Pergamon

Control Eng. Practice, Vol. 4, No. 12, pp. 1669-1679, 1996
Copyright © 1996 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0967-0661/96 $15.00 + 0.00

PII:S0967-0661(96)00184-0

MOBILE ROBOT NAVIGATION IN A PARTIALLY STRUCTURED STATIC ENVIRONMENT, USING NEURAL PREDICTIVE CONTROL

J. Gómez Ortega and E.F. Camacho

Dpto. Ingeniería de Sistemas y Automática, Univ. de Sevilla, Escuela Superior de Ingenieros, Avda. reina Mercedes s/n, 41012 Sevilla, Spain (juango@esi.us.es)

(Received January 1996; in final form September 1996)

Abstract: This paper presents a way of implementing a model-based predictive controller (MBPC) for mobile robot navigation when unexpected static obstacles are present in the robot environment. The method uses a nonlinear model of mobile robot dynamics, and thus allows an accurate prediction of the future trajectories. An ultrasonic ranging system has been used for obstacle detection. A multilayer perceptron is used to implement the MBPC, allowing real-time implementation and also eliminating the need for high-level data sensor processing. The perceptron has been trained in a supervised manner to reproduce the MBPC behaviour. Experimental results obtained when applying the neural-network controller to a TRC Labmate mobile robot are given in the paper.

Keywords: Mobile robots, predictive control, navigation, obstacle avoidance, neural networks.

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.

If the working environment is unknown, the path planning stage has no sense. In this case, the navigation strategy is purely reactive. The inputs to the mobile robot navigator are the final goal position and the sensor system data. If there are no obstacles between the robot and its goal, the reference path is just a straight line between them. If an unpredicted obstacle is detected, some avoidance strategy is used. Potential function based methods (Khatib, 1986; Borenstein and Koren, 1991), neural networks (Pomerlau, 1990; Meng and Kak, 1993; Nagata, *et al.*, 1990), and fuzzy logic based controllers (Goodridge and Luo, 1994; Saffioti, *et al.*, 1993), trained with a heuristic database of rules, are among the possibilities.

When the working environment is structured and the uncertainty level is low, this problem is reduced to a classic control problem with a desired precalculated path as the reference. This is called the path-tracking problem. If the possibility of unexpected obstacles is considered, as it is in this paper, the problem is much more complex, also involving sensor data processing and obstacle avoidance strategies.

Two general approaches can be encountered when only fixed obstacles are considered. In the first one, the mobile robot motion control and the obstacle avoidance problems are considered decoupled, using a two-level hierarchical strategy. The higher level replans a new reference path when an obstacle is encountered. The lower level drives the robot to follow the actual reference path, which can be the original path or the replanned path.

Sinha, *et al.* (1992), Lin, *et al.* (1994) and Foux, *et al.* (1993) used this kind of strategy.

In the second approach, these two problems are coupled. The mobile robot motion controller is fed with sensor data from the environment. This information is considered in the computation of the control law. In this case, there is no geometrical path replanning. These methods are often based on optimal control techniques, where the control law is obtained from the minimization of a cost function, where the proximity to any obstacle is also penalized (Feng and Krogh, 1991).

The problem that is raised in this paper is that of driving a mobile robot to follow a previously calculated desired path, avoiding unexpected fixed obstacles. As the desired future reference is known, it would seem that a predictive control technique is a suitable approach for this problem.

Model-based predictive control (MBPC) methods are a family of optimal control techniques that are characterized by the following common elements: a) previous knowledge of future references; b) explicit use of a system model for future system output prediction; c) minimization of a cost function to obtain the control law; d) a receding-horizon strategy.

Some of the advantages of the MBPC are: simplicity of the problem formulation which does not require complex mathematical developments; easy controller tuning, due to the intuitive definition of the cost function; easy extension to MIMO systems; intrinsic delay compensation; the possibility of considering constraints on the inputs and on the outputs.

The use of MBPC strategies for solving the path tracking and the navigation problems in a partially structured environment has been referenced in the literature (Papageorgiou and Steinkogler, 1993; Gómez Ortega and Camacho, 1994).

In this paper, a predictive strategy is used for the navigation module. A nonlinear model of the robot kinematics is used. Constraints in the control variables are also considered and a non quadratic cost function is proposed for computing the control signals. A neural-network scheme is presented for the implementation of this complex predictive controller to achieve real-time performance. The neural network will be trained in a supervised manner with training patterns obtained from an off-line simulation of a predictive controller, computed with numerical algorithms. On the other hand, the sensor data-processing problem is also solved in real time with the neural network. A ring of sonars is used for obstacle detection and the measurements obtained from each transducer are fed directly to the neural network, which has been trained off-line with a model of the sonar performance. This strategy avoids the computing cost of high-level processing of sensor data, and allows real-time performance for the navigation system.

2. MBPC TECHNIQUES FOR MOBILE ROBOT NAVIGATION

2.1 *MBPC strategy*

In general, most MBPC methods are based on a common performance scheme, shown in Fig. 1 (for a SISO system).

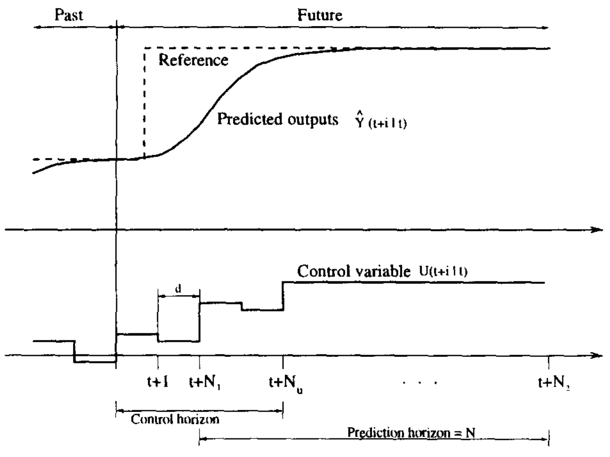


Fig. 1. MBPC strategy.

A cost function J is defined, where J is often a quadratic function of the sum of the future errors in reference tracking, $\hat{e}(k+i|k) = \hat{Y}(k+i|k) - Y_d(k+i)$, predicted over a *prediction horizon* $N = N_2 - N_1$, and also of the sum of the increments of the control actions foreseen for the future, $\Delta U(k+i|k)$, over a *control horizon* N_u:

$$J(N_1, N_2, N_u) = E\{\sum_{i=N_1}^{N_2} \mu(i)[\hat{Y}(k+i|k) - Y_d(k+i)]^2 + \sum_{i=1}^{N_u} \lambda(i)[\Delta U(k+i-1|k)]^2\}$$

The notation $x(k+i|k)$ indicates that $x(k+i)$ is calculated with the data known at sample time k. $\mu(i)$ and $\lambda(i)$ are penalty factors, which are usually chosen to be constant along the time. The future system outputs, $\hat{Y}(k+i|k)$ for $i = N_1, ..., N_2$, are predicted from a model of the process, from the inputs and outputs before instant k, and from the control actions foreseen for the future, $U(k+i|k)$ for i=0,...,$N_u - 1$, which are the unknown variables. In this way, J can be expressed as a function of only the future control actions. It is usual to suppose that the control actions are constant after a predefined time instant.

The objective of predictive control is to obtain a future control action sequence $(U(k), U(k+1|k), ..., U(k+N_u-1|k))$ in such a way that the future predicted outputs $\hat{Y}(k+i|k)$ will be as close as possible to the desired references $Y_d(k+i)$ over the prediction horizon. This is accomplished by the minimization of J with respect to the control variables. After this sequence is obtained, a *receding horizon* approach is considered. This consists of applying only the first control action $U(k)$ calculated. This process is repeated at every sampling interval in such a way that the calculated open-loop control law is applied in a closed-loop manner.

The problem raised in this paper (see Fig. 2) is that of driving a mobile robot to follow a previously calculated desired path (taking only known obstacles into account) avoiding the unexpected static obstacles. A predictive control technique is used as the robot navigator.

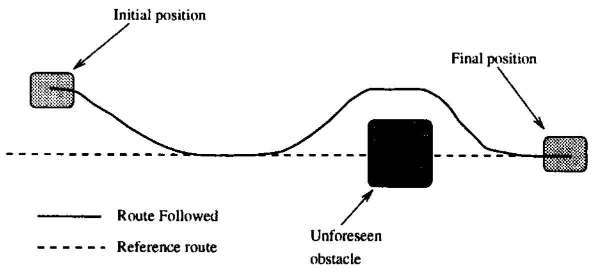


Fig. 2. Navigation problem definition.

The cost function used here is:

$$J(N_1, N_2, N_u) = \sum_{i=N_1}^{N_2} [\hat{Y}(k+i|k) - Y_d(k+i)]^2$$
$$+ \sum_{i=1}^{N_u} \lambda_1([\Delta\omega_r(k+i-1)]^2 + [\Delta\omega_l(k+i-1)]^2)$$
$$+ \sum_{i=1}^{N_u} \lambda_2[\omega_r(k+i-1) - \omega_l(k+i-1)]^2$$
$$+ \sum_{j=1}^{NFO} (\sum_{i=N_1}^{N_2} \frac{\psi}{[\text{dist}_f(\hat{Y}(k+i|k), FO_j)]^2})$$

where $\hat{Y}(k+i|k) = \{\hat{x}(k+i|k), \hat{y}(k+i|k)\}$ is an i-step prediction of the robot position made at instant k, 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 and ψ are constant weighting factors. The first term in J penalizes the position error; the second term penalizes the acceleration and the third penalizes the robot's angular velocity. These last two terms ensure smooth robot guidance. An error in the robot heading could be considered in J, but it has been noticed that this is not necessary when the control horizon N_u is sufficiently large.

The last term penalizes the proximity between the robot and the obstacles, which are detected with an ultrasonic proximity system placed on-board the mobile robot. NFO is the number of fixed obstacles detected in the environment. This is a potential function term, where $dist_f(\cdot)$ is a measurement in $k+i$ of the distance between the robot and a fixed obstacle FO_j, which is considered to have a polygonal or circular geometry in the plane. A more precise description of this function is presented below. A block diagram of the system is shown in Fig. 3.

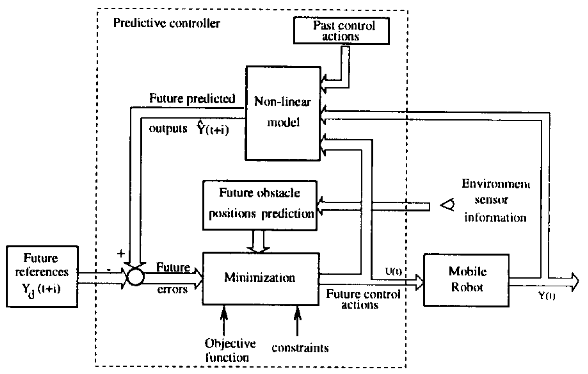


Fig. 3. The predictive controller scheme.

Notice that the minimum output horizon N_1 should be set to a value greater than the dead time d of the system, since the output for smaller time horizons cannot be affected by the first action $(\omega_r(k), \omega_l(k))$. In what follows, N_1 and N_2 will be considered to be $N_1 = d+1$ and $N_2 = N$, and N_u will be given a value of $N_2 - d$. In this formulation it is assumed that after the control horizon N_u, further increments in control are zero. So the controller has only one free parameter, N.

For the selection of the weighting factors λ_1, λ_2 and ψ some heuristic has to be considered. The value of λ_1 has to do with the robot's acceleration. If the value chosen for it is not large enough, and considering that the cost function is evaluated only at the sampling instants, the solution for avoiding a fixed obstacle could consist of greatly increasing the velocity for one sampling period. This control action would drive the robot far enough from the obstacle at instants k and $k+1$, but the path followed by the robot between these two instants would collide with the obstacle. This can be seen in Fig. 4. Obviously, this strategy is not acceptable. The selection of λ_2 depends on the smoothness desired for the mobile robot's motion. As this factor has to do with the robot's angular velocity, the higher the value of λ_2, the smoother the path followed by the robot. Finally, ψ is a penalty factor for a potential-like function. Its value should not be too large because it has a strong influence on the space left free for the mobile robot's motions. The greater the factor, the higher the values of the potential function. In an extreme case, the free space between two closely located obstacles may be eliminated because of a high value of the potential function.

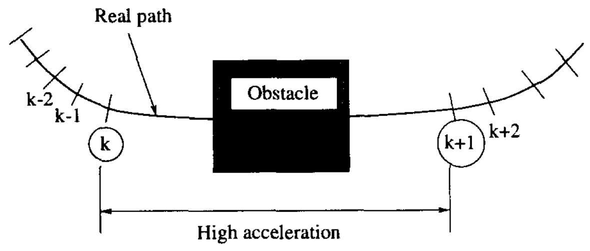


Fig. 4. λ_1 penalty factor selection.

The predictive problem, formulated under these circumstances, has to be solved using 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 implementation.

2.2 *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* mobile robot (TRC, 1989) has been used (Fig. 5).

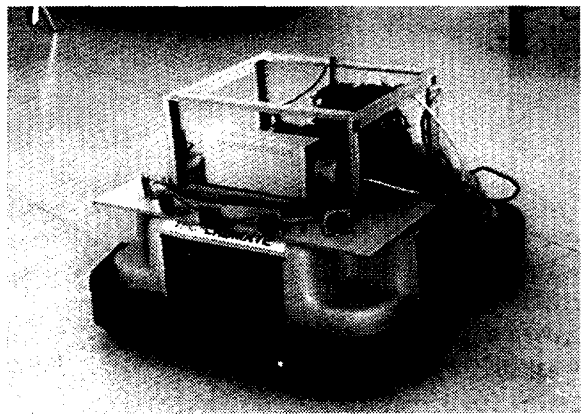


Fig. 5. The *LABMATE* mobile robot.

The mobile robot has been designed for indoor environments. Its dimensions are 0.8 x 0.8 m, with a weight of 50 kg (without batteries). The carrying capacity for payload is up to 90 kg. It provides the mechanical base, a two-wheel differential driving system (supported by 4 passive castors), as well as the servocontrol of the motors. The maximum attainable linear velocity is 1000 mm/s. An RS-232 radiomodem interface (of 915 Mhz) is used to communicate between the host (where the MBPC neural-network algorithm is executed) and

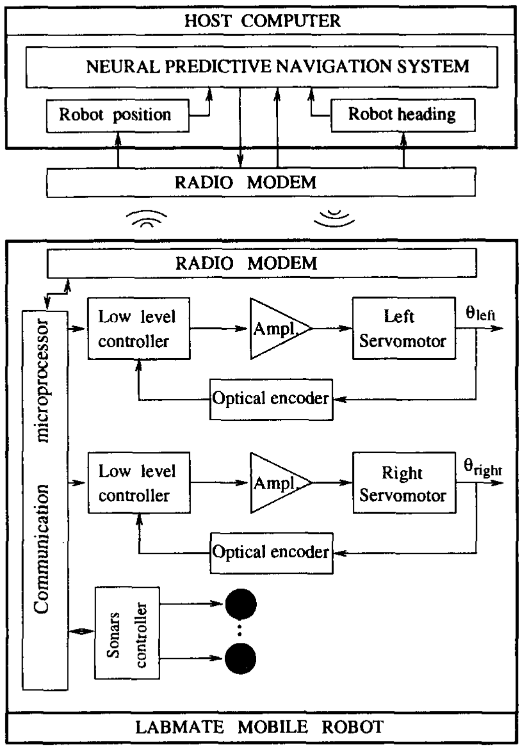


Fig. 6. *LABMATE* architecture.

the low-level control microprocessor. The robot architecture is shown in Fig. 6.

A model of the *LABMATE* mobile robot which takes account of 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 (Gómez Ortega, 1994).

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

$$\begin{aligned}
\theta(k+1) &= \theta(k) + \dot{\theta}(k-1)T \\
x(k+1) &= x(k) + \frac{V(k-1)}{\dot{\theta}(k-1)}\{\sin(\theta(k) + \dot{\theta}(k-1)T) \\
&\quad - \sin\theta(k)\} \\
y(k+1) &= y(k) - \frac{V(k-1)}{\dot{\theta}(k-1)}\{\cos(\theta(k) + \dot{\theta}(k-1)T) \\
&\quad - \cos\theta(k)\} \\
\dot{\theta}(k-1) &= R\,\frac{\omega_r(k-1) - \omega_l(k-1)}{2W} \\
V(k-1) &= R\,\frac{\omega_r(k-1) + \omega_l(k-1)}{2}
\end{aligned}$$

where x, y, θ are the position and heading of the robot in a fixed reference frame (see Fig. 7), T is the sample interval and W is the half-distance between the wheels, which value has been estimated to be 168 mm (Fig. 7). V is the linear velocity of the mobile robot, $\dot{\theta}$ is the steering speed, and $\omega_r(k-1), \omega_l(k-1)$ and R are the right and left wheel angular velocities (which are considered to be constant for each sample interval) and the wheel radius, respectively. These equations are valid in the case of a steering speed $\dot{\theta}(k-1) \neq 0$. In the case of a linear trajectory, the equations of motion are given by:

$$\begin{aligned}\theta(k+1) &= \theta(k)\\ x(k+1) &= x(k) + VT\cos\theta(k)\\ y(k+1) &= y(k) + VT\sin\theta(k)\end{aligned}$$

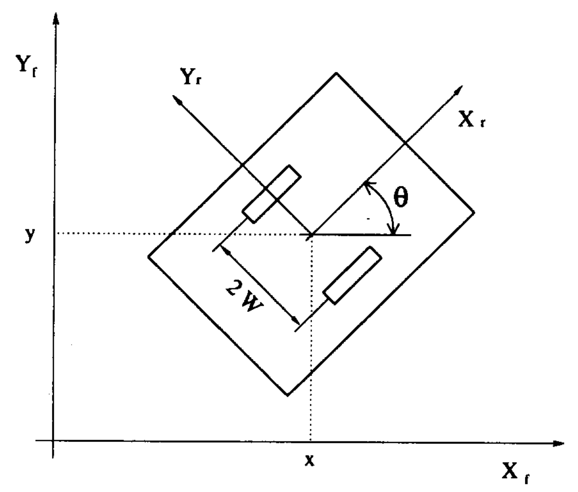


Fig. 7. Reference frame

Using the maximum acceleration value, the velocities of both wheels have been considered to be constant for each sample period (see Fig. 8).

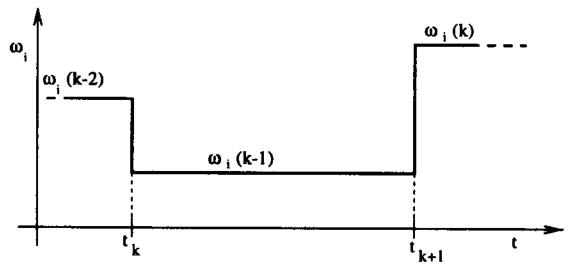


Fig. 8. Simplified dynamics of the *LABMATE* servomotors

2.3 *Desired path parametrization*

The reference 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 at the next N time instants. So, given the current position and heading of the robot, it is necessary to parametrize the desired path for the next N periods of time, in order to calculate the N future path points desired. As is shown in Fig. 9, the desired point for the current instant $(x_d(k), y_d(k))$ is obtained first. It is located at the intersection between the desired path and its perpendicular, traced from the actual robot position $(x(k), y(k))$. The next N points are spaced equally along the path, with a separation between them of ΔS, which is a design parameter. By using this approach, no approximation trajectory is needed when the robot position is not located on the desired path.

The mobile robot configuration (position and heading) has been computed at each sampling time using an odometry algorithm, based on internal sensors (incremental encoders) of the *LABMATE* mobile robot. This is not an error-free approach, and the robot's configuration estimation errors are accumulative. Thus, for long paths, the odometry system should be updated with data from an external sensor system (video camera, range laser, etc.), with a predefined frequency. In this paper, this updating procedure has not been accomplished because the reference paths used in the experiments were short (less than thirteen meters).

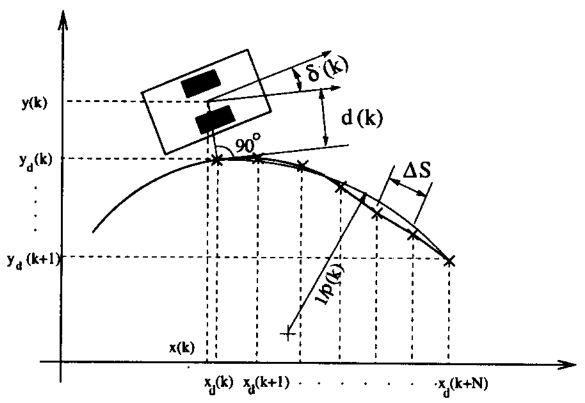


Fig. 9. Desired path parametrization.

2.4 *Potential function for fixed obstacles*

As stated above, the fixed obstacles are considered to have a circular or polygonal geometry in the plane, and the surfaces of the obstacles are considered to be perpendicular to the moving plane of the mobile robot. For a circular geometry, the potential function is calculated as the inverse of the squared distance between the robot's position $\hat{Y}(k+i|k)$ and the obstacle's position Y_{fo}, defined as the center of the circumference. This distance is decreased by the obstacle radius R_{fo} in order to consider its dimension:

$$p_{cir} = \frac{}{\|\hat{Y}(k+i|k) - Y_{fo}\|^2 - R_{fo}}$$

Two different potential functions have been used for the polygonal geometry case; one for convex polygons and another for concave polygons.

Potential function for a convex polygon (Hwang and Ahuja, 1992). A convex region will be described by a set of inequalities

$$g(x) \leq 0, \quad g \in L^m, x \in \mathbb{R}^n$$

where L is the set of linear functions and n is the space dimension (in this case n=2). The function

$$f(x) = \sum_{i=1}^{NSF} g_i(x) + |g_i(x)|$$

is zero inside the convex region and increases linearly outside of it, as the distance to the boundary increases. NSF is the number of segments of the obstacle boundary. The following potential function is used:

$$p_{cvx}(x) = \frac{1}{\delta + \sum_{i=1}^{NSF} (g_i(x) + |g_i(x)|)}$$

where δ is a small constant that limits the value of $p_{cvx}(x)$ inside the convex region. $p_{cvx}(x)$ reaches its maximum value δ^{-1} inside the region occupied by the obstacle, and decreases with the distance between the robot and the obstacle. A graphic example of this function is shown in Fig. 10, where two rectangular-shaped static obstacles are present in the proximity of the robot.

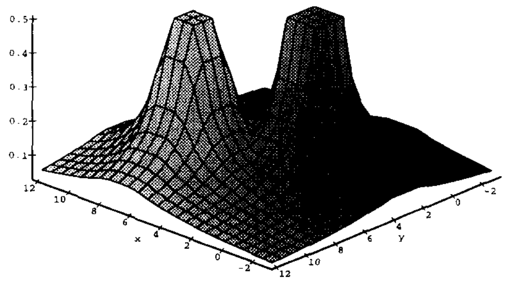


Fig. 10. Potential function of convex regions.

Potential function for a concave polygon. A concave region will be described by a set of inequalities

$$v(x) \geq 0, \quad v \in L^m, x \in \mathbb{R}^n$$

The potential function used in this case is

$$p_{ccv}(x) = \frac{1}{\delta + g_{ccv}(x)}$$

where δ is a small constant and $g_{ccv}(x)$ is the minimum of the distances between the robot position and every straight line that defines the obstacle boundary. This function, which is graphically shown in Fig. 11 for a rectangular-shaped obstacle, has the same characteristics as $p_{ccx}(x)$.

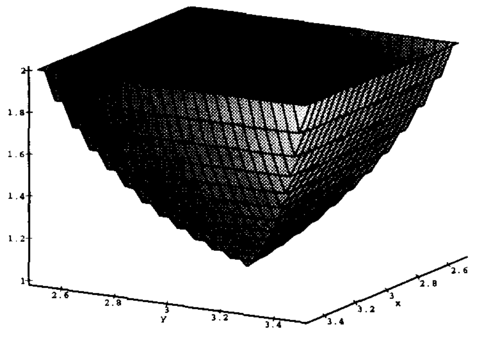


Fig. 11. Potential function of concave regions.

An example of cost function J, for a one-step prediction horizon (which is the only one that can be graphically represented), is shown in Fig. 12. In this Fig. the value of J for different left- and right-wheel velocities is represented. The existence of a rectangular obstacle can be noticed. Furthermore, the influence of the nonlinear prediction model can be observed.

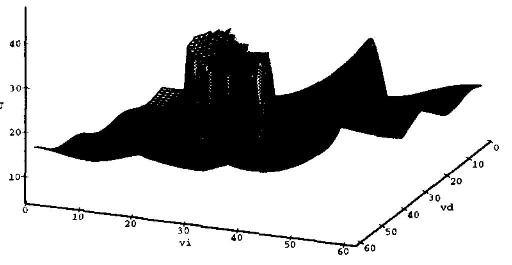


Fig. 12. Objective function J.

3. THE NEURAL PREDICTIVE APPROACH

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 implementation of the robot controller. Artificial neural networks (ANN) are parallel information-processing systems that can learn, from a set of training patterns, different relationships between variables regardless of their analytical dependency (Hush and Horne, 1993). Once the training stage is over, an ANN can reproduce this behaviour in real time, even for input patterns that have not been learned. Neural-network approaches for robot guidance have been proposed by other researchers (Pomerlau, 1990; Meng and Picton, 1992; Meng and Kak, 1993; Nagata, *et al.*, 1990).

The modules of the control scheme used in this work (see Fig. 13) are:

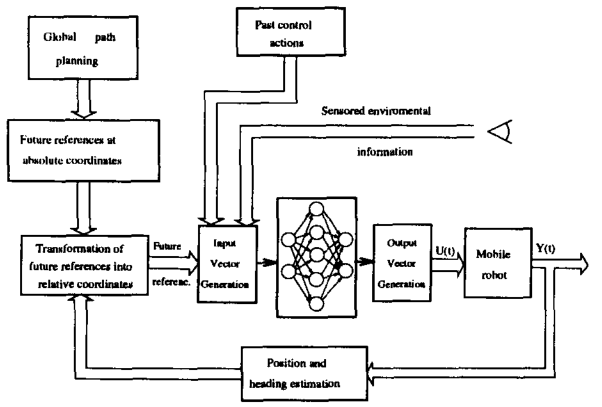

Global path planning
Past control actions
Sensored enviromental information
Future references at absolute coordinates
Transformation of future references into relative coordinates
Future referenc.
Input Vector Generation
Output Vector Generation
U(t)
Mobile robot
Y(t)
Position and heading estimation

Fig. 13. Predictive neural-network scheme for mobile robot navigation.

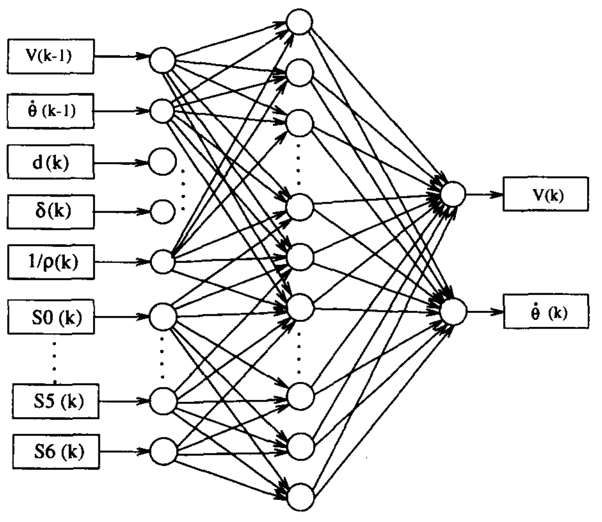


Fig. 14. Neural-network scheme

Artificial neural network controller. The ANN architecture chosen here is a multilayer perceptron (Rosenblatt, 1958), with one hidden layer (see Fig. 14).

The input layer consists of twelve neurons. The first two inputs correspond to the previous linear and angular velocities of the robot. The next three inputs are associated with the parametrization 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 $(1/\rho)$ (see Fig. 9). The last seven inputs correspond to the distances measured directly by the ring of sonar sensors. This fact avoids the high-level processing that usually has to be carried out with sensor data in order to provide useful environmental information, which is a difficult and high-computation, time-consuming phase. Thus, this architecture belongs to the *dynamic controller* class which is characterized by the inclusion of the sensor data in the control loop. This leads to a high reduction of the computation time, thus allowing real-time implementation.

The output layer consists of two nodes which correspond to the control commands (the linear and angular robot velocities, which are kinematically equivalent to the right and left wheel velocities).

Input vector-generation module. The main function of this module is to compute the values for the input layer of the ANN from different sources (the local reference path-generation module, the past control actions, and the sonar range system). Two main processing steps are applied at this stage. First, a normalization of the input values is made, in such a way that all the input neurons have the same input range. This will lead to better performance at the ANN training stage. Also, a symmetry analysis is made here, in order to reduce the number of training patterns needed to provide good performance of the neural-network controller (Gómez Ortega and Camacho, 1994).

The symmetry analysis for the inputs follows this scheme:

```
• if $d(k) \leq 0$
      $e_1 = V(k-1)$
      $e_2 = \dot{\theta}(k-1)$
      $e_3 = d(k)$
      $e_4 = \delta(k)$
      $e_5 = \frac{1}{\rho(k)}$
      for $i = 6, ..., 12$
          $e_i = sonar \ \ measurement_{i-5}(k)$
      endfor
• else,
      $e_1 = V(k-1)$
      $e_2 = -\dot{\theta}(k-1)$
      $e_3 = -d(t)$
      $e_4 = -\delta(k)$
      $e_5 = -\frac{1}{\rho(k)}$
      for $i = 6, ..., 12$
          $e_i = sonar \ \ measurement_{13-i}(k)$
      endfor
```

where e_i are the ANN inputs.

The symmetry cases for the reference path parameters are shown in Fig. 15. When an input pattern is symmetrical to one of the learned patterns, the input data to the network is set to the learned pattern. Afterwards, the outputs generated by the ANN will be symmetrically changed before sending the control actions to the mobile platform.

Output vector-generation module. The objective of this module is to perform the inverse symmetry transformation, when required. The algorithm is as follows:

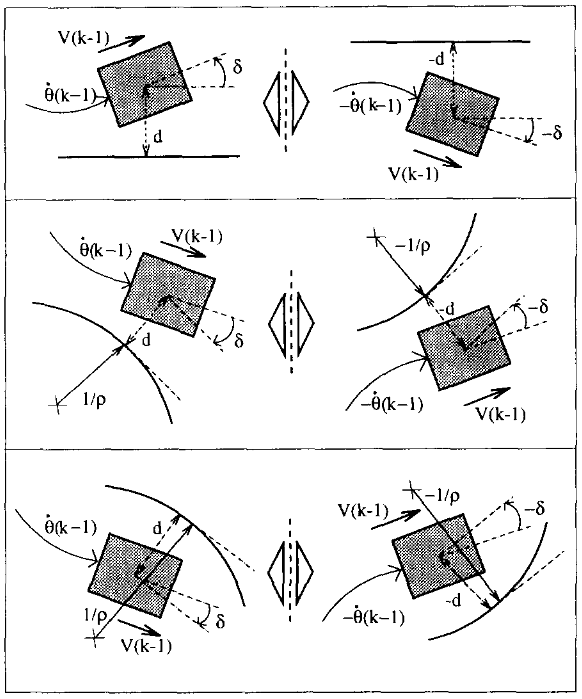


Fig. 15. Symmetry analysis.

- 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.

Reference-path coordinate transformation module. The desired path coordinates are transformed from a global reference system to a local reference system, attached to the mobile robot. This avoids the use of additional ANN inputs for the robot position and heading, which are implicitly given to the ANN in the reference path.

Past control actions. These are needed for the ANN to consider the delay time of the robot system.

Sonar range measurements. These measurements are directly used as inputs for the ANN. The ANN learns from the input patterns set, so there is no need for high-level sonar-measurement preprocessing, which guarantees real-time implementation.

3.1 *Training phase*

The ANN controller has been trained using a classic supervised training scheme, the backpropagation algorithm (Rumelhart, *et al.*, 1986) (see Fig. 16).

A set of 456 patterns has been used for training the ANN. Notice that this represents about 900

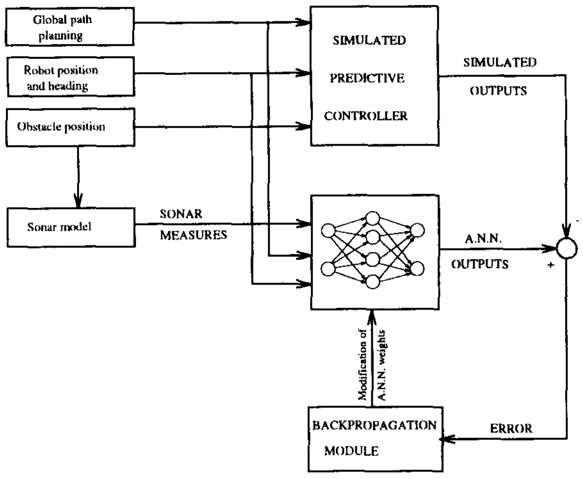
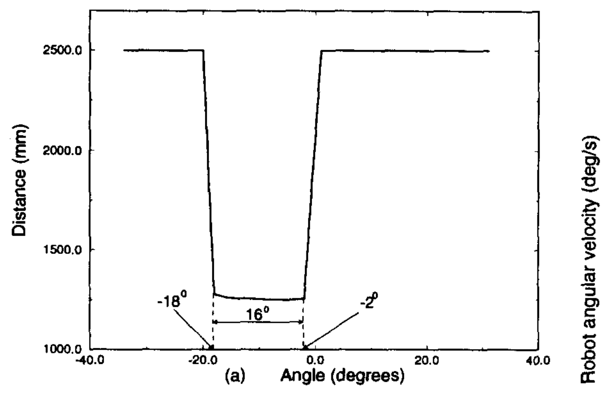


Fig. 16. ANN supervised training scheme.

patterns when the symmetrical situations are considered. The patterns were chosen locating obstacles in front and at both sides of the path, using different curvature radii for the desired paths. The desired output of the ANN was obtained with a predictive controller using a numerical algorithm and taking into account constraints on the control variables. For the minimization phase of the predictive controller, a *Powell* iterative algorithm (Press, *et al.*, 1988) has been used.

The sonar system measurements have been simulated in the training phase. The objects in the environment were described as a set of geometric primitives such as planes, cylinders, edges and corners (Leonard and Durrant-Whyte, 1992). This model also takes into account the sonar's physical parameters such as angular resolution of the transducers (the sonars used have a dispersion angle of 11 degrees), the *visibility angle*, which is defined as the maximum angle between the normal of the detected object's surface and the direction of the sonar beam, and the performance of the object's surface (two different kinds of surfaces are considered: reflecting surfaces and diffracting surfaces). The visibility angle depends not only on the sensor transducer, but also on the kind of surface. Thus, these values have to be experimentally estimated for each kind of geometrical primitive considered in the model (in Fig. 17, examples of visibility angle experimental calculations for a cylinder and an edge are shown).

In Figures 19 and 20, the control variables for MBPC simulation using numerical methods, and the neural-network outputs for the real experiment shown in Fig. 18 are presented. From the analysis of Figure 19, 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*.



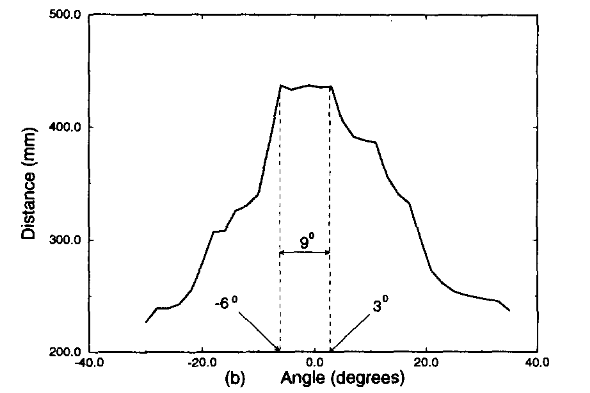


Fig. 17. Visibility angle for a cylinder (a) and a corner (b).

As expected, the NN 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.

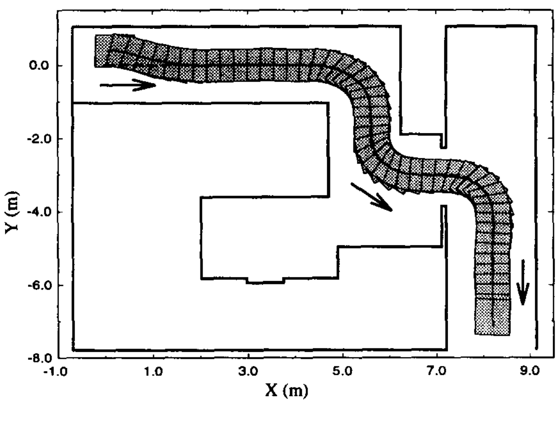


Fig. 18. Neural-network experimental results.

4. RESULTS

The proposed control structure has been tested with the *LABMATE* mobile robot. The ANN used consisted of twelve input neurons, corresponding to the past control actions, to a future reference trajectory and to the sonar range measurements.

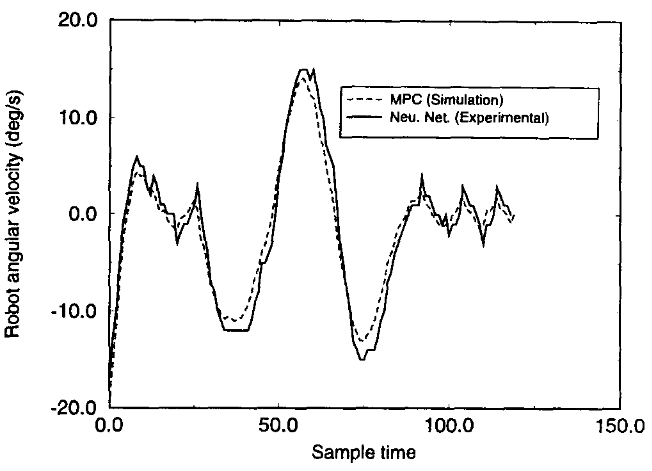


Fig. 19. Robot angular velocity

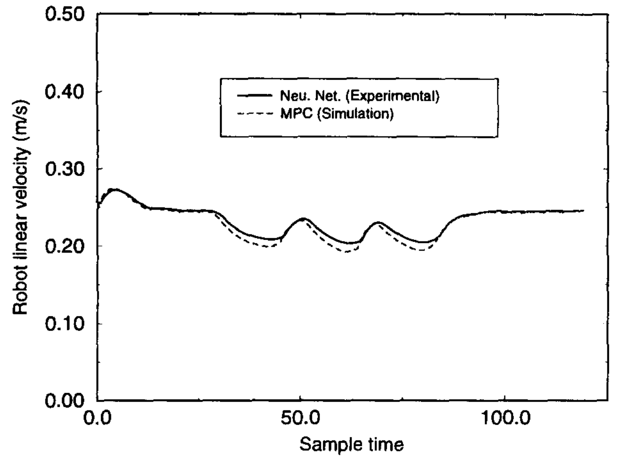


Fig. 20. Robot linear velocity.

The hidden layer was composed of eighteen neurons, and the output layer consisted of two neurons corresponding to the linear and angular robot velocities. The ANN was trained in a supervised manner, as described previously.

The value of N chosen for the MBPC was experimentally made equal to seven; thus N_1, N_2 and N_u were given the values 2, 7 and 6, respectively, and the weighting factors were given the following values: $\lambda_1 = 35$, $\lambda_2 = 5$ and $\psi = 0.5$. The maximum and minimum linear and angular velocities were given the following values respectively: 0 m/s, 0.4 m/s, -20 $^o/s$ and 20 $^o/s$. For Δs, a value of 0.15 m was chosen, which leads to an average linear robot velocity of 0.25 m/s.

The main source of spurious detections, when using ultrasonic sensors, is the *crosstalk* effect (Kortenkamp, *et al.*, 1993), which is significant for the *LABMATE* because several sensors are transmitting and receiving ultrasound signals at the same time. In order to eliminate these measurements, a filter was used, based on a strategy called *Eeruf* (error-eliminating rapid ultrasonic noise) (Kortenkamp, *et al.*, 1993). The main idea of this

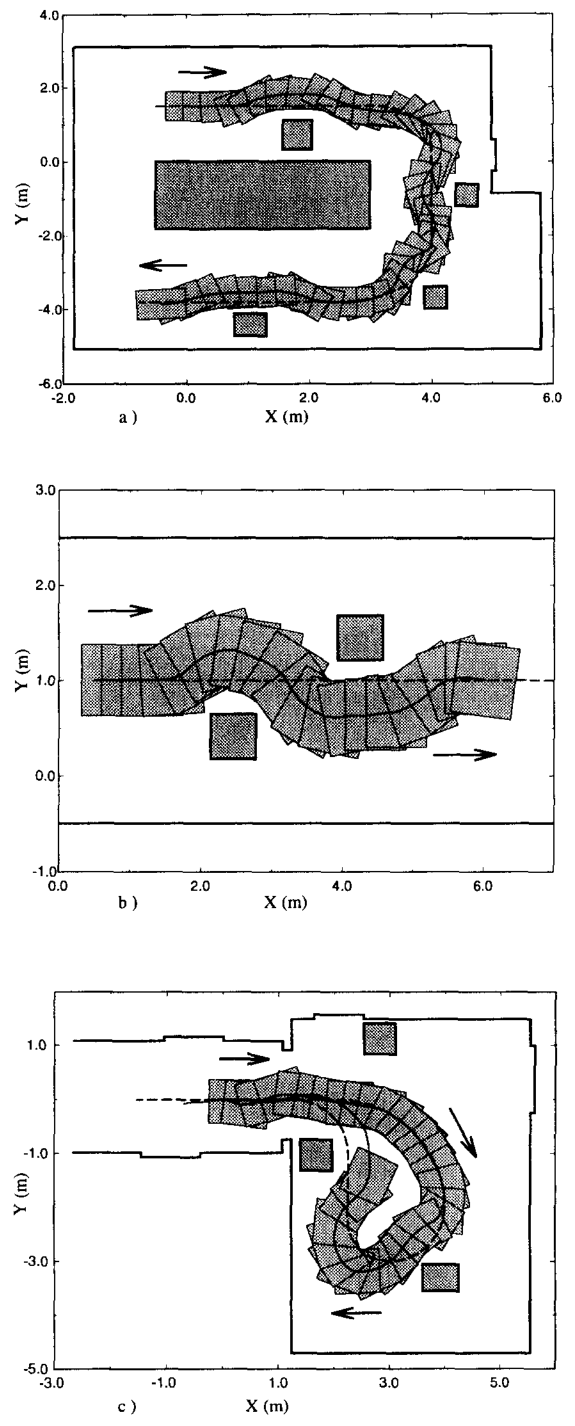


Fig. 21. Experimental results of the neural predictive controller for mobile robot navigation.

strategy is that the final value of a particular transducer is computed from a set of consecutive measurements obtained from the same sensor. As the *crosstalk* noise is systematic, a simple processing, such as a median filtering, will not reject a spurious detection. In order to make this noise random, the time between different sensor fires is chosen randomly. A wider description of the sonar model and practical aspects of the filter can be found in (Gómez Ortega, 1994).

Figure 21 shows some of the experimental results obtained in the laboratory when applying the proposed algorithm to the *LABMATE* mobile robot. In the figures, the dotted lines represent the reference paths and the solid lines the real paths. Although it is drawn as straight lines in the figures, the real environment includes laboratory objects such as chairs, tables, etc. This shows the robustness of the neural controller, as it has been trained with only a set of geometric primitives such as planes, cylinders, etc.

Figure 21(a) shows the desired trajectory and the real trajectory through the laboratory where unexpected static obstacles have been positioned. It is important to notice how the mobile robot returns to the reference path after an unexpected obstacle has been avoided.

Figure 21(b) shows an experiment where an unexpected obstacle is situated in the path that the robot must follow in order to avoid a previous unexpected obstacle. Again, the controller performance is quite good. Finally, Fig. 21(c) shows another test for a path where small curvature radii are specified. The tracking error observed in Fig. 21(c) is due to saturation in the angular velocity, and to the penalty chosen for the control actions.

5. CONCLUSIONS

A predictive ANN controller for mobile robot navigation in partially structured static environments has been presented. The navigator includes tracking of a precalculated reference path and a potential-function-like obstacle-avoidance strategy. A sonar range system has been used for obstacle detection. Practical implementation aspects have been accomplished.

The ANN has been trained in a supervised way, using a backpropagation algorithm. The desired ANN output was computed off-line by a predictive controller using a simulation model for the sonar measurements. A set of training patterns has been obtained, placing the robot heuristically in different situations. Control signal saturations and nonlinearities of the model were also considered, in order to obtain accurate predictions of the robot trajectories. The computation time required to solve this MBPC problem under these circumstances, using numerical methods, would be prohibitive for real-time implementation. The neural-network approach has proved to be an effective way of implementing the navigation predictive algorithm as shown by experimental tests.

The predictive control strategy allows the use of previously computed reference paths and avoids some of the problems found with reactive methods, which only take into account local data of the environment. This approach has also the great advantage of processing the sensor data in real time, feeding the sonar readings directly to the neural-network input, avoiding a high-level data processing.

6. ACKNOWLEDGEMENT

The authors would like to acknowledge the *CICYT* for funding this work.

7. REFERENCES

Borenstein, J. and Y. Koren (1991). Histogramic In-Motion Mapping for Mobile Robot Obstacle Avoidance. *IEEE Journal of Robotics and Automation* **7**(4), 535–539.

Feng, D. and B. H. Krogh (1991). Dynamic Steering Control of Conventionally Steered Mobile Robots. *Journal of Robotics Systems* **8**(5), 699–721.

Foux, G., M. Heymann and A. Bruckstein (1993). Two-Dimensional Robot Navigation Among Unknown Stationary Polygonal Obstacles. *IEEE Transactoins on Robotics and Automation* **9**(1), 96–102.

Gómez Ortega, J. (1994). Navegación en obots móviles basada en técnicas de control predictivo neuronal. PhD thesis. Univ. of Seville.

Gómez Ortega, J. and E. F. Camacho (1994). Neural Network MBPC for Mobile Robots Path Tracking. *Robotics and Computer Integrated Manufacturing Journal* **11**(4), 271–278.

Goodridge, S. G. and R. C. Luo (1994). Fuzzy Behavior Fusion for Reactive Control of an Autonomous Mobile Robot: MARGE. In: *Proceedings of the IEEE International Conference on Robotics and Automation.* San Diego (California). pp. 1622–1627.

Hush, D. and B. G. Horne (1993). Progress in Supervised Neural Networks. *IEEE Signal Processing Magazine* pp. 8–39.

Hwang, Y. K. and N. Ahuja (1992). A Potential Field Approach to Path Planning. *IEEE Transactions on Robotics and Automation* **8**(1), 23–32.

Khatib, O. (1986). Real-time Obstacle Avoidance for Manipulators and Mobile Robots. *The Int. Journal of Robotics Research* **5**(1), 90–98.

Kortenkamp, D., M. Huber and C. Cohen (1993). Integrated Mobile-Robot Design. *IEEE Expert* pp. 61–73.

Leonard, J. L. and H. F. Durrant-Whyte (1992). *Directed Sonar Sensing for Mobile Robot Navigation.* Kluwer Academic Publishers.

Lin, H., J. Xiao and Z. Michalewicz (1994). Evolutionary Navigator for a Mobile Robot. In: *Proceedings of the IEEE International Conference on Robotics and Automation.* San Diego (California). pp. 2199–2204.

Meng, H. and P. D. Picton (1992). Obstacle Avoidance Using a Neural Network Controller and Visual Feedback. In: *Proc. SICICA '92 IFAC Symposium.* Málaga. pp. 563–568.

Meng, M. and A. C. Kak (1993). Mobile Robot Navigation Using Neural Networks and Nonmetrical Environment Models. *IEEE Control Systems* pp. 30–39.

Nagata, S., M. Sekiguchi and K. Asakawa (1990). Mobile Robot Control by a Structure Hierarchical Neural Network. *IEEE Control System Magazine* pp. 69–76.

Papageorgiou, M. and A. Steinkogler (1993). Real-Time Optimal Control of Moving Vehicles in Changing Environments. In: *Proc. of the 12th IFAC World Congress.* Sydney. pp. 107–112.

Pomerlau, D. A. (1990). *Vision and Navigation. The Carnegie Mellon Navlab.* Chap. Neural Network Based Autonomous Navigation, pp. 83–93. Kluwer Academic Publishers.

Press, W. H., B. P. Flannery, S. A. Teukolsky and W. T. Vetterling (1988). *Numerical Recipes in C.* Cambridge University Press.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* **65**, 386–408.

Rumelhart, D. E., G. E. Hinton and R. J. Williams (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Chap. Learning Internal Representations by Error Propagation, pp. 318–362. MIT Press.

Saffioti, A., E. H. Ruspini and K. Konolige (1993). Robust Control of a Mobile Robot Using Fuzzy Logic. In: *First European Congress on Fuzzy and Intelligent Technologies (EUFIT).* Aachen. pp. 881–886.

Sinha, P. K., A. Benmounah and K. A. Mutib (1992). A Mobile Robot: Exploring and Planning Trajectories. In: *Proc. SICICA '92 IFAC Symposium.* Málaga. pp. 545–549.

TRC (1989). *Labmate reference manual.*