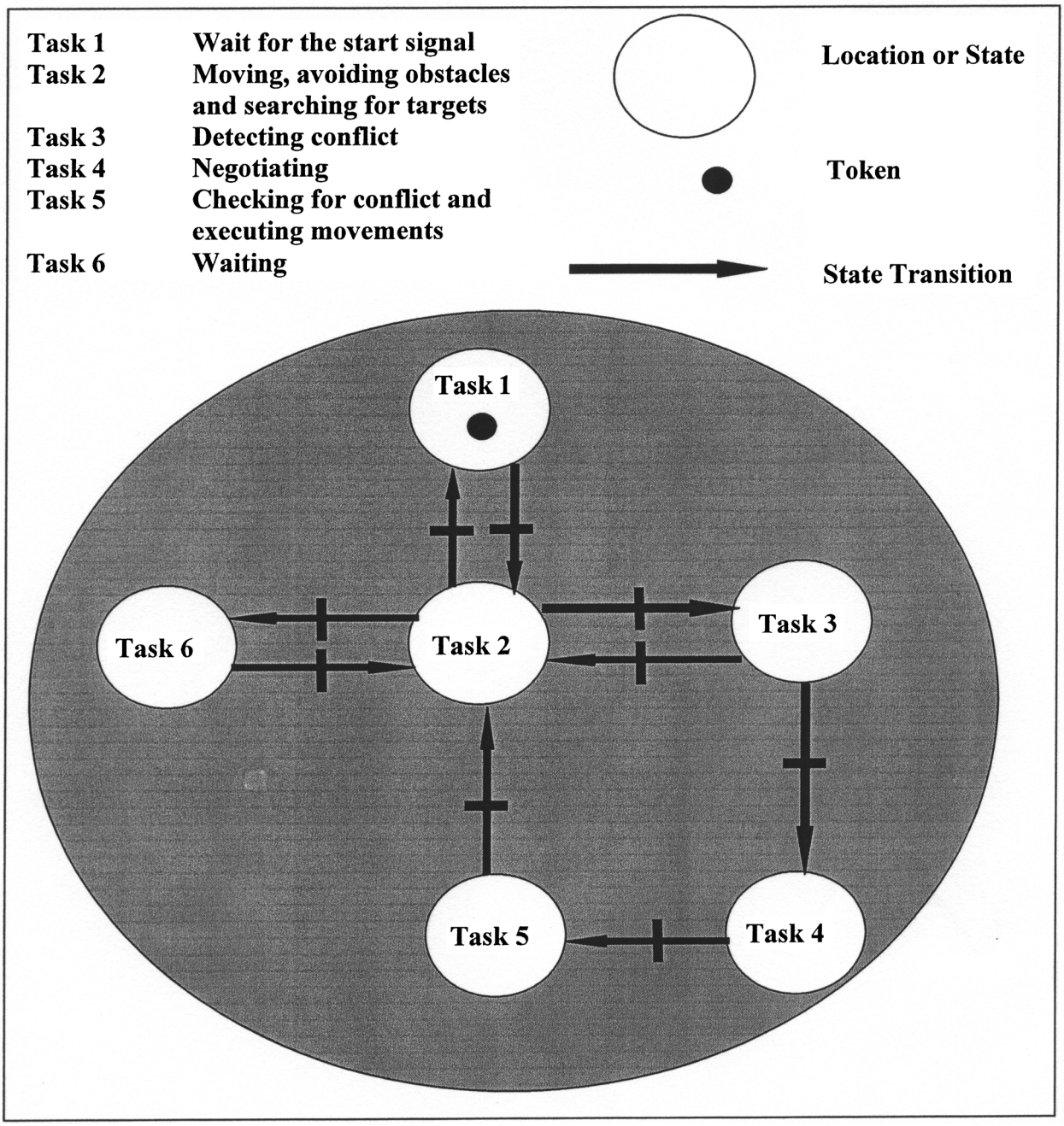


Fig. 3. Petri Net model for avoiding inter robot collision.

patterns:

$$\mathrm{Err}=\frac{1}{2}\sum_{\substack{\text{all training}\\ \text{patterns}}}(\theta_{\text{desired}}-\theta_{\text{actual}})^2 \tag{5}$$

The error back propagation method is employed to train the network.[11] This method requires the computation of local error gradients in order to determine appropriate weight corrections to reduce Err. For the output layer, the error gradient $\delta^{\{4\}}$ is:

$$\delta^{\{4\}}=f'(V_j^{\{4\}})(\theta_{\text{desired}}-\theta_{\text{actual}}) \tag{6}$$

The local gradient for neurons in hidden layer {lay} is given by:

$$\delta_j^{\{\text{lay}\}}=f'(V_j^{\{\text{lay}\}})\left(\sum_k \delta_k^{\{\text{lay}+1\}}W_{kj}^{\{\text{lay}+1\}}\right) \tag{7}$$

The synaptic weights are updated according to the following expressions:

$$W_{ji}(t+1)=W_{ji}(t)+\Delta W_{ji}(t+1) \tag{8}$$

and

$$\Delta W_{ji}(t+1)=\alpha\Delta W_{ji}(t)+\eta\delta_j^{\{\text{lay}\}}y_i^{\{\text{lay}-1\}} \tag{9}$$

https://doi.org/10.1017/S0263574702004526 Published online by Cambridge University Press

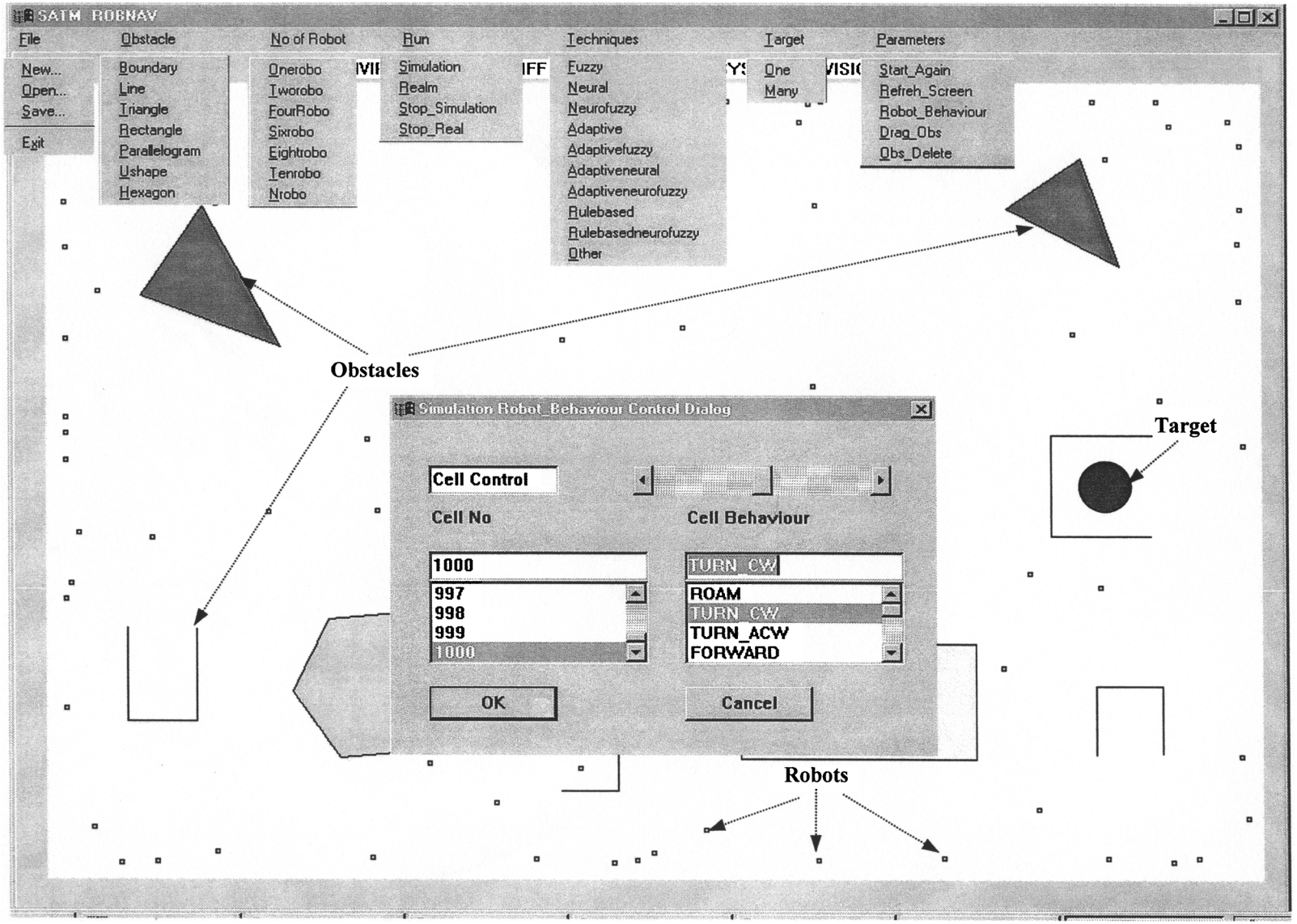


Fig. 4. Robot navigation software package.

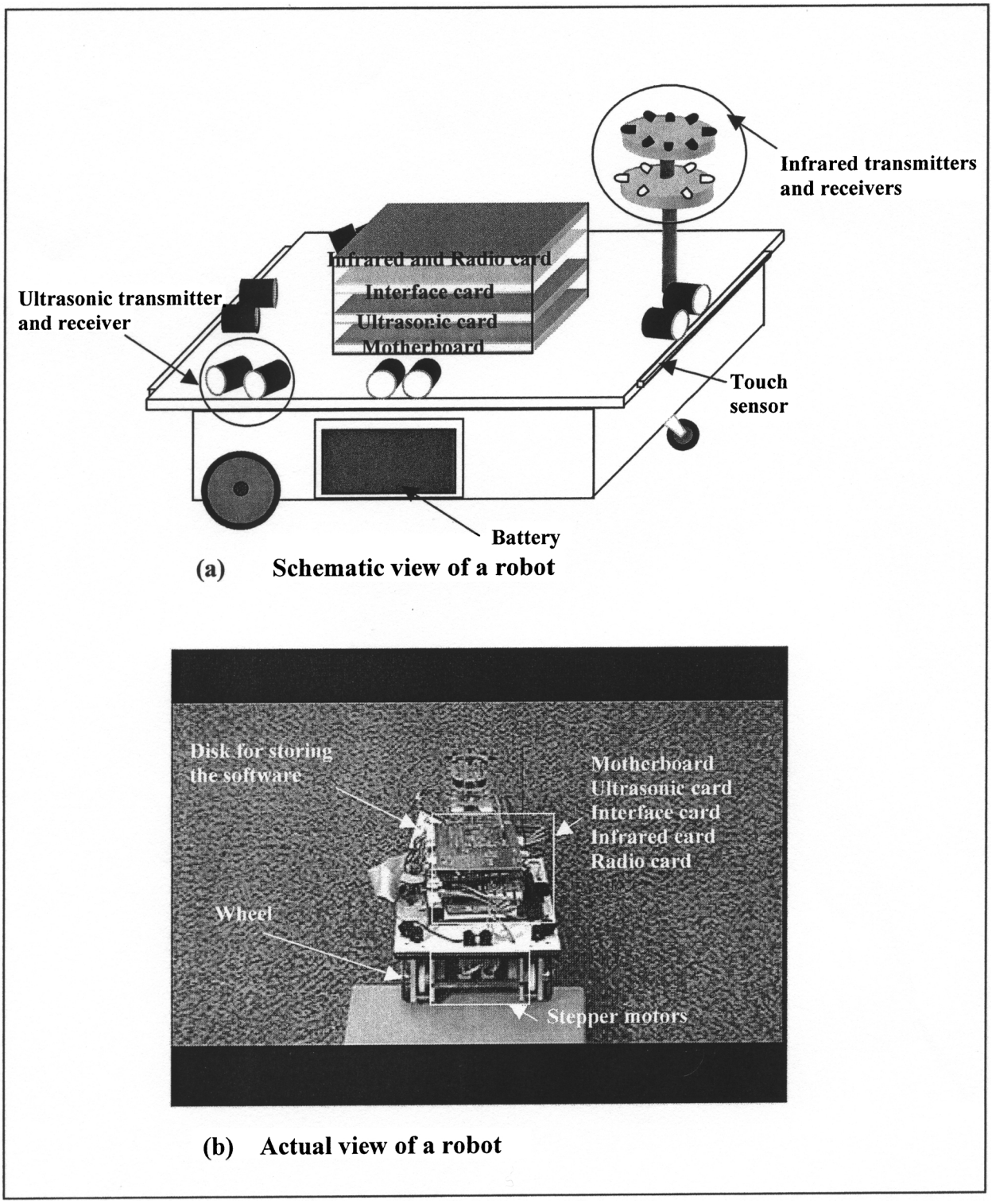


Fig. 5. A mobile robot.

where

a = momentum coefficient (chosen empirically as 0.2 in this work)

η = learning rate (chosen empirically as 0.35 in this work)

t = iteration number, each iteration consisting of the presentation of a training pattern and correction of the weights.

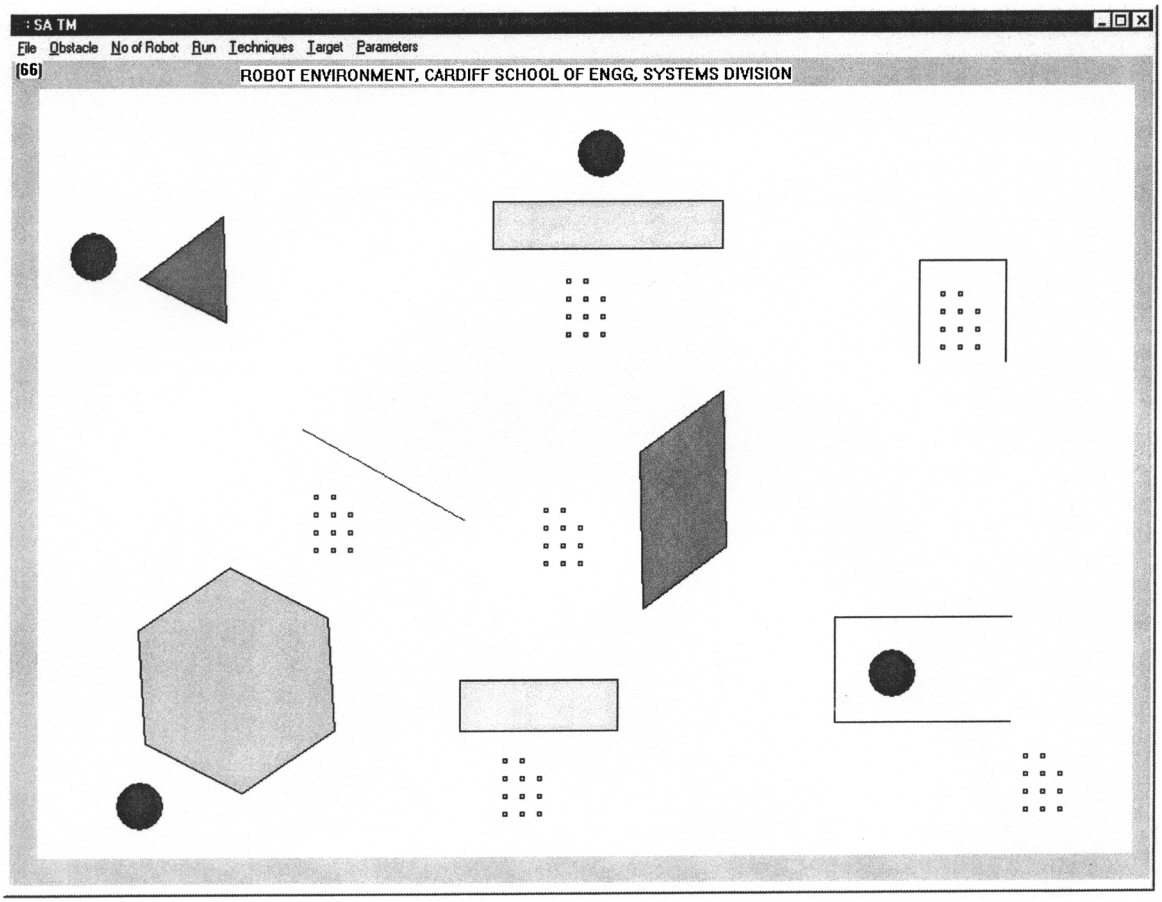
The final output from the neural network is:

$$\theta_{actual} = f(V_l^{\{4\}}) \tag{10}$$

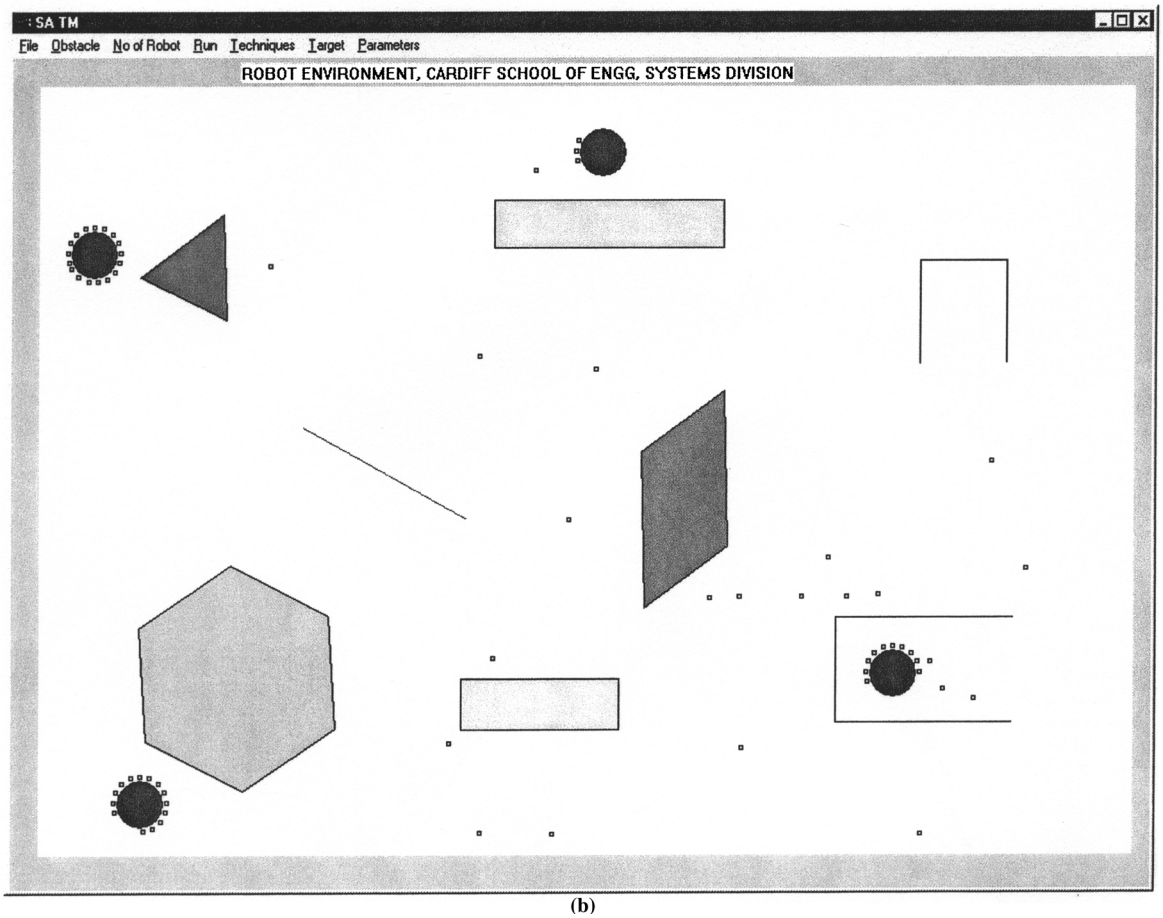
where

$$V_l^{\{4\}} = \sum_i W_{li}^{\{4\}} y_i^{\{3\}} \tag{11}$$

https://doi.org/10.1017/S0263574702004526 Published online by Cambridge University Press



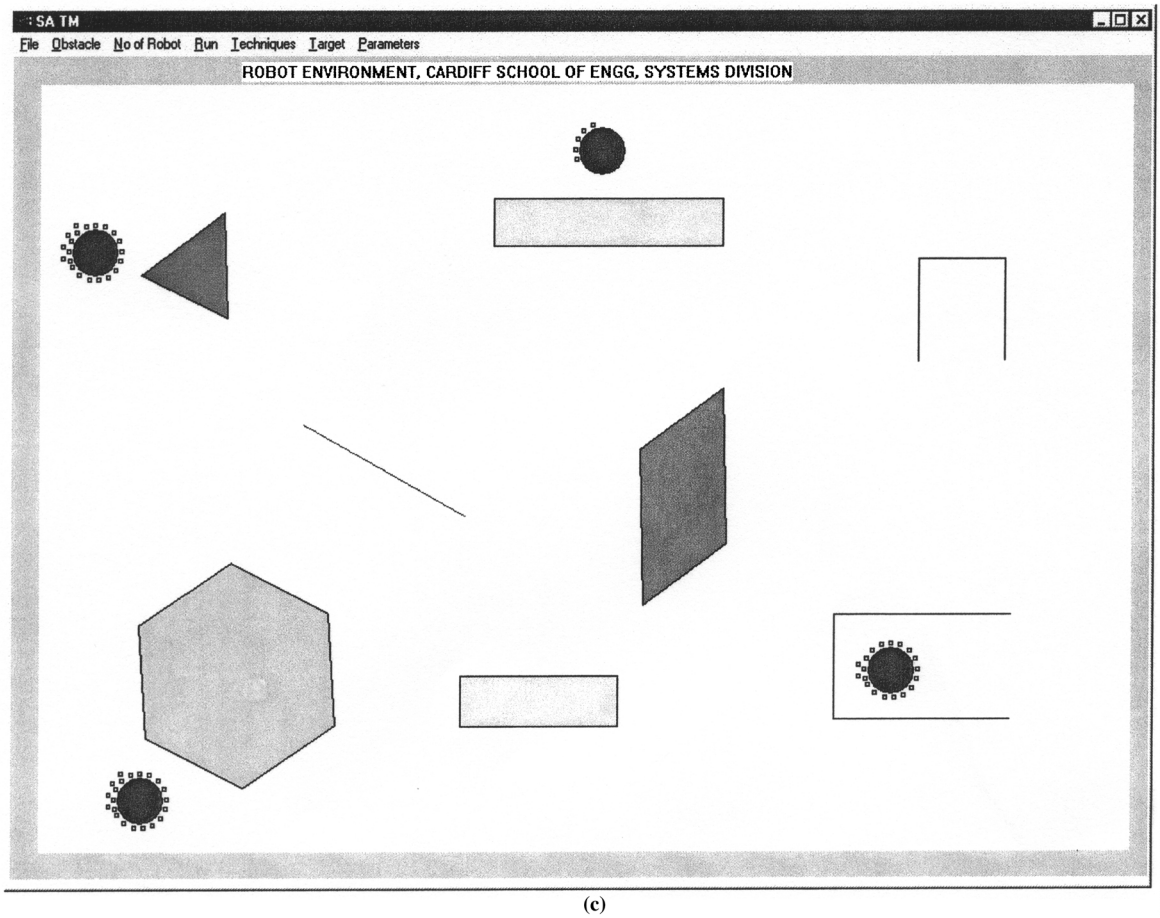
(a)



(b)

Fig. 6. (a) Target seeking by 66 mobile robots (initial state); (b) Target seeking by 66 mobile robots (intermediate state).

https://doi.org/10.1017/S0263574702004526 Published online by Cambridge University Press



(c)

Fig. 6. (c) Target seeking by 66 mobile robots (final state).

2.2. Inter robot collision avoidance

Petri Net modelling was developed by C. A. Petri[12] as a means of representing the behaviour of a dynamical system. Figure 3 depicts the Petri Net model built into each robot to enable it to avoid collision with other robots. The model comprises 6 states (or Tasks). The location of the token indicates the current state of the robot.

It is assumed that, initially, the robots are in a cluttered environment, without any prior knowledge of one another or of the targets and obstacles. This means each robot is in state "Task 1" ("Wait for the start signal") or the token is at location "Task 1" (see Figure 3).

Once the robots have received a command to start searching for the targets, they will try to locate them while avoiding obstacles and one another. The robots are thus in state "Task 2" ("Moving, avoiding obstacles and searching for targets").

During navigation, if the path of a robot is obstructed by another robot, then a conflict situation is detected (state "Task 3": "Detecting Conflict"). Two robots in conflict will negotiate with each other to decide which one has priority. The lower-priority robot will be treated as a static obstacle and the higher-priority robot as a proper mobile system (state "Task 4": "Negotiating"). As soon as a conflict is resolved, the robots will look for other conflicts and if there are none they will execute their movements (state "Task 5": "Checking for conflict and executing movements").

If a robot meets two other robots already in a conflict situation, then its priority will be lowest and it will be treated as a static obstacle (state "Task 6": "Waiting") until the conflict is resolved. When this is done, the robot will re-enter state "Task 2".

3. DEMONSTRATIONS

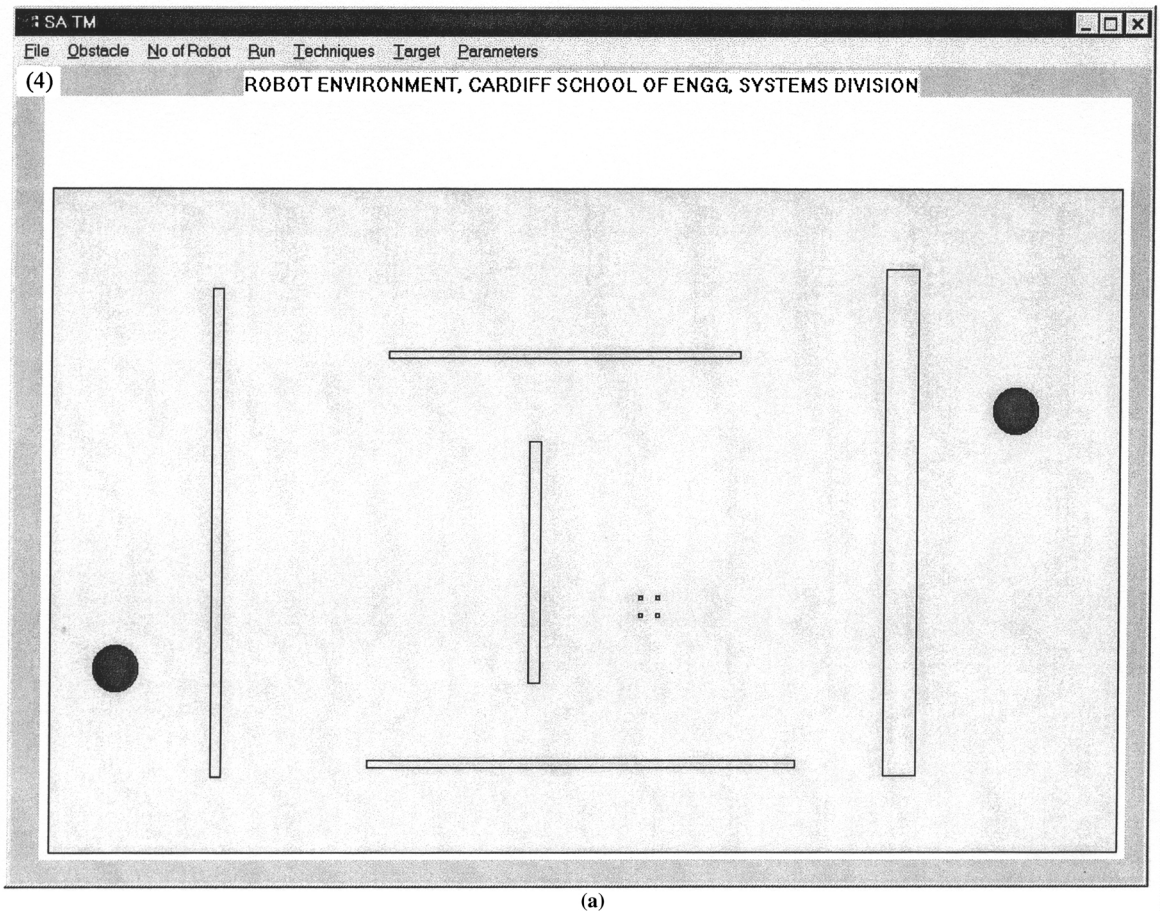
The neural-network-based navigation controller and the Petri Net obstacle avoidance model have been implemented in simulation and in a system of actual mobile robots.

The Windows-based simulation software developed enables groups of up to 1000 robots to be generated and controlled in an artificial environment containing static targets as well as static obstacles (straight walls and polygonal convex and concave objects). Figure 4 shows a typical screen produced by the software. It can be noted that, in addition to the neural-network-based navigation, the software also allows other navigation control techniques (for example, fuzzy logic and neuro-fuzzy-based techniques) to be simulated.

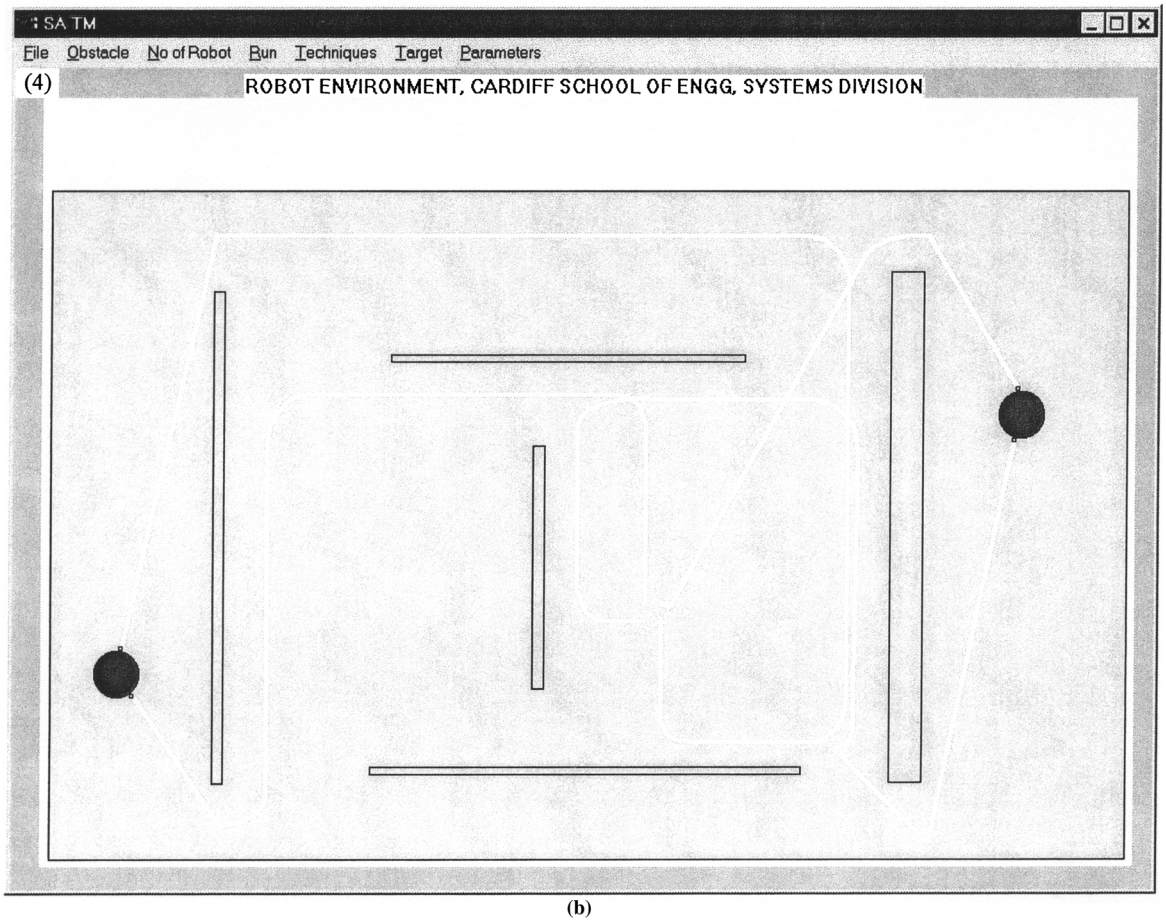
Figure 5 depicts one of the experimental mobile robots employed in this study. The robot hardware used was developed by Beutler[13] but the hardware control software was new.[14] Each mobile robot consists of:

(i) Three wheels, two of which are driven by stepper motors and the third is free.

https://doi.org/10.1017/S0263574702004526 Published online by Cambridge University Press



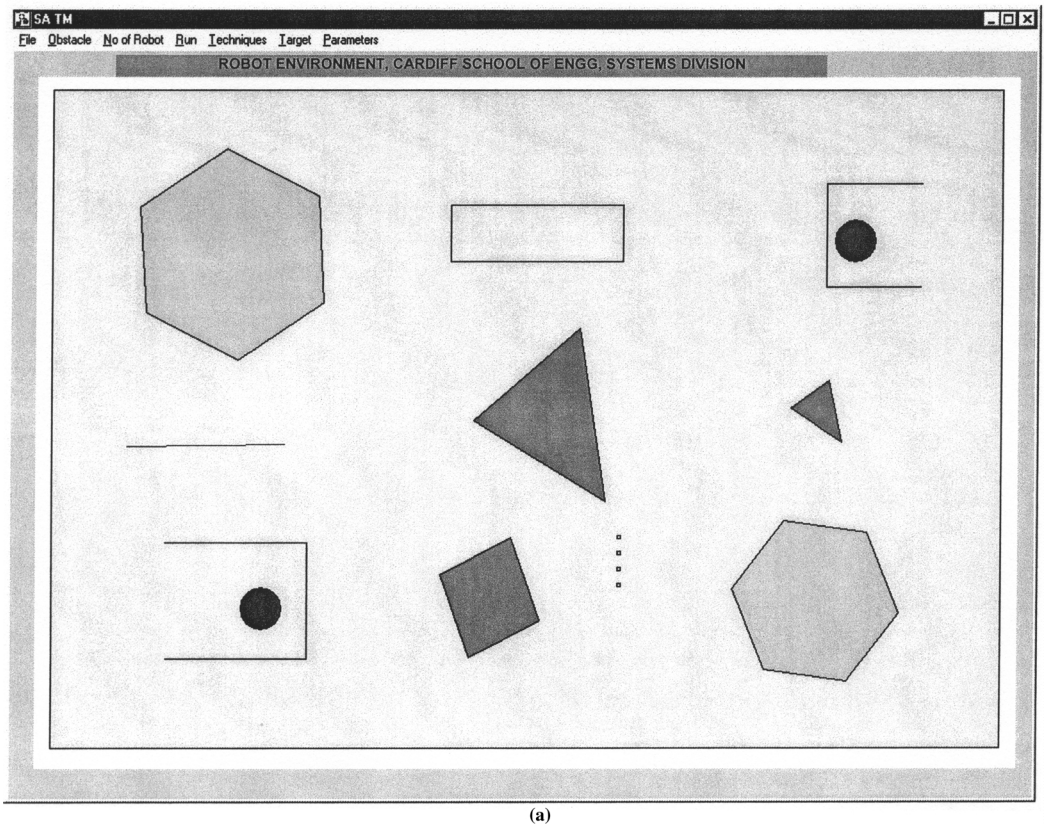
(a)



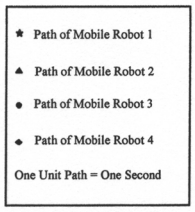
(b)

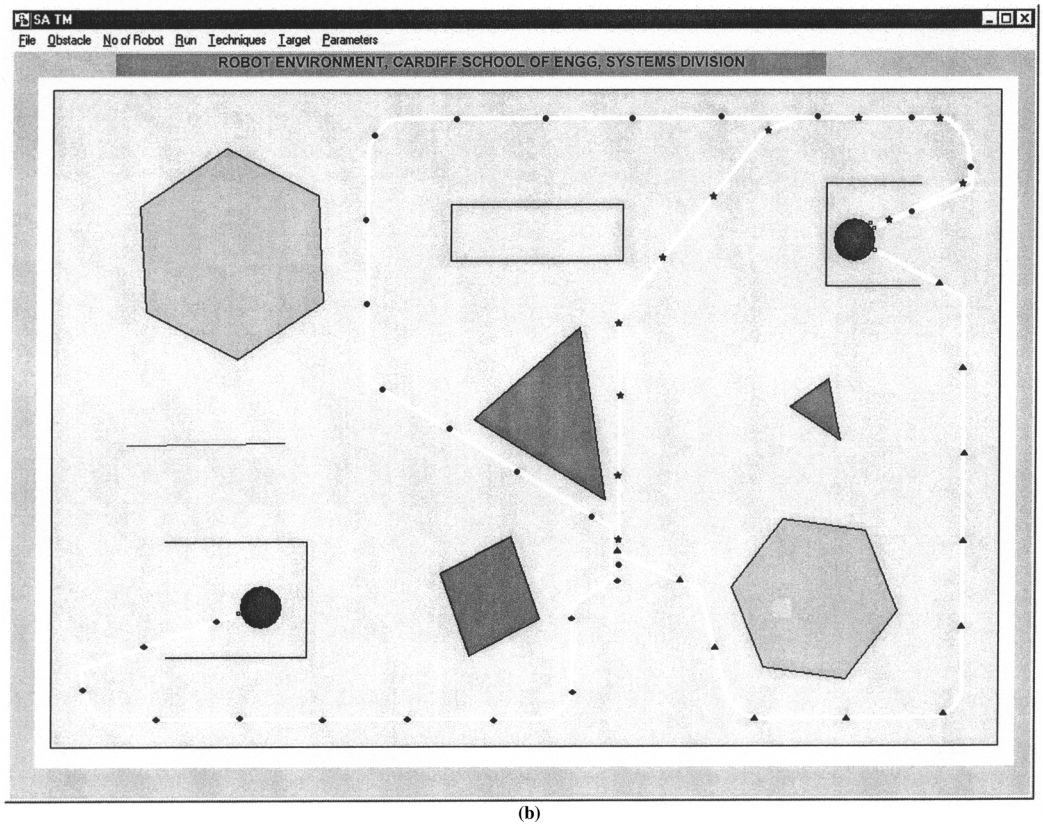
Fig. 7. (a) Wall following (initial state); (b) Wall following (final state).

https://doi.org/10.1017/S0263574702004526 Published online by Cambridge University Press



(a)

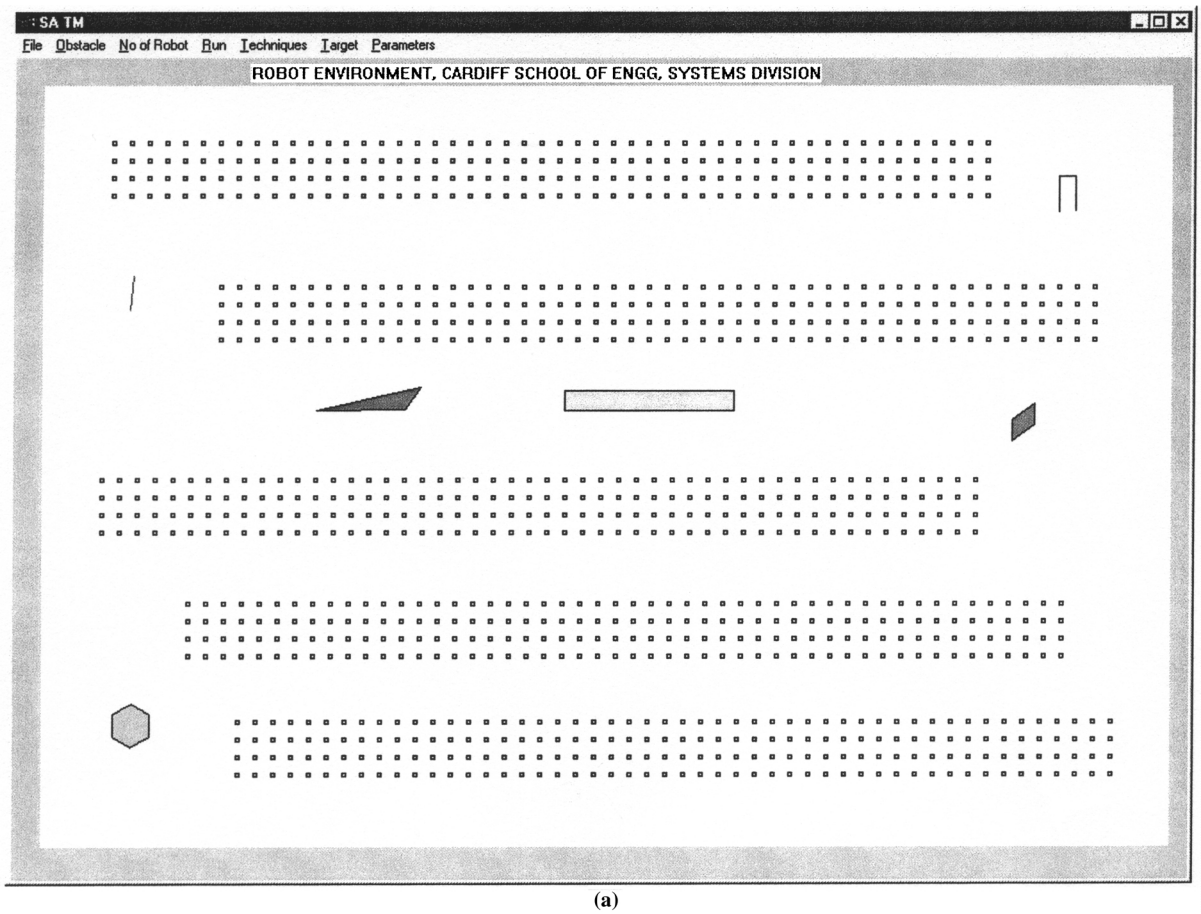




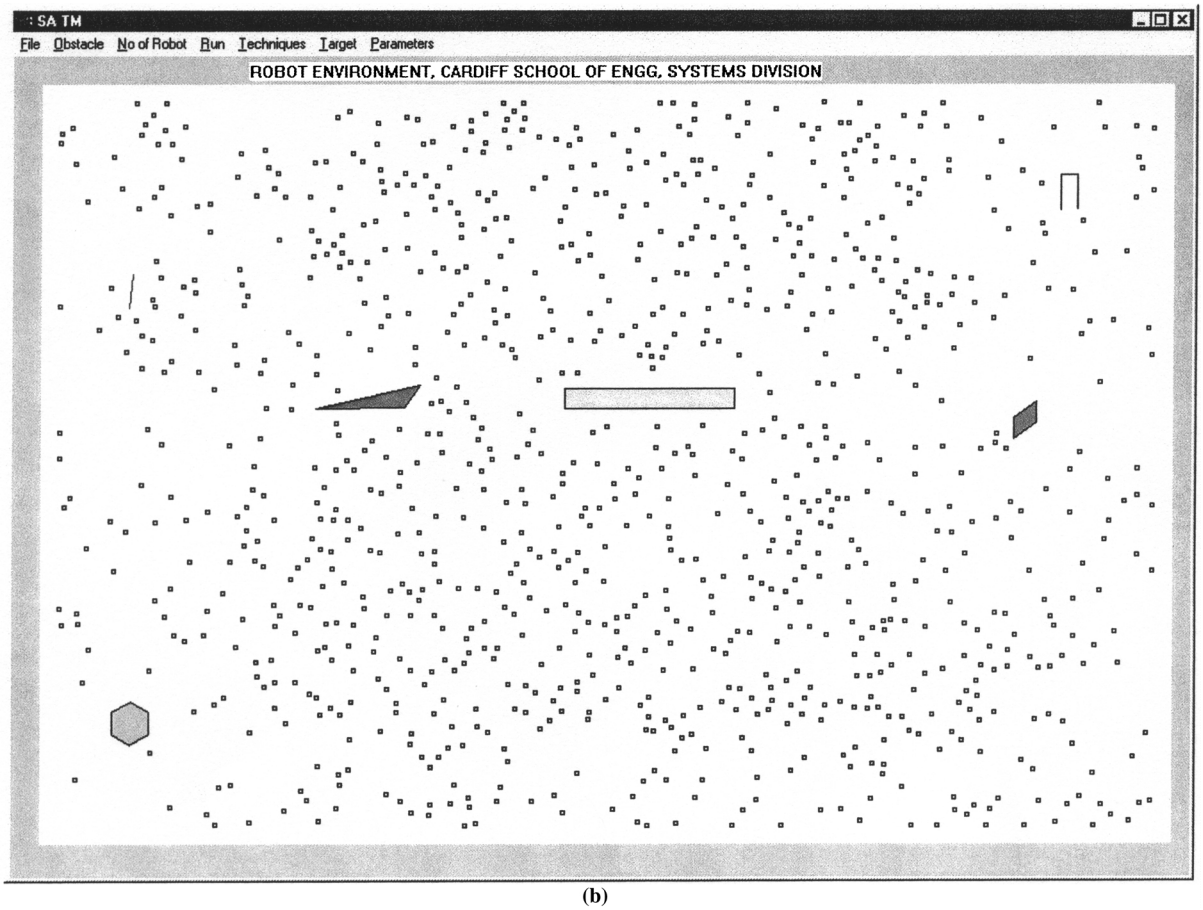
(b)

Fig. 8. (a) Collision free movements (initial state); (b) Collision free movements (neural controller).

https://doi.org/10.1017/S0263574702004526 Published online by Cambridge University Press



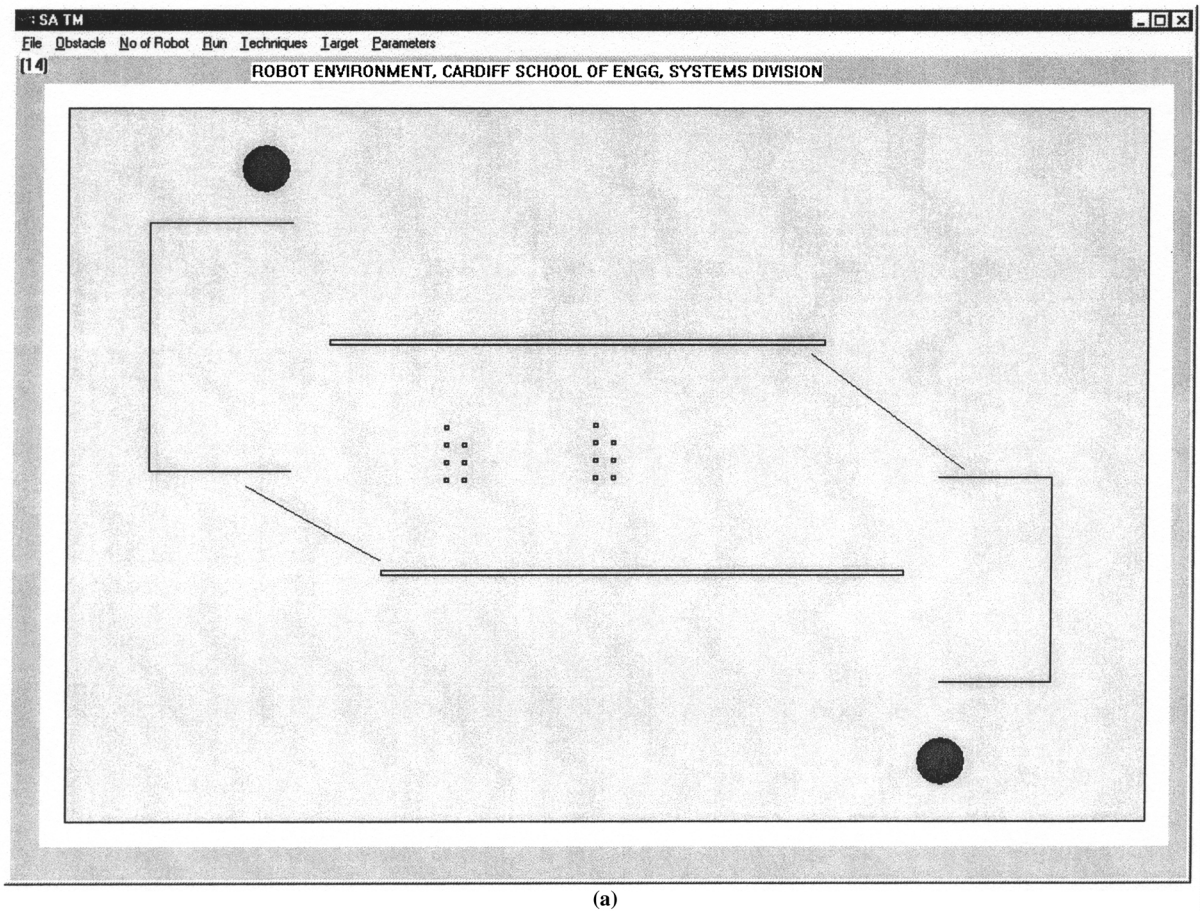
(a)



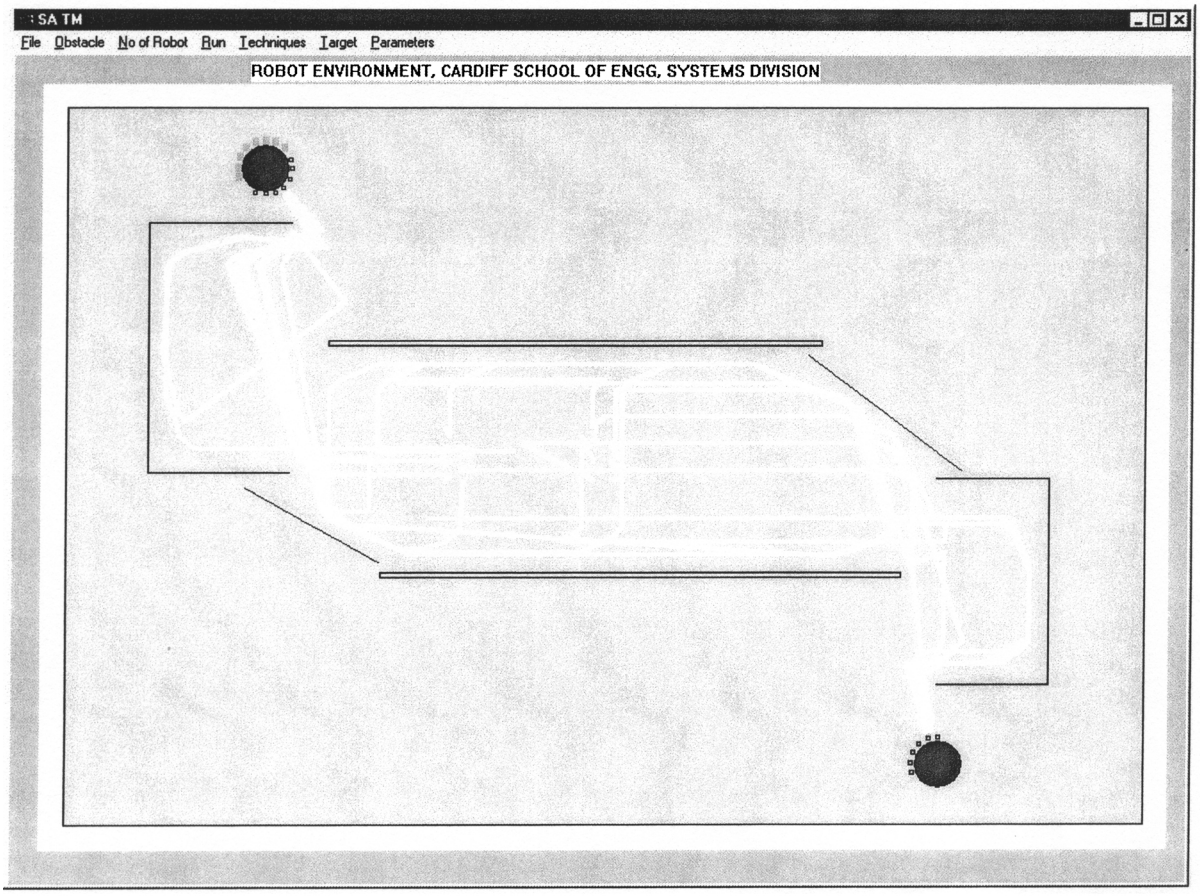
(b)

Fig. 9. (a) View of an environment consisting of one thousand mobile robots; (b) View of an environment consisting of one thousand mobile robots during navigation.

https://doi.org/10.1017/S0263574702004526 Published online by Cambridge University Press



(a)



(b)

Fig. 10. (a) Escape from dead ends (initial state); (b) Escape from dead ends (final state).

https://doi.org/10.1017/S0263574702004526 Published online by Cambridge University Press

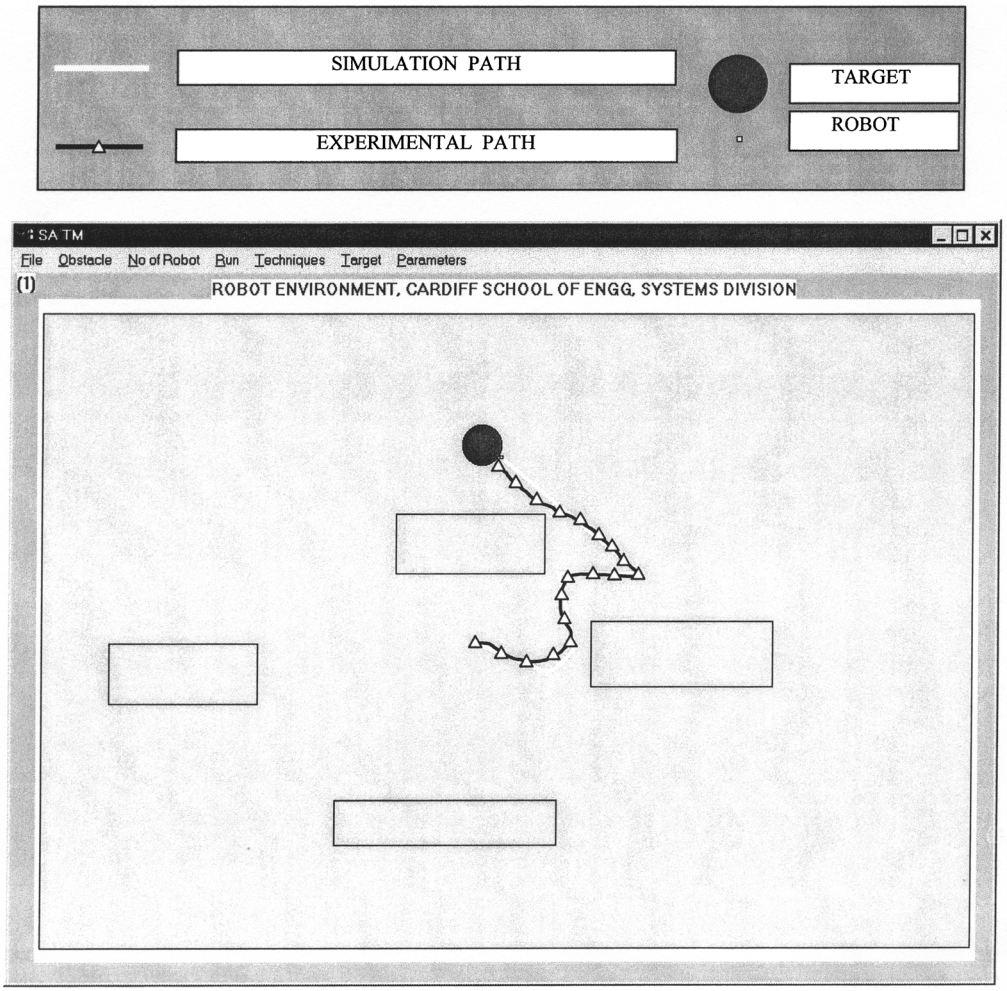


Fig. 11. (a) Experiment with one robot; (b) Experiment with four robots.

https://doi.org/10.1017/S0263574702004526 Published online by Cambridge University Press

Table I. Time taken by robots in simulation and experiment to reach targets.

Number of robots	Time during simulation (Seconds)	Time during experiment (Seconds)
1	19.23	20.39
2	24.23	25.17
4	27.44	28.95

(ii) A PC mother board (386 SX, 33 MHz, 2MB RAM).
(iii) Six ultrasonic transmitters and receivers for measuring distances from obstacles around the robot.
(iv) Six infrared transmitters and eight infrared receivers for detecting the targets.
(v) A radio modem card for communication of commands with other robots and also with other computers.
(vi) Two touch sensors, one at the front and one at the rear of the robot.

3.1. Simulation results

- **Target seeking behaviour.** This exercise demonstrates successful tracking by multiple mobile robots controlled using the proposed hybrid technique. Figure 6a illustrates the situation at the beginning of the exercise where 66 robots, grouped into six units of 11 robots each, are placed at different locations in a cluttered environment. Figure 6b shows that some targets in the environment have been found. Figure 6c depicts the final situation with all the targets located by the robots.
- **Wall following.** This exercise demonstrates the ability of the robots to follow a wall. For ease of visualisation, only four robots are employed. Figure 7a shows the state at the beginning of the exercise when the robots are located in a corridor. There are two targets to be reached both of which are hidden from the robots. Figure 7b depicts the final state as well as the trajectories of the four robots. It can be noted that both targets have been attained and that two of the robots reach them after following the walls of the corridor.
- **Inter-robot collision avoidance.** Figures 8a and 8b relate to an exercise designed to demonstrate that the robots reach their targets in a highly cluttered environment without colliding with obstacles or one another. Figure 8a depicts the beginning and Figure 8b the end of the exercise. Again, for ease of visualisation, only four robots are employed in the exercise.
- **Navigation of one-thousand mobile robots.** Obstacle avoidance by one thousand mobile robots is shown in Figures 9a and 9b. Figure 9a depicts the state at the beginning of the exercise. Figure 9b shows the situation some time after the exercise has begun. It can be noted that the robots stay well away from the obstacles.
- **Escape from dead ends.** Figures 10a and 10b show the ability of the robots to escape from dead ends. Figure 10a depicts the situation at the beginning of the exercise. Fourteen robots are involved in the exercise. Two of the obstacles are U-shaped with the bottom of each U representing a dead end. From Figure 10b, it can be seen that the robots can negotiate dead ends and find the target successfully.

3.2. Results of experiments with actual robots

Figures 11a and 11b show experimental results obtained for one and four mobile robots respectively. The positions of the robots are marked on the floor by a pen (attached to the front of the robots) as they move. The experimentally obtained paths follow closely those traced by the robots during simulation. From these figures, it is seen that the robots can indeed avoid obstacles and reach the targets. Table I shows the time taken by the robots in simulation and in the practical tests to complete their tasks of finding the targets. The results given are the averages of 100 experiments.

4. CONCLUSION

This paper has described a system for controlling the navigation of multiple mobile robots in a cluttered unknown environment. The system involves training a neural network in each robot to enable it to handle different generic types of static obstacles. Another component of the system is the use of a Petri Net model embedded in the robots to describe the behaviour required to avoid colliding among themselves.

The proposed system is simple to implement as it only needs the collecting of examples of static obstacles to train the neural network and a simple set of collision avoidance rules for the Petri Net model. The system has been tested in simulation and with actual mobile robots. The tests have shown that the system enables multiple robots to navigate and locate targets successfully in environments with several obstacles.

In addition to the system reported in this paper, work is also being conducted on different navigation techniques to determine the most appropriate paradigms for multiple mobile robot control.

ACKNOWLEDGEMENT

This work was part of the ITEE and SUPERMAN projects funded by the Welsh Assembly and the European Commission under the Objective 2 ERDF Programme for Industrial South Wales and the Objective 1 Programme for West Wales and the Valleys. The authors are grateful to the sponsors for their support.

References

1. N.J. Nilsson, *Learning Machines: foundations of trainable pattern-classifying systems* (McGraw-Hill, New York, 1965).
2. E. Sacerdoti, "Planning in a hierarchy of abstraction spaces," *Artificial Intelligence* **5**, No. 2, 115–135 (1974).
3. T. Lozano-Perez and M.A. Wesley, "An algorithm for planning collision free paths among polyhedral obstacles," **ACM22**, No. 10, 560–570 (1979).
4. R.A. Brooks, "A robust layered control system for a mobile robot," *IEEE Transactions on Robotics and Automation* **2**(1), 14–23 (1986).

https://doi.org/10.1017/S0263574702004526 Published online by Cambridge University Press

5. J.L. Crowley and J. Coutaj, "Navigation et modelisation pour un robot mobile," *Technique et Science Informatiques* **5**(5), 391–402 (1986).
6. R.C. Luoc and M.G. Kay, "Multisensor integration and fusion in intelligent system," *IEEE Transactions on Systems, Man and Cybernetics* **19**, No. 5, 901–931 (1989).
7. M. Benreguieg, P. Hoppenot, H. Maaref, E. Colle and C. Barret, "Fuzzy navigation strategy: Application to two distinct autonomous mobile robots" *Robotica* **15**, Part 6, 609–615 (1997).
8. P.K. Pal and A. Kar, "Sonar-based mobile robot navigation through supervised learning on a neural-net," *Autonomous Robots* **3**, No. 4, 355–374 (1996).
9. G.W. Wang, N. Fuziwara and Y. Bao, "Feed-forward multi-layer neural network model for vehicle lateral guidance control," *Advanced Robotics* **12**, No. 7-8, 735–753 (1999).
10. J.A. Janet, R. Gutierrez, T.A. Chase, M.W. White and J.C. Sulton, "Autonomous mobile robot global self-localization using Kohonen and region-feature neural network," *Journal of Robotic Systems* **14**, No. 4, 263–282 (1997).
11. D.T. Pham and X. Liu, *Neural Networks for Identification, Prediction and Control*. 4th Printing (Springer Verlag, London and Heidelberg, 1999).
12. J.L. Peterson, *Petri Net theory and the Modelling of Systems* (Prentice-Hall, Englewood Cliff, N.J., 1981).
13. J. Beutler, "A novel framework for creating agent societies," *Ph.D. Thesis* (Cardiff School of Engineering, University of Wales, UK, 1999).
14. D.R. Parhi, "Navigation of multiple mobile robots in an unknown environment," *Ph.D. Thesis* (Cardiff School of Engineering, University of Wales, UK, 2000).

https://doi.org/10.1017/S0263574702004526 Published online by Cambridge University Press