Pergamon

Robotics & Computer-Integrated Manufacturing, Vol. 11, No. 4, pp. 271–278, 1994
Copyright © 1995 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0736-5845/94 $7.00 + 0.00

0736–5845(95)00003–8

● *Paper*

NEURAL NETWORK MBPC FOR MOBILE ROBOT PATH TRACKING

J. GOMEZ-ORTEGA and E. F. CAMACHO

Departamento Ingenieria de Sistemas y Automática, Universidad de Sevilla, Avenida Reina Mercedes s/n, Spain

This paper presents a way of implementing a model-based predictive controller (MBPC) for mobile robot path tracking. The method uses a non-linear model of mobile robot dynamics and thus allows an accurate prediction of the future trajectories. Constraints on the maximum attainable speeds are also considered by the algorithm. A multilayer perceptron is used to implement the MBPC. The perceptron has been trained to reproduce the MBPC behaviour in a supervised way. Experimental results obtained when applying the neural network controller to a TRC labmate mobile platform are given in the paper.

1. INTRODUCTION

One of the most important issues in the design and development of intelligent mobile robots is the navigation problem which is the ability of a vehicle to plan and execute collision-free motions within its environment.

This problem can be divided into two hierarchical levels. The higher level, called global navigation or path planning, is concerned with the generation of a trajectory (space–time) from an initial configuration to a goal configuration, avoiding the static and mobile obstacles in the environment, which are considered to be known. This planning stage is accomplished off-line. Although some works presented in the literature consider the kinematic and dynamic models of the vehicle,[22] most of them consider only the geometric approach to the problem, and are usually based on the *configuration space* approach.[9] Their computation time is not acceptable for real-time control of mobile robots. Some well-known solutions have been proposed. Fujimura and Samet[5] included time as one of the dimensions of the model world. This allowed them to regard the moving obstacles as being stationary in the extended world. Kant and Zucker[8] proposed a solution based on dividing the trajectory planning problem (TPP) into two subproblems: the path planning problem (PPP), which is concerned with planning the path to avoid stationary obstacles, and the velocity planning problem (VPP), which is concerned with planning the velocities along the path to avoid moving obstacles. Although this reduces the complexity of the global problem, this solution does not change the predefined path and it cannot avoid moving obstacles with trajectories collinear to the robot's. Erdmann and Lozano-Perez[4] proposed a solution based on a planner for moving objects that constructs a configuration space each time an object in the scene changes its velocity.

The lower level, called local navigation or path tracking, is concerned with driving the vehicle through the trajectory generated by the global planner. The objective can be either to follow a desired path defined by a set of geometric primitives (such as right segments, arcs, target points), as in this work, or simply to follow some environmental feature, like a road edge.[14] As the future desired outputs are known previously, a model-based predictive control technique (MBPC) seems to be a suitable approach for solving this problem.

The objective of the MBPC is to drive future system outputs (in this case robot position and heading) close to the desired values in some way, bearing in mind the control activity required to do so. At each sampling interval, and by using a suitable model of the plant, a sequence of control actions is computed in such a way that the predicted output will be driven as close as possible to the desired path. This is accomplished by minimizing a quadratic function that measures the tracking errors and the control effort over the costing and control horizons.

If the plant is linear and signals are not bounded, the MBPC leads to a linear set of equations that have to be solved at each sampling period.[3] If the process is time invariant, most of the computation has to be carried out only once and the MBPC requires little computation time. If the control signal is constrained (as is always the case in real processes), the MBPC results in a much more complex and time consuming problem: that is, a quadratic problem with linear constraints.[2]

Acknowledgement— The authors would like to acknowledge the CICYT for funding this work under grants TAP93-0408 and TAP93-0581.

Hopfield neural networks have been proposed in Ref. 15 to reduce the computation time needed for processes that can be modelled by the reaction curve method. When the system is non-linear (as is the case of mobile robots) the MBPC turns out to be a much more complex and time consuming problem.

This paper presents a way of implementing MBPC for mobile robot path tracking that solves the problems mentioned above. An MBPC was proposed by Ollero and Amidi,[13] but only a linearized prediction model for future positions and headings of the mobile robot was used and robot constraints, such as maximum velocities, were not taken into account. However, linear models are not accurate enough when the angle between the robot heading and the desired path orientation is not sufficiently small. The method presented here uses a non-linear and therefore more precise model of the mobile robot, thus allowing a more accurate prediction of the future trajectories. Constraints on the maximum attainable speeds are also considered by the algorithm. As the resulting MBPC gives rise to a very complex optimization problem that has to be solved by numerical methods requiring high computation time, a multilayer perceptron is used to implement the MBPC controller. The perceptron has been trained to reproduce the MBPC controller behaviour in a supervised way.

The paper is organized as follows. Section 2 presents the MBPC approach. The neural network solution is described in Section 3. Experimental results obtained when applying the neural network controller to a TRC labmate mobile platform are given in Section 4. Finally, some concluding remarks are made in Section 5.

2. MODEL-BASED PREDICTIVE CONTROL PATH TRACKING

MBPC is a set of optimal control techniques that have inspired much research work in recent years. The MBPC approach consists of applying a control sequence that minimizes a multistage cost function of the form

$$J(H_1, H_2, H_3) = E\left\{ \sum_{i=H_1}^{H_2} \xi(i)[\hat{Y}(t+i|t) - Y_d(t+i)]^2 + \sum_{i=1}^{H_3} \lambda(i)[\Delta U(t+i-1)]^2 \right\}$$

where $E\{\cdot\}$ is the expectation operator, $\hat{Y}(t+i|t)$ is an i-step prediction of the system outputs, $Y_d(t+i)$ is a future desired output sequence, $U(t+i)$ is the control signal vector foreseen for the sampling time $t+i$ at instant t, and $\Delta = 1 - q^{-1}$, where q^{-1} is the backward shift operator. In this function, the position errors and the control effort are penalized through the weighting factors sequences $\xi(i)$ and $\lambda(i)$. H_1 and H_2 are the minimum and maximum prediction horizons, respectively, and H_3 is the costing horizon. In order to reduce the dimension of the optimization problem usually infinity value is given to the weighting factors after a control horizon H_c.[1,16]

The objective function used here is:

$$J(H_1, H_2, H_3) = E\left\{ \sum_{i=H_1}^{H_2} [\hat{Y}(t+i) - Y_d(t+i)]^2 + \sum_{i=1}^{H_3} (\lambda_1([\Delta\omega_r(t+i-1)]^2 + [\Delta\omega_l(t+i-1)]^2) + \sum_{i=1}^{H_3} (\lambda_2[\omega_r(t+i-1) - \omega_l(t+i-1)]^2) \right\}$$

where $\hat{Y}(t+i|t) = \{x_d(t+i|t), y_d(t+i|t)\}$ is an i-step prediction of the robot position, ω_r and ω_l are the right and left angular velocities of the two driving wheels, which are the control variables, and λ_1, λ_2 are constant weighting factors. The objective of predictive control is to compute the future control sequence $(\omega_r(t), \omega_l(t))$, $(\omega_r(t+1), \omega_l(t+1))$, ... in such a way that the future robot position $Y(t+i)$ is driven close to $Y_d(t+i)$. This is accomplished by minimizing J, where the control activity for doing so has been taken into account. The first term in J penalizes the position error, the second term penalizes the acceleration and the third penalizes the robot angular velocity. The last two terms ensure smooth robot guidance. An error in the robot heading could be considered in J but it has been noticed that it is not necessary when the control horizon H_c is sufficiently large.

A receding horizon approach is used and only the first element of the computed control sequence is applied. The process is repeated in the next sampling intervals, resulting in a control law that belongs to the class known as Open-Loop-Feedback–Optimal control. A block diagram of the system is shown in Fig. 1. Notice that the minimum output horizon H_1 should be set to a value bigger than the dead time d of the system, since the output for smaller time horizons cannot be affected by the first action $[\omega_r(t), \omega_l(t)]$. In the following H_1 and H_2 will be considered to be $H_1 = d+1$ and $H_2 = d + H_c$, and H_3 is given a value of H_c. In this formulation it is assumed that after the control horizon H_c, further increments in control are zero.

The predictive problem, formulated under these circumstances, has to be solved with numerical optimization methods, which are not acceptable for real-time control. The controller proposed in this work will be implemented using a neural network scheme, which allows real time.

2.1. *Prediction model*

For an MBPC formulation, a dynamic model of the mobile platform is needed to predict the future positions and headings of the robot. As a testbed for the experiments, a TRC LABMATE[22] mobile robot has been used (Fig. 2).

The mobile robot has been designed for indoor environments. Its dimensions are 0.8×0.8 m with a weight of 50 kg (without batteries). The carrying

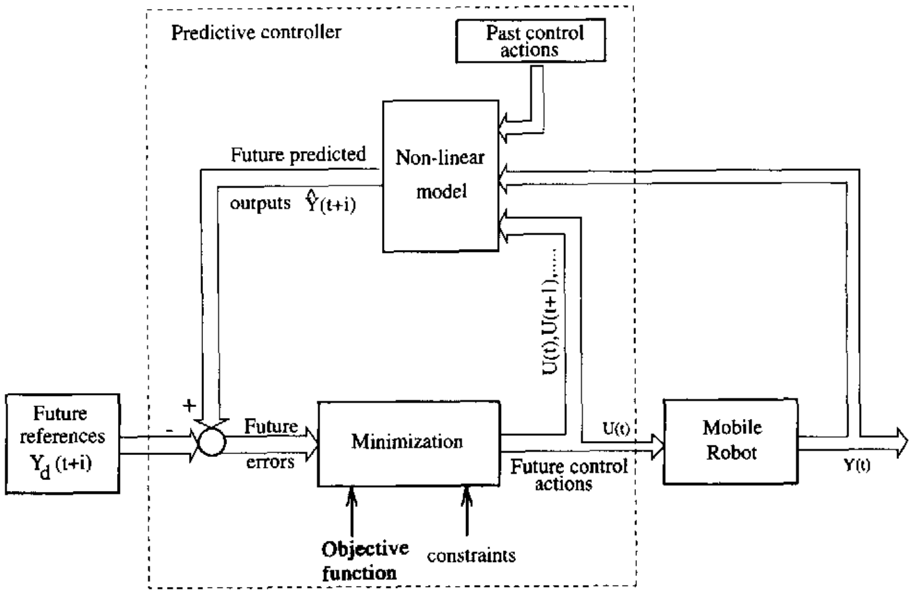


Fig. 1. The predictive controller scheme.

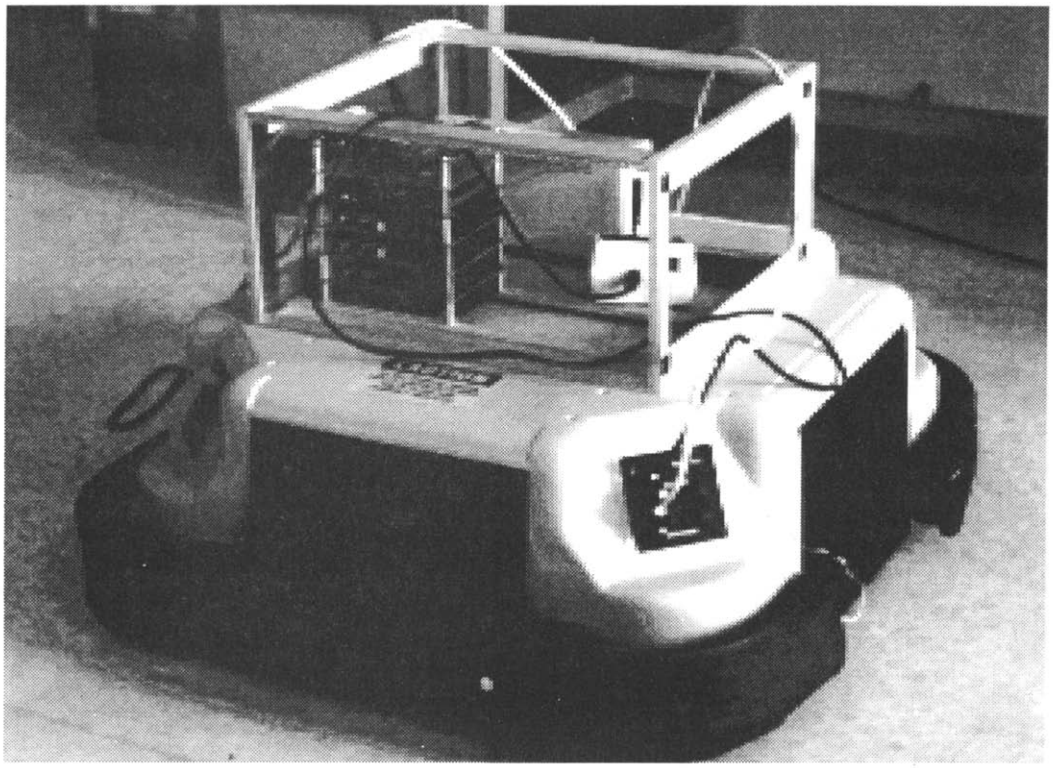


Fig. 2. The LABMATE mobile robot.

capacity for payload is up to 90 kg. It provides the mechanical base, a two wheel differential driving system (supported by four passive casters), as well as the servocontrol of the motors. The maximum attainable linear velocity is 1000 mm/sec. A RS-232 radio-modem interface (of 915 MHz) is used to communicate the host (where the MBPC neural network algorithm is executed) with the low level control microprocessor. The robot architecture is shown in Fig. 3.

A model of the LABMATE mobile robot taking into account the low level servocontrol dynamics, as well as the dead time produced by communications with the host process, was obtained by using kinematic equations and identification tests. A more detailed model can be found in Ref. 6.

The following dynamic model (which corresponds to a differential-drive vehicle) is used for computing the predictions:

$$\theta(k+1)=\theta(k)+\mathscr{A}T$$

$$x(k+1)=x(k)+\frac{V}{\mathscr{A}}(\sin(\theta(k)+\mathscr{A}T)-\sin(\theta(k)))$$

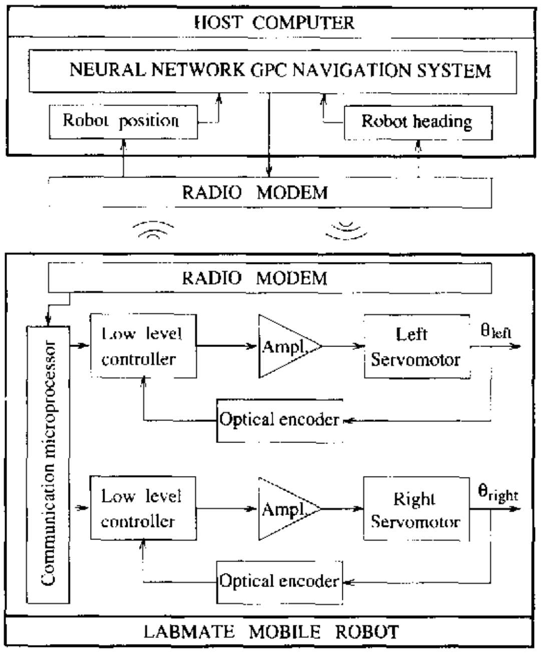


Fig. 3. LABMATE architecture.

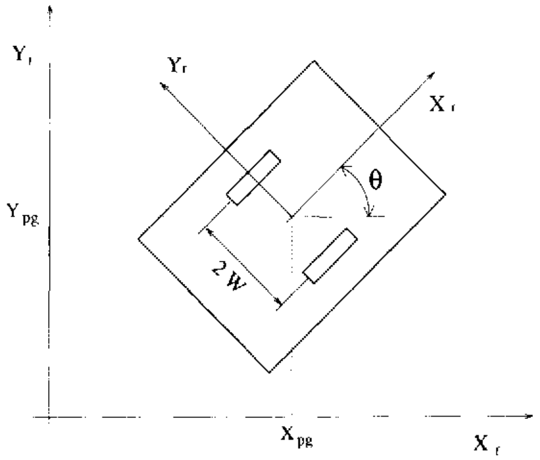


Fig. 4. Reference frame.

$$y(k+1)=y(k)-\frac{V}{\mathscr{A}}(\cos(\theta(k)+\mathscr{A}T)-\cos(\theta(k)))$$

$$\mathscr{A}=\frac{\omega_d(k-1)-\omega_i(k-1)}{2WR}$$

$$V=\frac{\omega_d(k-1)+\omega_i(k-1)}{2R}$$

where x, y, θ are the position and heading of the robot in a fixed reference frame, T is the sample interval and W is the half-distance between wheels, the value of which has been estimated to be 168 mm (Fig. 4). V is the lineal velocity of the mobile robot and $\mathscr{A}$ is the steering speed. These equations are valid in the case of a steering speed $\mathscr{A} \neq 0$. In the case of a linear trajectory, the equations of motion are given by:

$$\theta(k+1)=\theta(k)$$
$$x(k+1)=x(k)+VT\cos\theta(k)$$
$$y(k+1)=y(k)+VT\sin\theta(k).$$

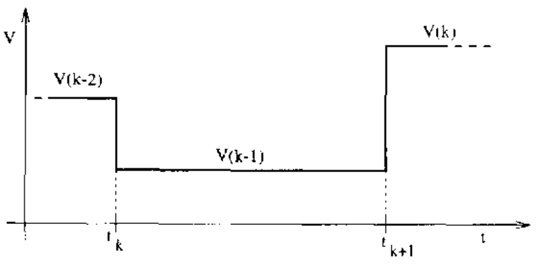


Fig. 5. Simplified dynamics of the LABMATE servomotors.

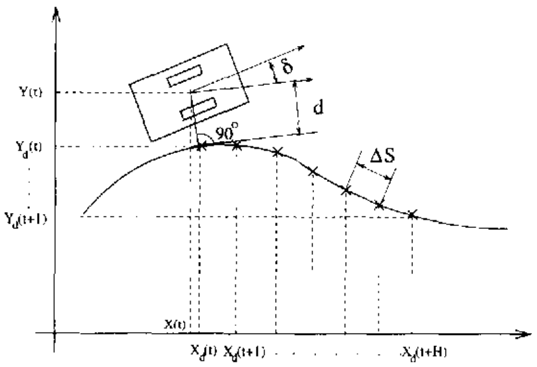


Fig. 6. Desired path parameterization.

Using the maximum acceleration value, the velocities of both wheels have been considered constants for each sample period (Fig. 5).

2.2. *Desired path parameterization*

The desired path is given to the MBPC neural network controller as a set of straight lines and circular arcs. The MBPC approach needs the desired positions and headings of the mobile platform in the next H_c time instants; so, given the current position and heading of the robot, it is necessary to parameterize the desired path for the next H_c periods of time, in order to calculate the H_c future desired path points. As is shown in Fig. 6, the desired point for the current instant $[X_d(t), Y_d(t)]$ is obtained first. It is located at the intersection between the desired path and its perpendicular, traced from the actual robot position $[X_r(t), Y_r(t)]$. The next H_c points are spaced equally on the path, with a separation between them of ΔS.

By using this approach, no approximation trajectory is needed when the robot position is not located on the desired path.

3. THE NEURAL NETWORK APPROACH

The artificial neural networks (ANNs) are parallel information process systems that can learn, from a set of training patterns, different relationships between variables regardless of their analytical dependency.[7] Once the training stage is finished, the ANN can reproduce this behaviour in real time, even for input patterns that have not been learned.

As was mentioned before, the minimization of the cost function J has to be carried out by a numerical optimization method which requires too much computation time to be used in real time. A neural network solution is proposed, which guarantees real time for the robot control. Neural network approaches for robot guidance has been proposed by other researchers[10–12,14]

There are many different ANN architectures, with different functional capabilities. The one chosen here is a multilayer perceptron,[17,18] with one hidden layer. The input layer consists of five neurons (see Fig. 7). The first two inputs correspond to the previous linear and angular velocities of the robot (these control variables are kinematically equivalent to the right and left wheel velocities). The last three inputs are associated with the parameterization of the desired trajectory over the prediction horizon. In order to reduce the number of inputs, the parameters given to the network are the distance d from the robot guide point to the path, the angle δ between the robot heading and the path orientation and an average of the inverse of the curvature of the future desired points $(1/\rho)$ (Fig. 6). The output layer consists of two nodes which correspond to the control command (the linear and angular robot velocities).

The training is done by backpropagation,[19] using the scheme shown in Fig. 8, where the training output patterns are generated by an MBPC module which uses a numerical optimization method to calculate the outputs (a Powell algorithm[20] has been used in this case as it does not require the calculation of the objective function derivate). The training patterns were selected to represent a significant variety of possible driving situations.

In order to reduce the range of parameter variation, a local reference system was used. Also, the symmetry analysis made reduces the number of training patterns

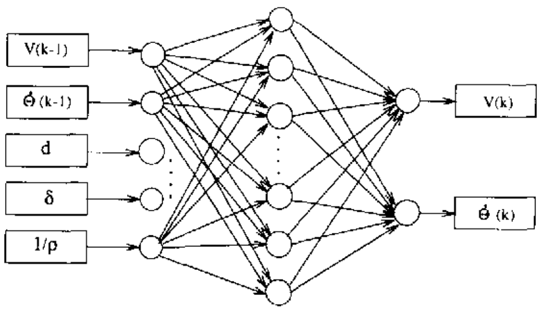


Fig. 7. Neural network scheme.

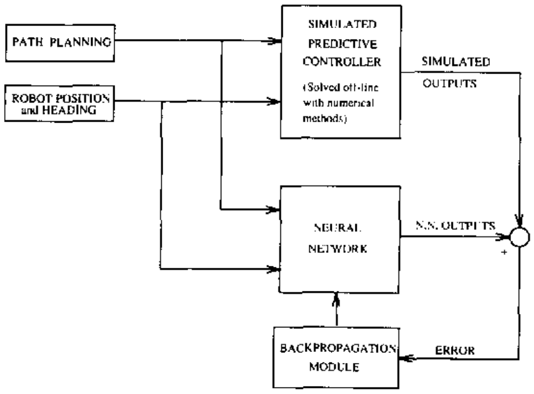

PATH PLANNING
ROBOT POSITION and HEADING
SIMULATED PREDICTIVE CONTROLLER (Solved off-line with numerical methods)
SIMULATED OUTPUTS
NEURAL NETWORK
N.N. OUTPUTS
BACKPROPAGATION MODULE
ERROR

Fig. 8. Neural network training scheme.

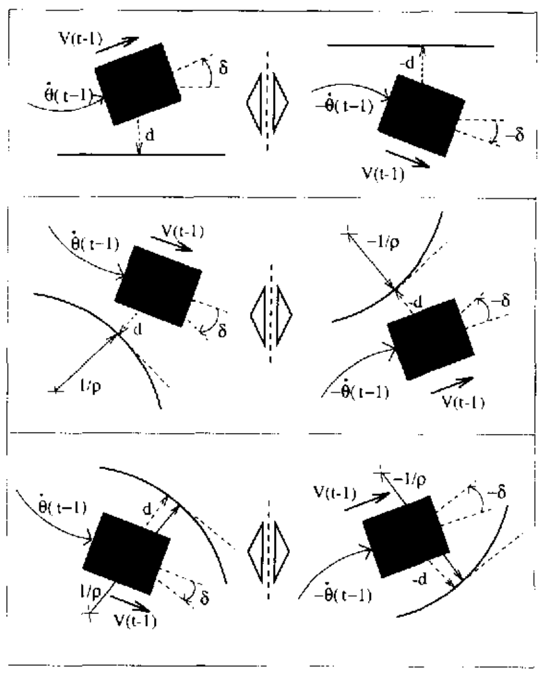


Fig. 9. Symmetry analysis.

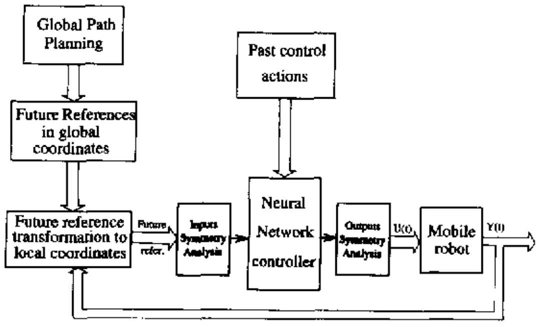

Global Path Planning
Future References in global coordinates
Future reference transformation to local coordinates
Future refer.
Inputs Symmetry Analysis
Past control actions
Neural Network controller
Outputs Symmetry Analysis
U(t)
Mobile robot
Y(t)

Fig. 10. Neural network training scheme.

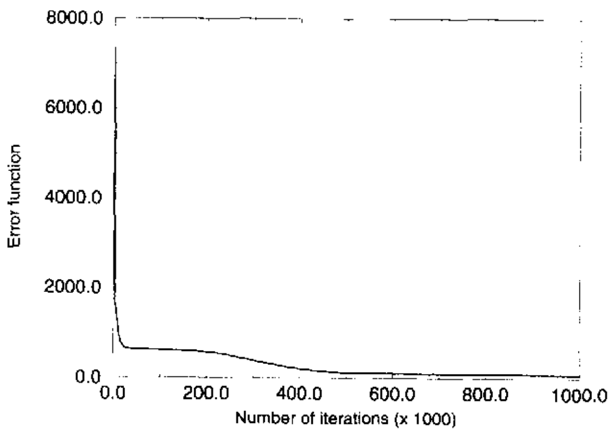
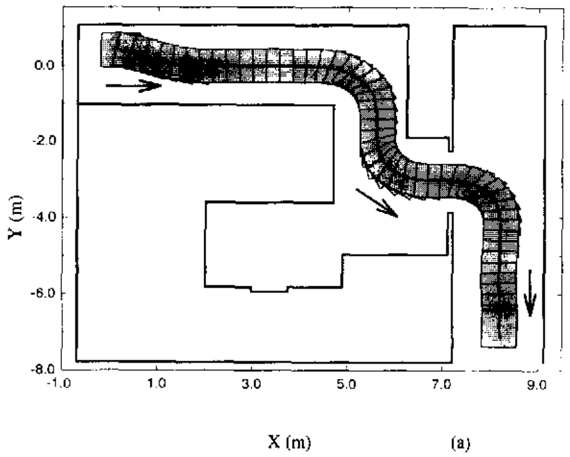


Fig. 11. Backpropagation error function.

needed to obtain good performance of the neural network.

The symmetry analysis for the inputs follows this scheme:



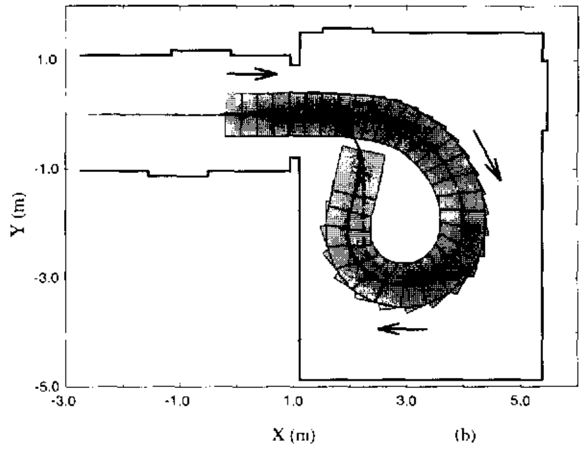
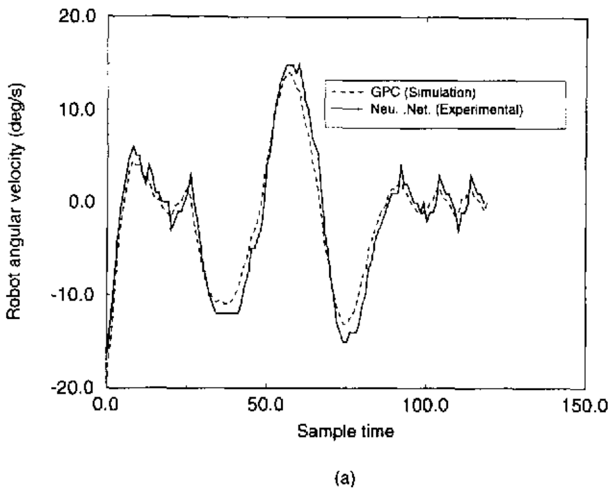


Fig. 12. Neural network experimental results.



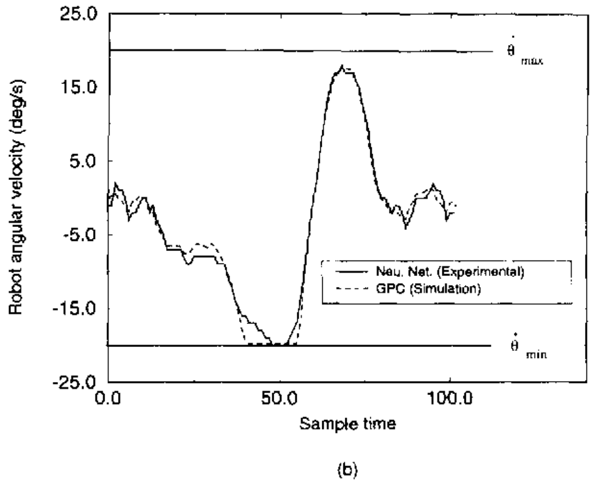


Fig. 13. Robot angular velocity.

- If $d(t) \leq 0$

$$e_1 = V(t-1)$$
$$e_2 = \dot{\theta}(t-1)$$
$$e_3 = d(t)$$
$$e_4 = \delta$$
$$e_5 = \frac{1}{\rho}$$

- Else,

$$e_1 = V(t-1)$$
$$e_2 = -\dot{\theta}(t-1)$$
$$e_3 = -d(t)$$
$$e_4 = -\delta$$
$$e_5 = -\frac{1}{\rho}$$

where e_i are the ANN inputs, and for the outputs:

- If $d(t) \leq 0$

$$V(t) = s_1(t)$$
$$\dot{\theta}(t) = s_2(t)$$

- Else,

$$V(t) = s_1(t)$$
$$\dot{\theta}(t) = -s_2(t)$$

where s_i are the ANN outputs.

The symmetry cases are shown in Fig. 9. When an input pattern is detected as being symmetrical to one learned by the ANN, the input data to the network will be those learned. Afterwards, the outputs generated by the ANN will be symmetrically changed before sending the control actions to the mobile platform. The neural network guidance system is shown in Fig. 10.

4. RESULTS

The proposed control structure has been tested with the LABMATE mobile robot. The neural network used consisted of five input neurons corresponding to the past control actions and a future reference trajectory. The hidden layer was composed of 10 neurons and the output layer consisted of two neurons corresponding to the linear and angular robot velocities.

The neural network was trained in a supervised manner, as described previously. The sampling interval T was given a value of 0.6 sec. The control horizon

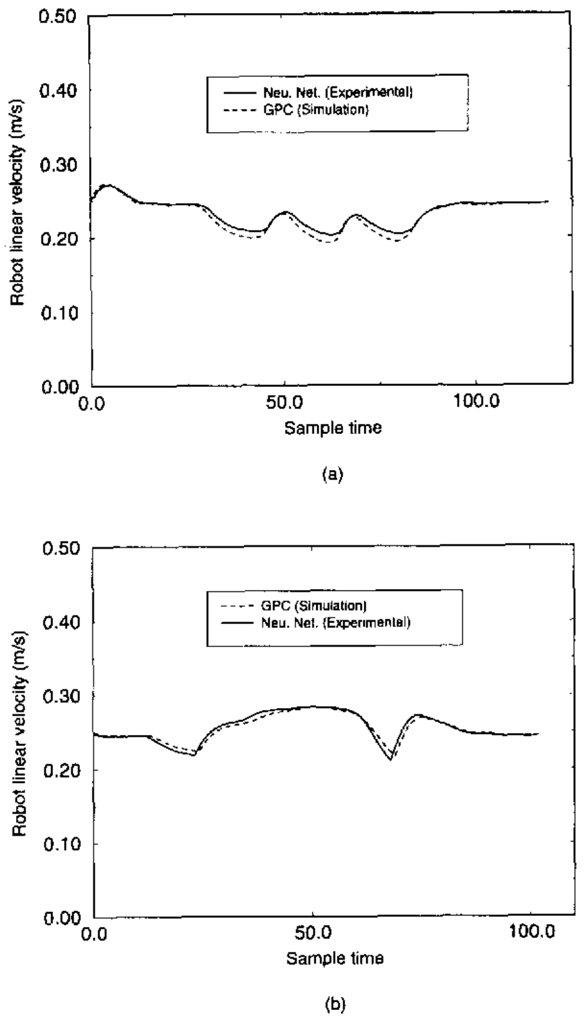


Fig. 14. Robot linear velocity.

H chosen for the MBPC was made equal to six, thus H_1 and H_2 were given the values 2 and 7, respectively, and the weighting factors were given the values $\lambda_1 = 35$ and $\lambda_2 = 5$. The maximum and minimum linear and angular velocities were given the following values respectively: 0 m/sec, 0.4 m/sec, $-20°/s$ and $20°/s$. For Δs, a value of 0.15 m was chosen, which leads to an average linear robot velocity of 0.25 m/sec. Figure 11 shows the evolution of the error function

$$E(i) = \sum_{i=1}^{N} (O_d(i) - O(i))^2$$

of the training phase, where N is the number of training patterns and $O_d(i)$ and $O(i)$ are the desired output and the network output for the iteration i.

Figure 12 shows some of the experimental results obtained in the laboratory when applying the proposed algorithm to the LABMATE mobile robot. Figure 12(a) shows the desired trajectory and the trajectory through a narrow corridor and a door followed by the robot. As can be seen, the mobile robot follows the desired trajectory in spite of being in an initial position separated about 400 mm from the desired path. Figure 12(b) shows another test for a path where small curvature radii are specified. The tracking error observed in Fig. 12(b) is due to the saturation in the angular velocity [see Fig. 13(b)] and to the penalization chosen for the control actions.

In Figs 13 and 14 the control variables for MBPC simulation using numerical methods, and the neural network outputs for the real experiment are presented. From the analysis of Fig. 13, it can be seen that the neural network experimental results are not as smooth as the MBPC simulation angular velocity. This is because only integer values for the angular velocity can be sent to the low level control processor of the LABMATE.

As expected, the neural network reproduces the behaviour of the MBPC quite well and takes only a small fraction of the computation time required for solving the MBPC which has to be solved using a numerical optimization algorithm.

5. CONCLUSIONS

A neural network-based path tracking algorithm for mobile robots has been presented. The neural network has been trained in a supervised way with a back-propagation algorithm. The desired neural network output was computed off line by a predictive controller. Control signal saturations and non-linearities of the model were considered in order to obtain accurate predictions of the robot trajectories. The computation time required to solve this MBPC problem under these circumstances would be prohibitive for real time. The neural network approach has proved to be an effective way of implementing the path tracking predictive algorithm as shown by the simulation and experimental tests.

REFERENCES

1. Abu el Ata-Doss, S. A., Fiani, P., Richalet, J.: Handling input and state constraints in predictive functional control. In *30th Conference on Decision and Control*, Brighton. 1991, pp. 985–990.
2. Camacho, E. F.: Constrained generalized predictive control. *IEEE Trans. Automatic Control* **38**: 327–332, 1993.
3. Clarke, D. W., Mohtadi, C., Tuffs, P. S.: Generalized predictive control—Part I. The basic algorithm. *Automatica* **23**: 137–148, 1987.
4. Erdmann, M., Lozano-Perez, T.: On multiple moving objects. *Algorithmica* **2**: 477–521, 1987.
5. Fujimura, K., Samet, H.: A hierarchical strategy for path planning among moving obstacles. *IEEE Trans. Robotics Automation* **5**: 61–69, 1989.
6. Gómez-Ortega, J.: Identificación del modelo dinámico del robot LABMATE. Internal Report, Sevilla University, 1993.
7. Hush, D., Horne, B. G.: Progress in supervised neural networks. *IEEE Signal Processing Magazine*, pp. 8–39, 1993.
8. Kant, K., Zucker, S. W.: Toward efficient trajectory planning: the path-velocity decomposition. *Int. J. Robotics Res.* **5**: 72–89, 1986.
9. Lozano-Perez, T.: Spatial planning: a configuration

space approach. *IEEE Trans. Comput.* **C-32**: 108–120, 1983.

10. Meng, H., Picton, P. D.: Obstacle avoidance using a neural network controller and visual feedback. In *Proceedings of SICICA 92*, Malaga. 1992, pp. 563–568.
11. Meng, M., Kak, A. C.: Mobile robot navigation using neural networks and nonmetrical environment models. *IEEE Control Systems* **13**(5): 30–39, 1993.
12. Nagata, S., Sekiguchi, M., Asakawa, K.: Mobile robot control by a structure hierarchical neural network. *IEEE Control System Magazine*, pp. 69 76, 1990.
13. Ollero, A., Amidi, O.: Predictive path tracking of mobile robots. Applications to the CMU Navlab. In *Proceedings of the IEEE Fifth International Conference on Advanced Robotics*, Pisa, 1991, Vol. 2, pp. 1081–1086.
14. Pomerlau, D. A.: Neural network based autonomous navigation. *Vision and Navigation. The Carnegie Mellon Navlab*, Thorpe, C. E. (Ed.). Dordrecht. Kluwer Academic. 1990, pp. 83–93.
15. Quero, J. M., Camacho, E. F., Franquelo, L. G.: Neural network for constrained predictive control. *IEEE Trans. Circuits Systems* **40**: 621–626, 1993.
16. Richalet, J., Rault, A., Testud, J. L., Papon, J.: Model predictive heuristic control: application to industrial processes. *Automatica* **14**: 413–428, 1978.
17. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Rev.* **65**: 386–408, 1958.
18. Rosenblatt, F.: *Principles of Neurodynamics*. Spartan. 1961.
19. Rumelhart, D. E., Hinton, G. E., Williams, R. J.: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge, MA, MIT Press. 1986, pp. 318 362.
20. Schwefel, H. P.: *Numerical Optimization of Computer Models*. Chichester, John Wiley. 1977.
21. Shiller, Z., Gwo, Y.: Dynamic motion planning of autonomous vehicles. *Trans. IEEE Robotics Automation* **7**: 241 249, 1991.
22. TRC.: *Labmate Reference Manual*. Danbury, CT, Transitions Research Corporation, 1981.