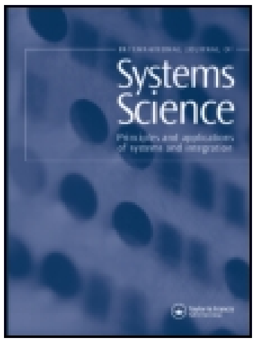


International Journal of Systems Science

ISSN: 0020-7721 (Print) 1464-5319 (Online) Journal homepage: https://www.tandfonline.com/loi/tsys20

Path optimisation of a mobile robot using an artificial neural network controller

M.K. Singh & D.R. Parhi

International Journal of Systems Science
Vol. 42, No. 1, January 2011, 107–120

Path optimisation of a mobile robot using an artificial neural network controller

M.K. Singh[a]* and D.R. Parhi[b]

[a]*Department of Mechanical Engineering, Government Engineering College, Bilaspur, Chhattisgarh 495009, India;* [b]*Department of Mechanical Engineering, National Institute of Technology, Rourkela 769008, Orissa, India*

(*Received 28 November 2008; final version received 29 October 2009*)

This article proposed a novel approach for design of an intelligent controller for an autonomous mobile robot using a multilayer feed forward neural network, which enables the robot to navigate in a real world dynamic environment. The inputs to the proposed neural controller consist of left, right and front obstacle distance with respect to its position and target angle. The output of the neural network is steering angle. A four layer neural network has been designed to solve the path and time optimisation problem of mobile robots, which deals with the cognitive tasks such as learning, adaptation, generalisation and optimisation. A back propagation algorithm is used to train the network. This article also analyses the kinematic design of mobile robots for dynamic movements. The simulation results are compared with experimental results, which are satisfactory and show very good agreement. The training of the neural nets and the control performance analysis has been done in a real experimental setup.

Keywords: artificial neural network; mobile robot; intelligent control; path optimisation

1. Introduction

A lot of research is going on around the globe to find a suitable intelligent control to be used for navigation of a mobile robot without human interaction. One of the most important issues in the design and development of an intelligent controller for a mobile robot is the navigation problem, i.e. the sequences of actions required during goal-achieving without collision with static as well as dynamic obstacles. This consists of the abilities of a mobile robot to plan and execute collision-free motions within its environment. However, this environment may be imprecise, vast, dynamical and either partially or non-structured. Robots must be able to understand the structure of this environment (Nelson, Grant, and Henderson 2004; Gregor and Igor 2007; Wolf and Sukhatme 2008; Parhi, Pradhan, Panda, and Behra 2009). To reach their targets without collisions, robots must be endowed with perception, data processing, recognition, learning, reasoning, interpreting, and decision-making and action capacities. Bio-inspired robotics, in this context, tries to give an answer to issues such as mimicking the behaviours and structure of living creatures in the controller of the mobile robot.

The bio-inspired neural controller is based on the working principal of the nervous system of simple animals like arthropods or invertebrates and can be used to control a service mobile robot. Service robotics today requires the synthesis of robust automatic systems able to cope with a complex and dynamic environment (Folgheraiter, Gini, Nava, and Mottola 2006). To demonstrate this kind of autonomy, Muñiz, Zalama, Gaudiano, and López Coronado (1995) have introduced a neural controller for a mobile robot that learns both forward and inverse odometry of a differential drive robot through an unsupervised learning by doing cycle. They introduce an obstacle avoidance module that is integrated into the neural controller. However, generally, the evolved neural controllers could be fragile in unexperienced environments, especially in real worlds, because the evolutionary optimisation processes would be executed in idealised simulators. This is known as the gap problem between the simulated and real worlds. To overcome this, Kondo (2007) has focused on evolving an online learning ability instead of weight parameters in a simulated environment. Based on this, a neuromodulatory neural network model was proposed by him and is utilised as a mobile robot controller. Nolfi and Tani (1999) have investigated mobile robot navigation in a structured environment by extracting regularities from time series through prediction learning which are hidden in the sensory level. Racz and Dubrawski (1995) have presented a neural network-based approach to a mobile robot localisation in front of a certain local object. Corradini, Ippoliti, and Longhi (2003) have used a neural networks approach to the

*Corresponding author. Email: mukesh3003@yahoo.co.in

ISSN 0020–7721 print/ISSN 1464–5319 online
© 2011 Taylor & Francis
DOI: 10.1080/00207720903470155
http://www.informaworld.com

solution of the tracking problem for mobile robots. Yang and Meng (2000) have proposed a biologically inspired neural network approach to real-time collision-free motion planning of mobile robots or robot manipulators in a non-stationary environment. Braganza, Dawson, Walker, and Nath (2007) have described a controller for continuum robots which utilises a neural network feed-forward component to compensate for dynamic uncertainties. Sgouros (2001) has described a novel qualitative navigation method for mobile robots in indoor environments. Oriolo, Panzieri, and Ulivi (2000) have proposed a different approach which does not rely on a separation between planning and control. Research in autonomous multi-robot systems often focuses on mechanisms to enhance the efficiency of the group through some form of cooperation among the individual agents. Moreover, the versatility of a multi-robot system can provide the heterogeneity of structures and functions required to undertake different missions in unknown environmental conditions (Nelson et al. 2004; Gregor and Igor 2007; Parhi 2000; Pallottino, Scordio, Bicchi, and Frazzoli 2007).

This article proposes a biologically inspired neural network approach for real-time collision-free motion planning of mobile robots in a real world dynamic environment. The first requirements of the mobile robot is tracking and reaching a given target by avoiding static as well as dynamic obstacles. In addition to avoid collision, the other requirements are smoother motion, shorter travelling time and optimum clearance from the obstacles. Therefore, the path analysis and planning involves optimisation with respect to certain performance measures. This is the main research area being addressed in this article. The path and time optimisation of a mobile robot depends on the intelligence of the controller. Many researchers have used various methods to optimise the path and time. Their proposed methods are complicated and they may be good for local path planning but not for global path planning. The developed method is simple and the obtained results depict that the method is efficient and effective for navigation of a mobile robot in a dynamic cluttered environment. Four layer perceptron neural networks have been used to design an intelligent controller. The first layer is used as an input layer which directly reads signals from the arrays of sensors. The input of the network is front obstacle distance, right obstacle distance, left obstacle distance and target angle. The neural network consists of two hidden layers which adjust the weight of neurons. The output layer provides the steering angle of the robot. A back propagation algorithm is used to minimise the error and optimise the path and time of the mobile robot to reach the target.

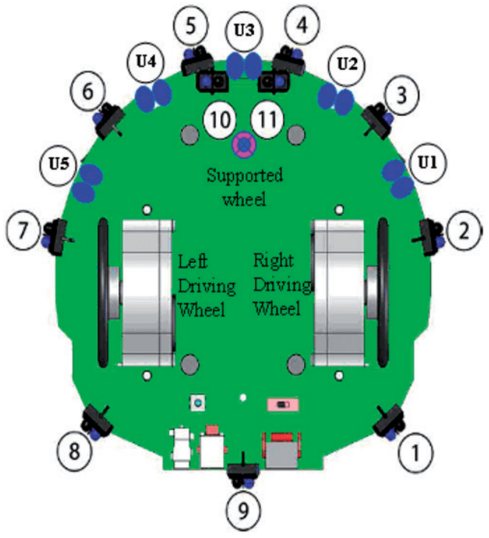


Figure 1. The chassis of the KHEPERA-III robot, 1 to 11 – infrared sensors, U1 to U5 – Ultrasonic sensors.

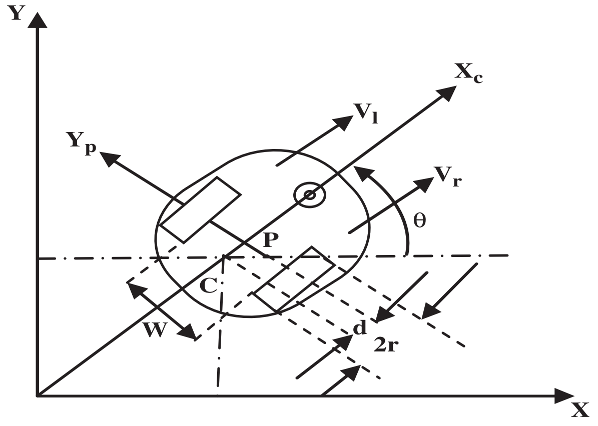


Figure 2. Kinematic analysis of the mobile robot.

The outline of this article is as follows. Following the Introduction, the kinematics analysis of the mobile robot is described in Section 2. Neural network architecture for navigation of the mobile robot is presented in Section 3. The simulation results are discussed and, to prove feasibility of the developed methodology, a comparison is made with other methods (Hamel, Soueres, and Meizel 2001; Sanchis, Isasi, Molina, and Segovia 2001; Pradhan, Parhi, Panda, and Behra 2009) in Section 4. In Section 5, experimental results are verified with simulation to demonstrate the superiority of the proposed methodology. Finally, conclusions are discussed in Section 6.

2. Kinematics analysis of mobile robot

Kinematics is the most basic study of how mechanical systems behave. It plays a greater role in following a

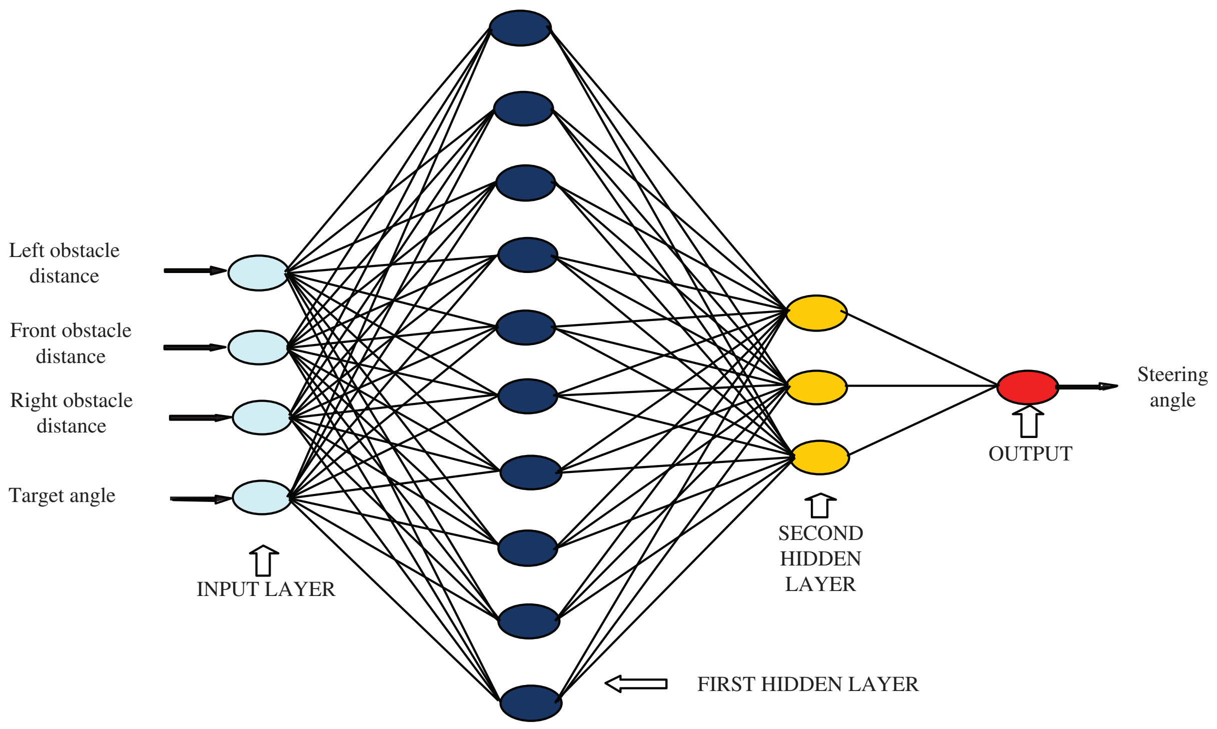


Figure 3. Four-layer neural network for robot navigation.

desired trajectory. In mobile robotics, it is necessary to understand the mechanical behaviour of the robot both in order to design appropriate mobile robots for tasks and to understand how to create control software for mobile robot hardware. The non-holonomic mobile robot is defined by Tchon and Jakubiak (2006) and its motion is subject to non-integrable velocity constraints. The kinematics model of KHEPERA-III mobile robots is shown in Figure 1. It consists of a vehicle chassis with two driving wheels mounted on the same axis and a front point sliding support. The two driving wheels are independently driven by two actuators to achieve motion and orientation. Both wheels have the same diameter, denoted by '2*r*' (Figure 2). The two driving wheels are separated by distance '*W*'. The center of gravity (COG) of the mobile robot is located at point '*C*'. The point '*P*' is located in the intersection of a straight line passing through the middle of the vehicle and a line passing through the axis of the two wheels. The distance between points *P* and *C* is '*d*'. A motion controller based on neural network techniques is proposed for navigation of the mobile robot. The main component in the motion controller is the low level inverse neural controller, which controls the dynamics of the mobile robot. The kinematics of the differential drive mobile robot is based on the assumption of pure rolling, and there is no slip between the wheel and surface.

$$v_t = \frac{1}{2}[v_r + v_l] \tag{1}$$

$$\omega_t = \frac{1}{w}[v_r - v_l] \tag{2}$$

$$v_r = r\omega_r \quad \text{and} \quad v_l = r\omega_l \tag{3}$$

where, v = linear velocity and ω = angular velocity of the mobile robot. Suffix r, l and t stand for right, left wheel and tangential (with respect to its centre of gravity point '*C*' measured in a right wheel), respectively.

The position of the robot in the global coordinate frame (O X Y) is represented by the vector notation as,

$$q = [\,x_c \quad y_p \quad \theta\,]^T \tag{4}$$

where x_c and y_p are the coordinates of the point P in the global coordinate frame (Figure 2). The variable θ is the orientation of the local coordination of the local coordinate frame ($P \quad x_c \quad y_p$) attached on the robot platform measured from the horizontal axis. Three generalised coordinates can describe the configuration of the robot as Equation (4).

The mobile robot system considered here is a rigid body and the wheels are pure rolling with no slippage. This states that the robot can only move in the direction normal to the axis of the driving wheels.

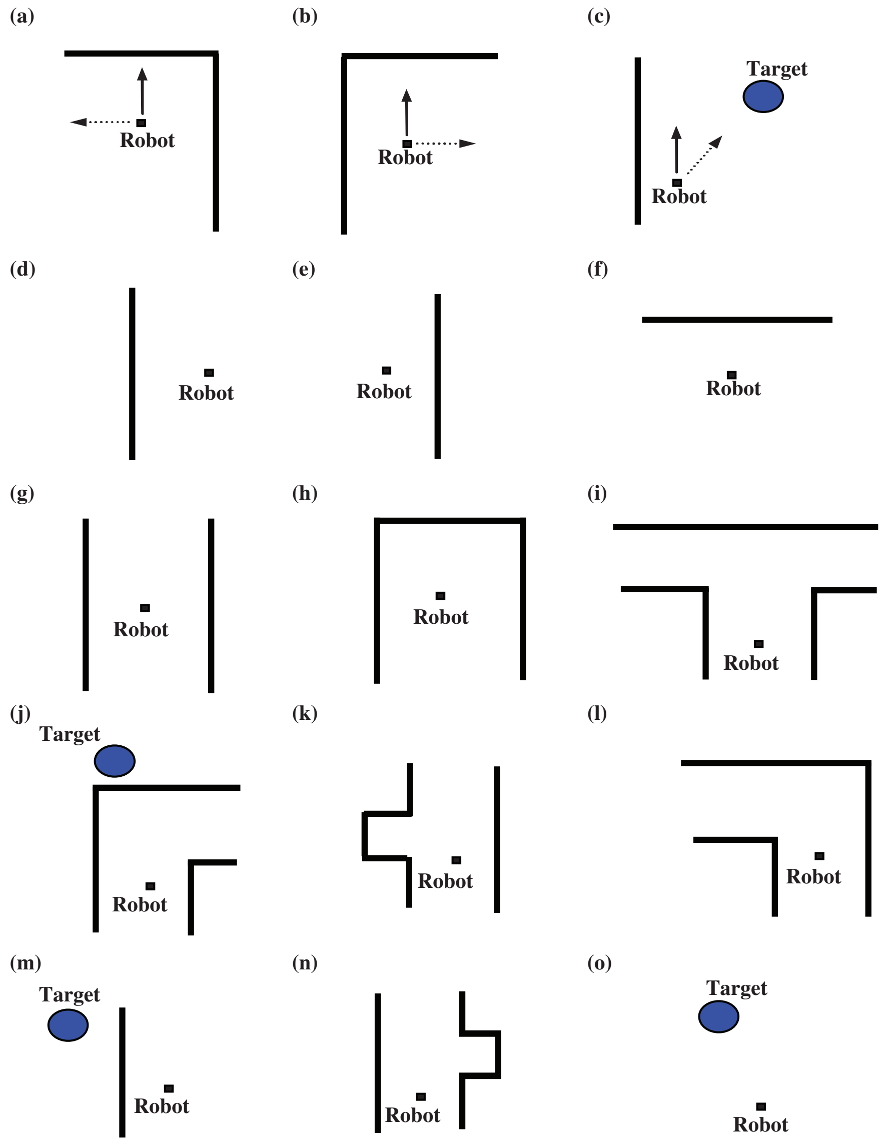


Figure 4. Example of training patterns.

Therefore, the component of the velocity of the contact point with the ground, orthogonal to the plane of the wheel, is zero (Zhang, Shippen, and Jones 1999; Das and Kar 2006), i.e.

$$[\dot{y}_p \cos\theta - \dot{x}_c \sin\theta - d\dot{\theta}] = 0 \qquad (5)$$

All kinematics constraints are independent of time, and can be expressed as

$$A^T(q)\dot{q} = 0 \qquad (6)$$

where $A(q)$ is the input transformation matrix associated with the constraints.

$$C^T A(q) = 0 \qquad (7)$$

where $C(q)$ is the full rank matrix formed by a set of smooth and linearly independent vector fields spanning the null space of $A^T(q)$.

From Equations (6) and (7) it is possible to find an auxiliary vector time function v_t for all time 't'

$$\dot{q} = C(q)v_t \qquad (8)$$

The constraint matrix in (6) for a mobile robot is given by

$$A^T(q) = [-\sin\theta \quad \cos\theta \quad -d] \qquad (9)$$

and $C(q)$ matrix is given by

$$C(q) = \begin{bmatrix} \cos\theta & -d\sin\theta \\ \sin\theta & d\cos\theta \\ 0 & 1 \end{bmatrix} \qquad (10)$$

Table 1. Some of the training pattern of the neural controller.

	Inputs of the network				Output
Position Figure 4	Front obstacle distance (cm)	Right obstacle distance (cm)	Left obstacle distance (cm)	Target angle (degrees)	Steering angle (degrees)
(a)	20	20	100	0	−90
(b)	20	100	20	0	90
(c)	100	100	10	30	30
(d)	100	100	20	0	0
(e)	100	15	100	0	0
(f)	15	100	100	0	90
(g)	100	15	15	0	0
(h)	20	15	15	0	180
(i)	50	15	15	0	0
(j)	25	10	10	−15	10
(k)	100	15	15	0	0
(l)	25	10	10	0	−10
(m)	100	100	15	−30	0
(n)	100	15	15	0	0
(o)	100	100	100	−20	−20

and

$$V_t = [v \quad \omega]^T \tag{11}$$

where V is the linear velocity of the point 'p' along the robot axis and ω is the angular velocity.

Therefore, the kinematics equation in (8) can be described as

$$\dot{q} = \begin{bmatrix} \dot{x}_c \\ \dot{y}_p \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -d\sin\theta \\ \sin\theta & d\cos\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{12}$$

Equation (12) represents the steering system of the vehicle. The control laws are designed to produce suitable left and right wheel velocities for driving the mobile robot to follow required path trajectories. The steering angle (SA) can be computed as,

$$\text{SA} = \frac{V_l - V_r}{w} \tag{12a}$$

where V_l and V_r are left and right wheel velocities and w is the wheel base. If $V_l > V_r$, the steering angle is in a clockwise direction and if $V_l < V_r$, the steering angle is in a counterclockwise direction. The control problem is to find a suitable control law so that the robot can follow the desired trajectory.

3. Analysis of neural network using back propagation algorithm

Neural networks are non-parametric estimators which can fit smooth functions based on input–output examples. The neural network used is a four-layer perceptron (Haykin 2006). 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 numbers of neurons are found to be based on the number of training patterns and the convergence of error during training to a minimum threshold error. Two hidden layers are used, as with one hidden layer there is difficulty in training the neural network within a specified error limit. 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 3 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 4. For example, Figure 4(a) shows a robot is 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 from the robot to steer towards its left. Table 1 shows an empirical training pattern consisting of some data based on Figure 4. During training and during normal operation, the input patterns fed to the neural network comprise the following components:

$$y_1^{\{1\}} = \text{Left obstacle distance from the robot} \tag{13a}$$

$$y_2^{\{1\}} = \text{Front obstacle distance from the robot} \tag{13b}$$

$$y_3^{\{1\}} = \text{Right obstacle distance from the robot} \tag{13c}$$

$$y_4^{\{1\}} = \text{Target bearing of the robot} \tag{13d}$$

These input values are distributed to the hidden neurons which generate outputs given by

$$y_2^{\{\text{lay}\}} = f\left(V_j^{\{\text{lay}\}}\right) \tag{14}$$

where

$$V_j^{\{\text{lay}\}} = \sum\nolimits_i W_{ji}^{\{\text{lay}\}} \cdot y_i^{\{\text{lay}-1\}} \tag{15}$$

where lay = layer number (2 or 3); j = label for jth neuron in hidden layer 'lay'; i = label for ith neuron in hidden layer 'lay − 1'; $W_{ji}^{\{\text{lay}\}}$ = weight of the connection from neuron i in layer 'lay − 1' to neuron j in layer 'lay'; $f(\cdot)$ = activation function, chosen in this work as

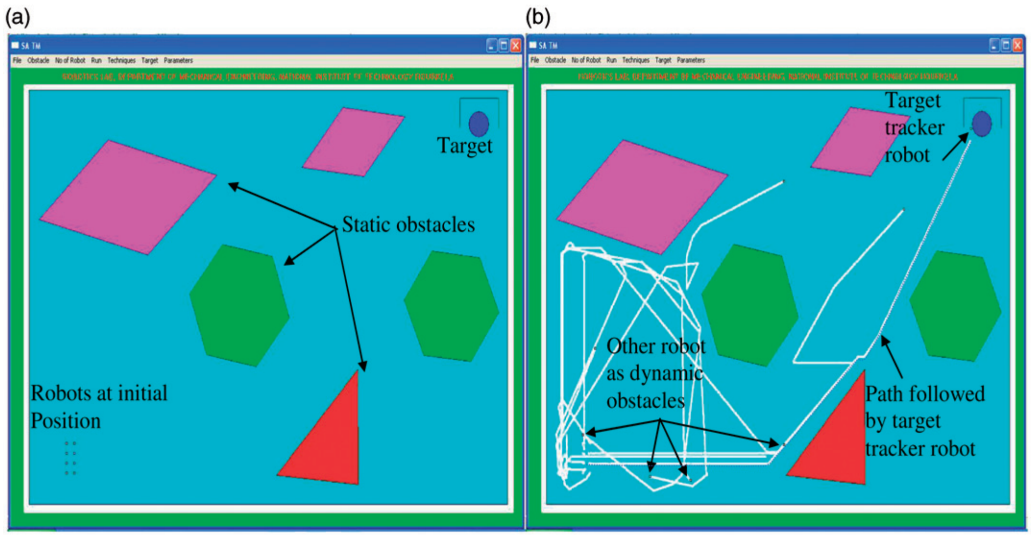

(a)
(b)
Target
Static obstacles
Robots at initial Position
Target tracker robot
Other robot as dynamic obstacles
Path followed by target tracker robot

Figure 5. Static as well as dynamic obstacle avoidance behaviour. (a) At initial position before simulation; (b) navigational path after simulation.

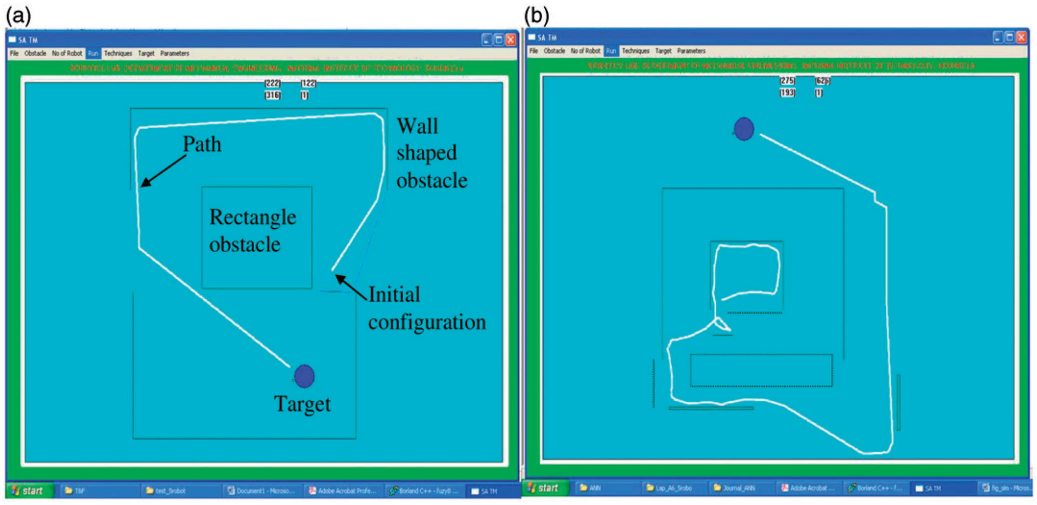


Figure 6. (a) Robot with wall following behaviour; (b) robot escaping from dead end obstacle.

the hyperbolic tangent function

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{16}$$

During training, the network output θ_{actual} may differ from the desired output θ_{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 θ_{desired} and θ_{actual} for the set of presented training patterns.

$$E_{\text{rr}} = \frac{1}{2} \sum_{\substack{\text{all training} \\ \text{patterns}}} (\theta_{\text{desired}} - \theta_{\text{actual}})^2 \tag{17}$$

The error back propagation method is employed to train the network (Haykin 2006). This method requires the computation of local error gradients in order to determine appropriate weight corrections to reduce error. For the output layer, the error gradient $\delta^{\{4\}}$ is

$$\delta^{\{4\}} = f'(V_1^{\{4\}})(\theta_{\text{desired}} - \theta_{\text{actual}}) \tag{18}$$

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

$$\delta_j^{\{\text{lay}\}} = f'\left(V_j^{\{\text{lay}\}}\right)\left(\sum\nolimits_k \delta_k^{\{\text{lay}+1\}} W_{kj}^{\{\text{lay}+1\}}\right) \tag{19}$$

The synaptic weights are updated according to the following expressions

$$W_{ij}(t+1) = W_{ij}(t) + \Delta W_{ij}(t+1) \tag{20}$$

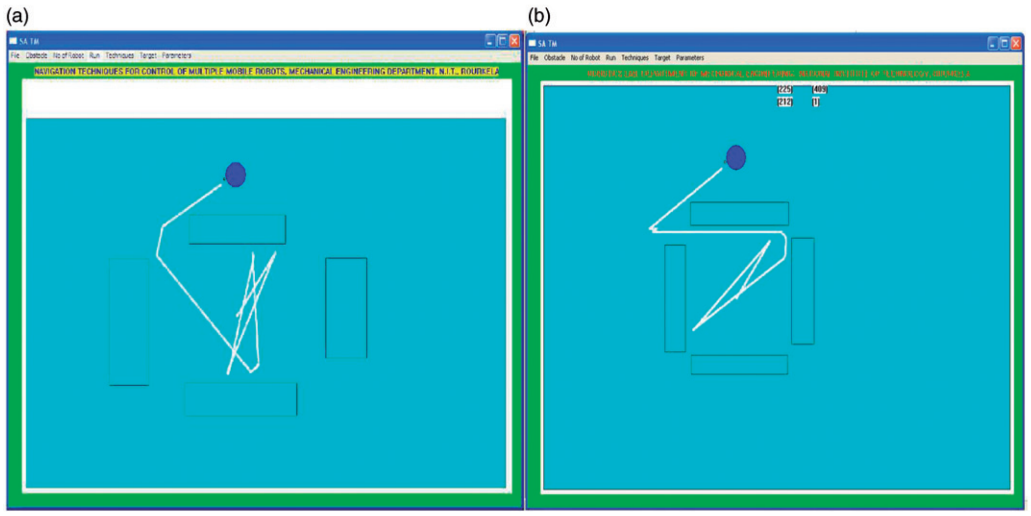

(a)
SA TM
File Obstacle No of Robot Run Techniques Target Parameters
NAVIGATION TECHNIQUES FOR CONTROL OF MULTIPLE MOBILE ROBOTS, MECHANICAL ENGINEERING DEPARTMENT, N.I.T., ROURKELA
(b)
SA TM
File Obstacle No of Robot Run Techniques Target Parameters
[225] [409]
[212] [1]

Figure 7. Result comparisons with fuzzy controller. (a) Navigation path of mobile robot using fuzzy controller by Pradhan et al. (2009); (b) navigation path of mobile robot using proposed neural controller.

Table 2. Simulation results comparison between the fuzzy controller developed by Pradhan et al. (2009) and the current developed neural controller.

Sl. No.	Navigation with different method	Fuzzy controller (Figure 7a)	Neural controller (Figure 7a)	Result deviation (%)
01.	Length of path (in meter)	13.8	12.2	11.59
02.	Time taken (seconds)	14.67	12.97	11.59

and

$$\Delta W_{ij}(t+1) = \alpha \Delta W_{ij}(t) + \Delta \eta \delta_j^{\{\text{lay}\}} y_i^{\{\text{lay}-1\}} \tag{21}$$

where $\alpha =$ momentum coefficient (chosen empirically as 0.2 in this work); $\eta =$ 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

$$\theta_{\text{actual}} = f\left(V_1^4\right) \tag{22}$$

where

$$V_1^4 = \sum\nolimits_i W_{1i}^{\{4\}} y_i^{\{3\}} \tag{23}$$

It should be noted that learning can take place continuously even during normal target seeking behaviour. This enables the controller to adopt the changes in the robot's path while moving towards the target. The proposed controller and kinematics gives steering angle from wheel velocities based on the environmental conditions. Mainly, three behaviours (obstacle avoidance, wall following and target seeking) are required to train and design an intelligent controller for a mobile robot being used to navigate in a cluttered environment.

4. Simulation results and discussions

Existing approaches for learning to control a mobile robot rely on supervised methods, where correct behaviour is explicitly given. The simulation results present the effectiveness of a novel approach that evolves the neural network controller. The series of simulation tests have been conducted with the ROBNAV software being developed in the laboratory using C++ (Appendix A). To demonstrate the effectiveness and robustness of the proposed method, simulation results on mobile robot navigation in various environments are exhibited.

For visual guidance of behaviour, in navigation the perception of motion plays a prominent role. Visual motion provides information about the structure of the environment. An important part of robot behaviour is avoidance of obstacles. Examples of static obstacles include walls, poles, fences, trees etc., as well as other moving obstacle like vehicles, people, animals etc. Encountering such objects can cause avoidance behaviour, which consists of any combination of slowing down, turning and stopping. The obstacle avoidance behaviour is activated when the readings from any sensors are less than the minimum threshold values. This is how the robot determines whether an object is close enough for a collision. When an object is detected

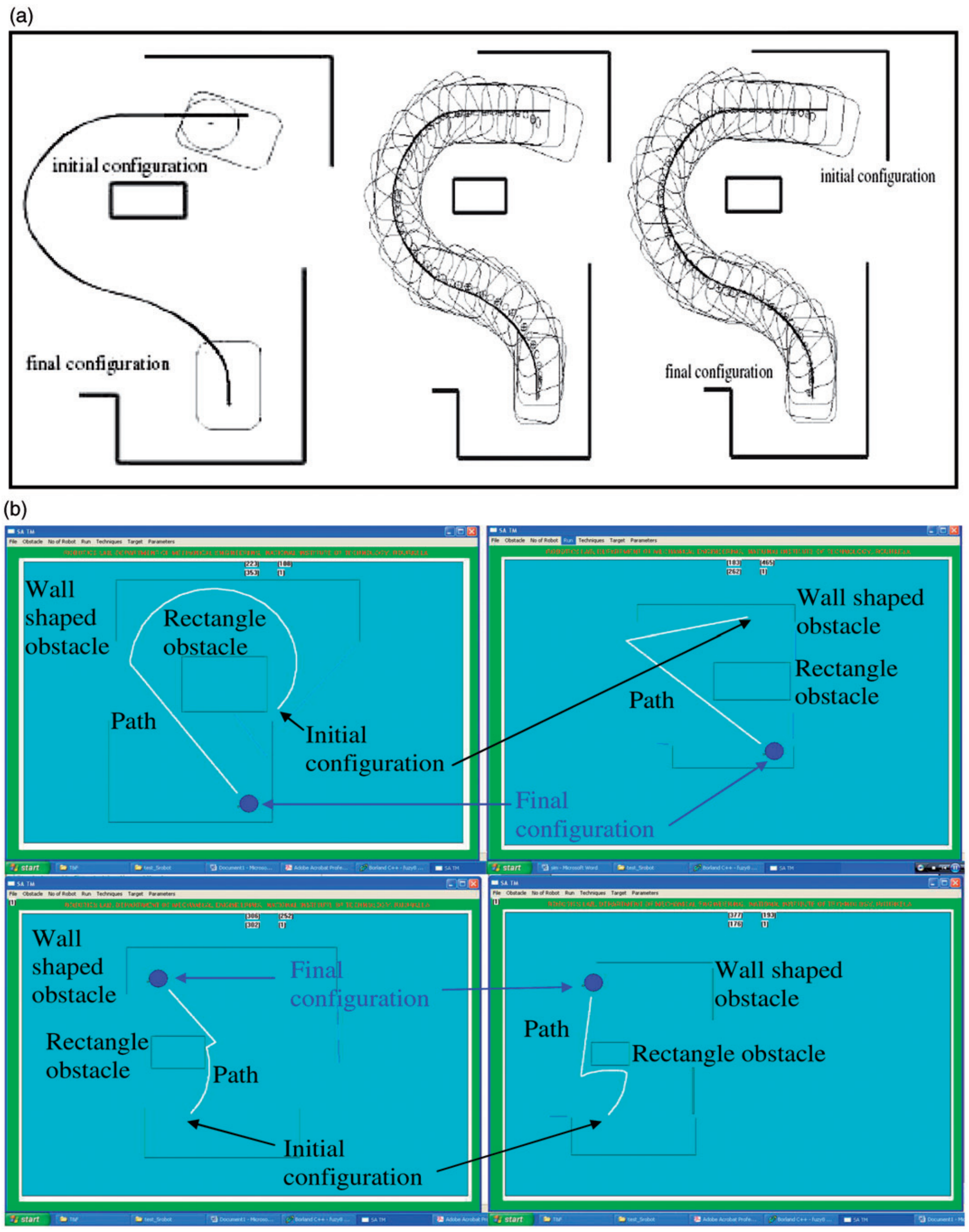


Figure 8. Result comparisons with Hamel et al. (2001); (a) experimental result of planned path by Hamel et al. (2001); (b) navigation of mobile robot using developed controller.

as too close to the robot, it avoids a collision by moving away from it in the opposite direction. Collision avoidance has the highest priority and, therefore, it can override other behaviours; in this case, its main reactive behaviour is decelerating for static as well as dynamic obstacle avoidance as shown in Figures 5(a) and (b) in different scenarios.

The wall following behaviour is required to move from one room to another room. The wall following behaviour mode will be adopted when the mobile robot detects an obstacle in front while it is moving towards the target, the mobile robot may turn left or right because of presence of the obstacle in the front. Another special condition appears as the mobile robot detects an obstacle in the front while the target tracking control mode is on operation. In this case, the fixed wall following behaviour should be performed first; that is, the mobile robot must rotate clockwise or

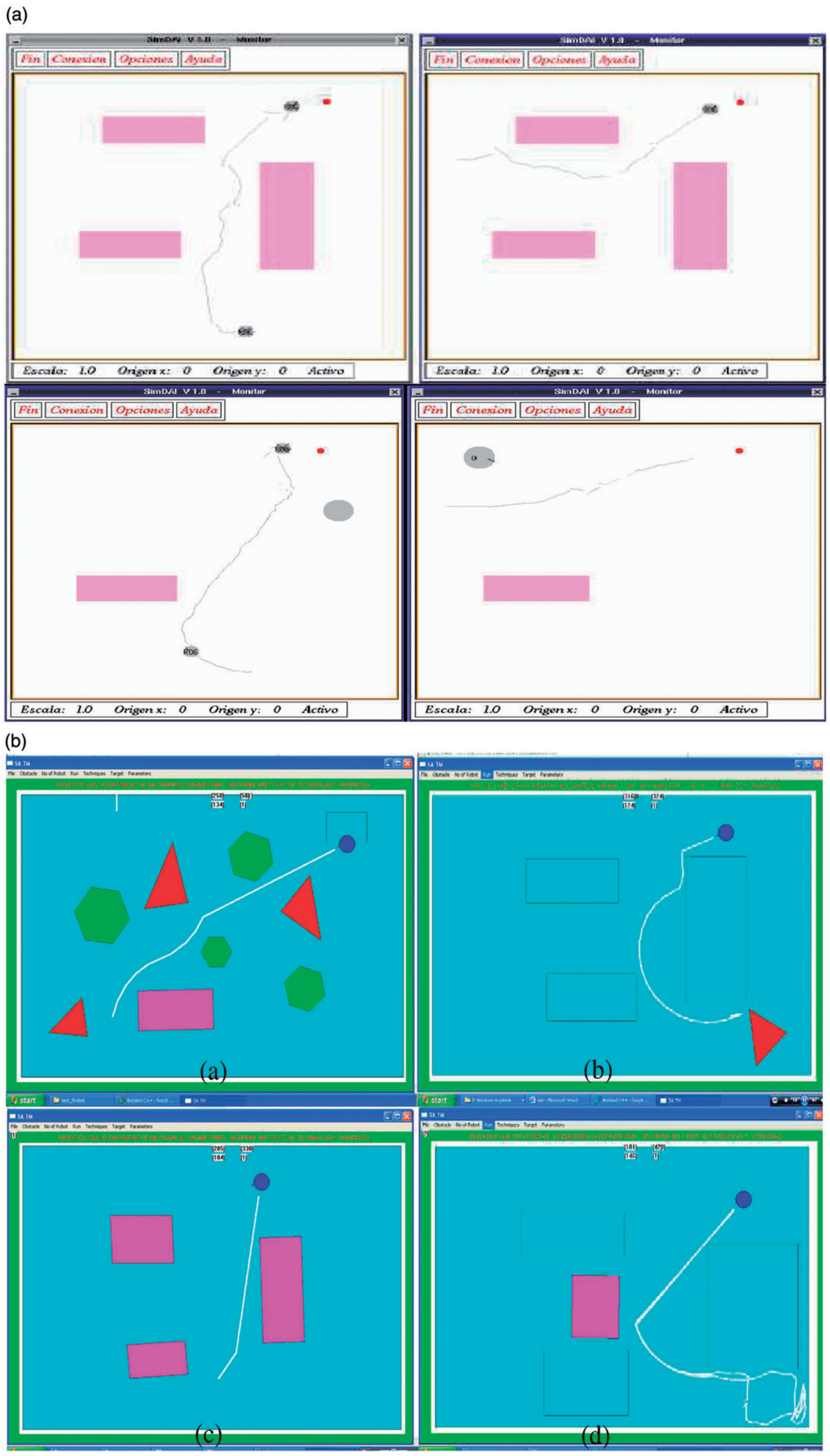


Figure 9. Result comparisons with Sanchis et al. (2001). (a) Static and dynamic experimental result by Sanchis et al. (2001); (b) static and dynamic simulation result by developed neural controller.

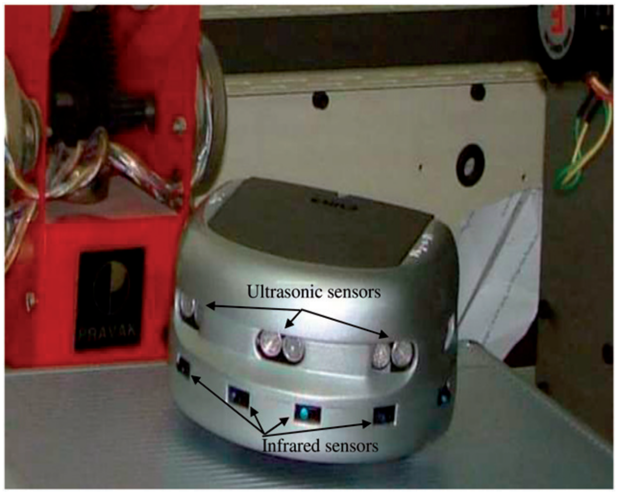


Figure 10. KHEPERA-III mobile robot used for experimental results.

counterclockwise such that it can align and move along the wall (Figure 6(a) and (b)). In the absence of wall following behaviour, the robot is incapable of reaching the goal position when it encounters U shaped or dead end obstacles in its path. In such a situation, the robot should keep on heading towards the goal position. But when it moves towards the goal position, the robot also comes closer to the obstacles. Any obstacle-avoidance behaviour except wall following would make the robot divert from its goal position. To avoid such a situation, it must reflect following edge behaviour so that the robot may locate, find and reach the specified target as shown in Figure 6(a) and escape from the dead end shown in Figure 6(b).

Target searching algorithms assume that the goal state is fixed and does not change during navigation of mobile robot. For example, in the problem of moving from the current location to a desired goal location along a network of paths, it is assumed that the target location is fixed and does not change during the navigation. When the acquired information from the sensors shows that there are no obstacles around the robot, its main reactive behaviour is target steer. A neural controller mainly adjusts the robot's motion direction and quickly moves it towards the target if there are no obstacles around the robot, as shown in Figure 5(a) and (b). In the proposed control strategy, reactive behaviours are formulated and trained by the neural network.

In order to exhibit the superiority of the proposed neural controller of the mobile robot, the results obtained from the developed neural controller for the mobile robot have been compared with the results obtained from the fuzzy controller by Pradhan et al. (2009) (Figure 7a) in the same environment, shown in Table 2. The length of path and time required to reach the target are less than the fuzzy controller (Figure 7b). The results obtained using the feedback control law by Hamel et al. (2001) (Figure 8a) have been compared with the results obtained from the proposed neural controller of the mobile robot (Figure 8b). Also, a comparison has been done between the static and dynamic result using classifier systems described by Sanchis et al. (2001), shown in Figure 9(a) and results obtained from the current developed controller (Figure 9b). The comparisons show a good agreement.

5. Experimental results

The navigation method has been tested in a series of experiments to exhibit its effectiveness. All experiments were carried out using C++ software loaded on a KHEPERA-III mobile robot. The chassis of the robot measured 130×70 mm ($D \times H$), weight 690 g, payload 2000 g approximately. Fitted DsPIC 30F5011 at 60 MHz processor with 4 KB on DsPIC and 64 MB KoreBot RAM. 2 DC brushed servo motors with incremental encoders (roughly 22 pulses per mm of robot motion). Nine infrared proximity and ambient light sensors with up to 25 cm range were placed around the robot and two were placed on the bottom. The bottom infrared ground proximity sensors were used for experiments like the line following application. They were positioned and numbered as shown in Figure 1. There were also five pairs of ultrasonic sensors, where each pair was composed of one transmitter and one receiver, as shown in Figure 10. The ultrasonic sensors were powered by a 20 V dc source. The nominal frequency of these transducers was 40 kHz +/− 1 kHz. Parameters such as max number of echo, timeout and active sensors were parametrisable through a configuration. When the upper body was mounted, there was some noise because of inside rebound echo, which was deleted by the software. In fact, detection of the nearest obstacles could improve (0.2–0.4 cm) by removing the upper body. Each measure returned the number of found echoes, the distance in cm of each echo, the amplitude of each, and the time when the echo was seen. The speed range of this mobile robot was 0–0.5 m/s. A power adapter lithium-polymer battery pack (1400 mAh) was used. The assumptions about the mechanical structure and

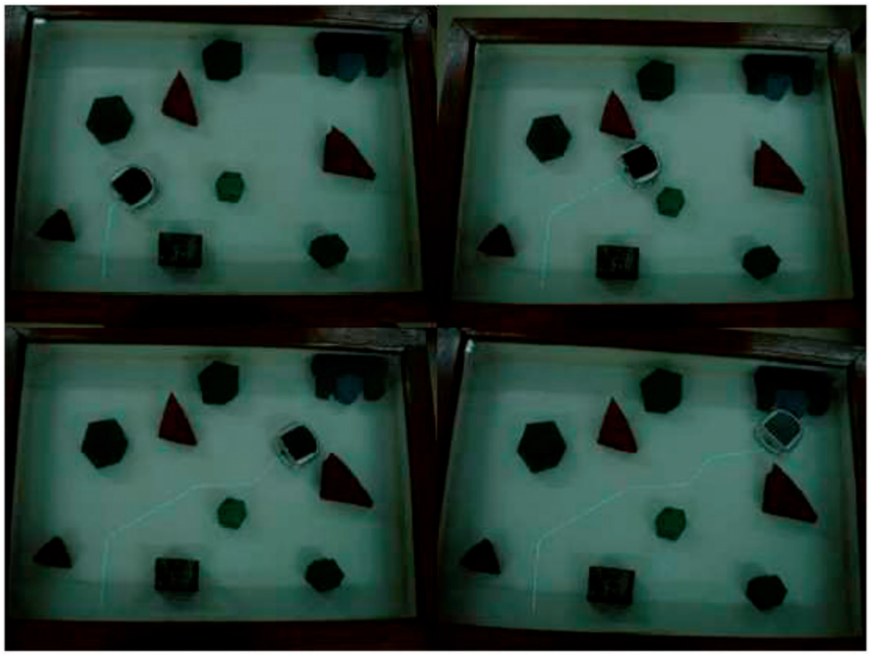


Figure 11. Experimental validation of navigation in unknown environment.

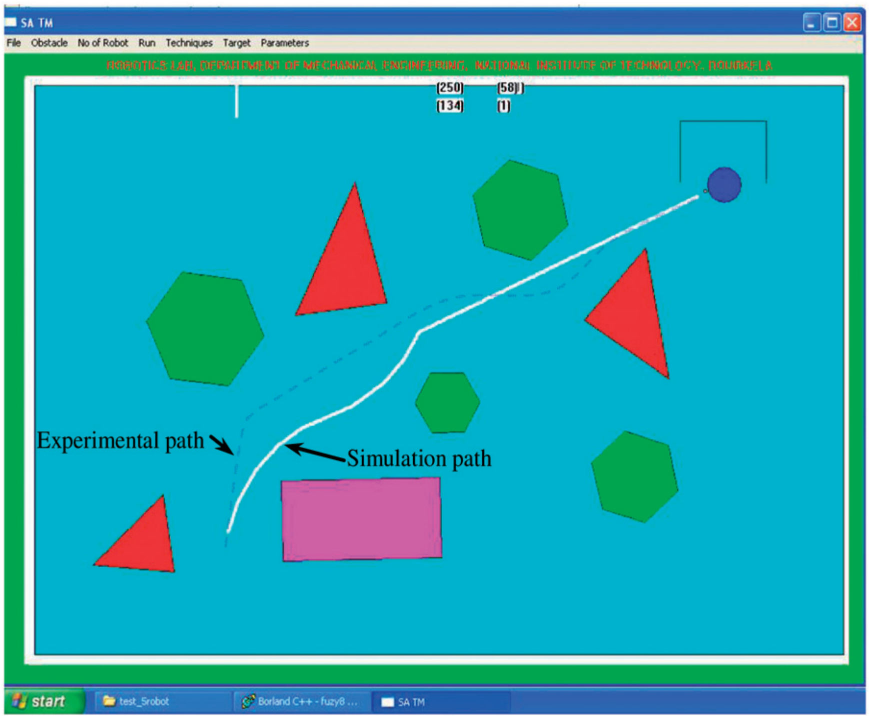


Figure 12. Experimental result comparison with simulation mode.

Table 3. Time taken by robots in simulation and experiment to reach targets.

Sl. No.	Experiments	Path length (in pixel)	Time during simulation (s)	Time during experiment (%)
01	For 1st experiment	134	1.58	3.16
02	For 2nd experiment	135	1.59	3.18
03	For 3rd experiment	134	1.58	3.16
04	For 4th experiment	135	1.59	3.19
05	For 5th experiment	134	1.58	3.17

the motion of a mobile robot to which our proposed method was applied are as follows:

(a) The mobile robot moves on a plane surface (on lab specified floor area).
(b) The wheel of a mobile robot rolls on the floor without any translational slip.

In the experiment to verify the effectiveness of the control algorithms the experimental paths followed by mobile robots to reach the target were traced as shown in Figure 11. From the neural controller (inputs: left, front, right obstacle distances and heading angle) after learning, training and testing, robots got left and right wheel velocities which subsequently gave the new steering angles. The experimentally obtained paths followed closely those traced by the robots during simulation (Figure 12). During the experiment, it was found that the experimental path length and time were more than the simulation path length and time. This was due to the presence of various errors (e.g. signal transmission error in the data-cable, obstacle/target tracking error, presence of friction in the rotating elements, slippage between floor and wheels, friction between supported point and floor, etc.). Table 3 shows the time taken by the robots in the simulations and in the experimental tests while finding the targets.

The experiment was conducted in the laboratory for different environmental scenarios. From these figures, it can be seen that the robots can indeed avoid obstacles and reach the targets. These robotic behaviours were verified in simulation and experimental modes (Figure 12). It was observed that the robots were able to reach the targets efficiently during simulation and experiment. It was found that the robot with the neural controller had better performance than with the fuzzy controller in terms of positioning accuracy and collision avoidance. Also neural controllers require less computing time and computing memory than the fuzzy controller and provide optimised paths to reach the goal.

6. Conclusion

The conclusion drawn on the bases of the kinematics, simulations and experimental analysis are depicted as follows.

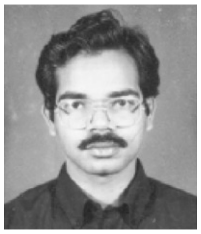
Both in simulation and experimental modes, the developed controller worked efficiently. The simulation results are also compared with the results obtained from the other investigations and they show very good agreement. A back-propagation neural network is used to design the controller. A software tool is developed using C++ to obtain the simulation results. The developed neural controller has the following salient features:

(1) Mobile robots are able to avoid any static and dynamic obstacles on their path. In a dynamic environment, the proposed neural controller can be efficiently applied.
(2) During navigation using sensor information, robots are able to map the surroundings.
(3) On comparison with three various approaches (i.e. feedback control law, classifier systems and fuzzy controller) it is found that the developed controller is simple but efficient for navigation of a mobile robot in a dynamic environment.
(4) Training patterns of each network can be generated by simulation rather than by experiment, saving considerable time and effort.

Simulation results are validated by experimental results. Hence the simulation environment can be used for future analysis of the navigational path.

Notes on contributors

Mukesh Kumar Singh received his PhD in Mobile Robot Navigation from National Institute of Technology Rourkela, Orissa, India and his MTech in CAD/CAM from Motilal Nehru Regional Engineering College (now the National Institute of Technology), Allahabad, Utter Pradesh, India. He has 13 years of research and teaching experience in his fields. Presently, he is engaged in mobile robot navigation research and is a faculty member of the Department of Mechanical Engineering, Government Engineering College Bilaspur, Chhattisgarh, India.



Dayal R. Parhi received his first Ph.D. in Mobile Robotics from Cardiff School of Engineering, UK and second PhD in Vibration Analysis of Cracked structures from Sambalpur University, Orissa, India. He has 16 years of research and teaching experience in his fields. Presently, he is engaged in Mobile robot Navigation research, and a faculty member in the Department of Mechanical Engineering, National Institute of Technology Rourkela-08, Orissa, India.

References

Braganza, D., Dawson, D.M., Walker, I.D., and Nath, N. (2007), 'A Neural Network Controller for Continuum Robots', *IEEE Transactions on Robotics*, 23, 1270–1277.

Corradini, M.L., Ippoliti, G., and Longhi, S. (2003), 'Neural Networks Based Control of Mobile Robots: Development and Experimental Validation', *Journal of Robotic Systems*, 20, 587–600.

Das, T., and Kar, I.N. (2006), 'Design and Implementation of an Adaptive Fuzzy Logic-based Controller for Wheeled Mobile Robots', *IEEE Transactions on Control system Technology*, 14, 501–510.

Folgheraiter, M., Gini, G., Nava, A., and Mottola, N. (2006) 'A Bio Inspired Neural Controller For a Mobile Robot', *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, Kunming, China, 17–20 December, pp. 1646–1651.

Gregor, K., and Igor, S. (2007), 'Tracking-Error Model-Based Predictive Control for Mobile Robots in Real Time', *Robotics and Autonomous Systems*, 55, 460–469.

Haykin, S. (2006), *Neural Networks: A Comprehensive Foundation* (2nd ed.), India: Pearson Prentice Hall.

Hamel, T., Soueres, P., and Meizel, D. (2001), 'Path Following with a Security Margin for Mobile Robots', *International Journal of Systems Science*, 32, 989–1002.

Kondo, T. (2007), 'Evolutionary Design and Behaviour Analysis of Neuromodulatory Neural Networks for Mobile Robots Control', *Applied Soft Computing*, 7, 189–202.

Muñiz, F., Zalama, E., Gaudiano, P., and López Coronado, J., (1995) 'Neural Controller for a Mobile Robot in a Nonstationary Environment, in *Second IFAC Conference on Intelligent Autonomous Vehicles*, Espoo, Finland, 12–14 June, pp. 279–284.

Nelson, A.L., Grant, E., and Henderson, T.C. (2004), 'Evolution of Neural Controllers for Competitive Game Playing with Teams of Mobile Robots', *Robotics and Autonomous Systems*, 46, 135–150.

Nolfi, S., and Tani, J. (1999), 'Extracting Regularities in Space and Time Through a Cascade of Prediction Networks: The Case of a Mobile Robot Navigating in a Structured Environment', *Connection Science*, 11, 125–148.

Oriolo, G., Panzieri, S., and Ulivi, G. (2000), 'Learning Optimal Trajectories for Non-holonomic Systems', *International Journal of Control*, 73, 980–991.

Pallottino, L., Scordio, V.G., Bicchi, A., and Frazzoli, E. (2007), 'Decentralized Cooperative Policy for Conflict Resolution in Multivehicle Systems', *IEEE Transactions on Robotics*, 23, 1170–1183.

Parhi, D.R. (2000) 'Navigation of Multiple Mobile Robots in an Unknown Environment', Ph.D. thesis, Cardiff School of Engineering, University of Wales, UK.

Parhi, D.R., Pradhan, S.K., Panda, A.K., and Behra, R.K. (2009), 'The stable and Precise Motion Control for Multiple Mobile Robots', *Applied Soft Computing*, 9, 477–487.

Pradhan, S.K., Parhi, D.R., Panda, A.K., and Behra, R.K. (2009), 'Fuzzy Logic techniques for Navigation of Several Mobile Robots', *Applied Soft Computing*, 9, 290–304.

Racz, J., and Dubrawski, A. (1995), 'Artificial Neural Network for Mobile Robot Topological Localization', *Robotics and Autonomous Systems*, 16, 73–80.

Sanchis, A., Isasi, P., Molina, J.M., and Segovia, J. (2001), 'Applying Classifier Systems to Learn the Reactions in Mobile Robots', *International Journal of System Science*, 32, 237–258.

Sgouros, N.M. (2001), 'Qualitative Navigation for Mobile Robots in Indoor Environments', *Applied Artificial Intelligence*, 15, 237–251.

Tchon, K., and Jakubiak, J. (2006), 'Extended Jacobian Inverse Kinematics Algorithm for Nonholonomic Mobile Robots', *International Journal of Control*, 79, 895–909.

Wolf, D.F., and Sukhatme, G.S. (2008), 'Semantic Mapping Using Mobile Robots', *IEEE Transactions on Robotics*, 24, 245–258.

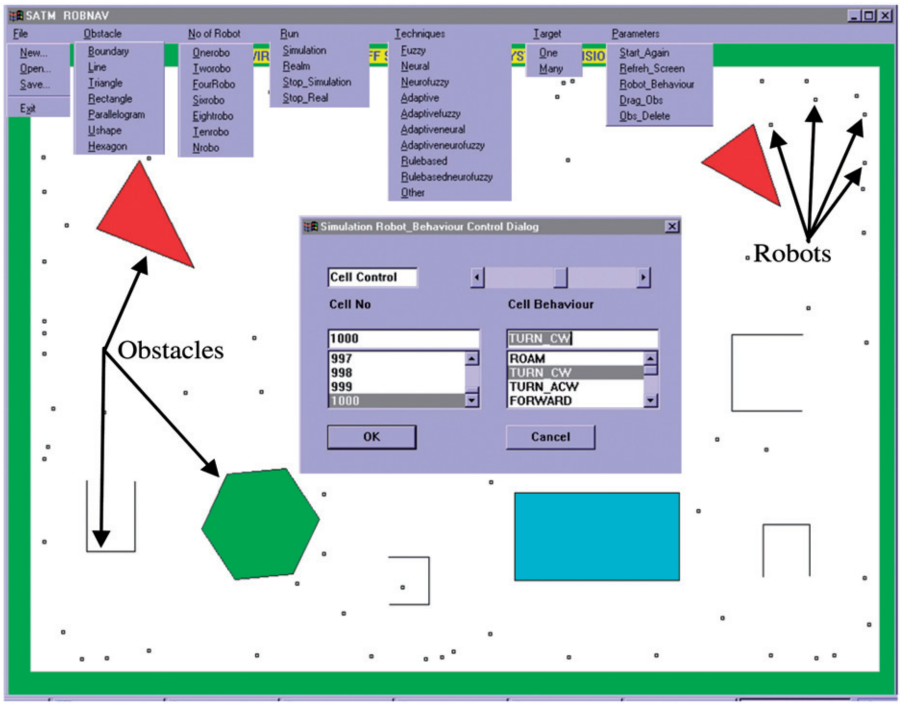
Yang, S.X., and Meng, M. (2000), 'An Efficient Neural Network Approach to Dynamic Robot Motion Planning', *Neural Networks*, 13, 143–148.

Zhang, Q., Shippen, J., and Jones, B. (1999), 'Robust Backstepping and Neural Network Control of a Low Quality Nonholonomic Mobile Robot', *International Journal of Machine Tools Manufacturing*, 39, 1117–1134.

Appendix A
ROBNAV software

The typical screen of 'ROBNAV' software has been developed using C++, shown in Figure A1 (Parhi 2000). The software runs under WINDOWS NT/95/98/2000/XP/Vista. The menus incorporated in the software are shown in Figure A1.

(3) **Run Menu:** With this menu, the user can choose to run the software in simulation mode or control the navigation of real mobile robots.
(4) **Techniques Menu:** This menu enables the user to select the techniques to control the navigation of the robots.
(5) **Target Menu:** This menu is for placing targets in the environment.



Cell Control

Cell No: 1000 (997, 998, 999, 1000)

Cell Behaviour: TURN_CW (ROAM, TURN_CW, TURN_ACW, FORWARD)

Figure A1. The typical screen of ROBNAV software used for navigation of mobile robots.

(1) **Obstacle Menu:** The Obstacle Menu allows the user to draw different types of obstacles in the robots' environment.
(2) **Number of Robot Menu:** Using this menu, a user can draw any number of robots ($\leq$1000) as required to be placed in the environment.
(6) **Parameter Menu:** This menu enables the user to start again a process, refresh the screen, robot behaviour to select a particular robot and control its movements manually, and drag any of the obstacles to any place in the environment, and delete an obstacle from the environment.