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

J. Gómez Ortega and E. F. Camacho [1]

Dpto. Ingeniería de Sistemas y Automática, Univ. de Sevilla
Escuela Superior de Ingenieros, Avda. reina Mercedes s/n, 41012 Sevilla, Spain.
Fax: +34-5-4556849, email: juango@esi.us.es

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 non-linear model of mobile robot kinematics 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 and also eliminating the need for data sensor high level processing. The perceptron has been trained to reproduce the MBPC behaviour in a supervised manner. Experimental results obtained when applying the neural network controller to a mobile robot are given in the paper.

Keywords. Mobile robots; Predictive control; Navigation; Obstacle avoidance; Neural networks.

1. INTRODUCTION

One of the main problems in the development of autonomous mobile robots is the problem of path tracking in an environment with unexpected obstacles. *Model Based Predictive Control (MBPC)* is a suitable technique for applying to this problem (Papageorgiou and Steinkogler, 1993) because the objetive of 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. 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, the signals are not bounded and the objetive function is a quadratic one, then the MBPC requires little computation.

If the control signal is constrained, the system model is nonlinear and the objetive function is not quadratic (as in the case of mobile robot path tracking with unexpected obstacles), the MBPC results in a much more complex and time consuming problem. This paper shows how an Artificial Neural Network can be used to solve the problems mentioned above.

2. MODEL BASED PREDICTIVE CONTROL NAVIGATION

2.1 MBPC strategy

The objetive of predictive control is to obtain a future control action sequence $(U(t), U(t+1|t), ..., U(t+N_u-1|t))$ in such a way that the future predicted outputs $\hat{Y}(t+i|t)$ will be as close as possible to the desired references $Y_d(t+i)$ over the prediction horizon. This is

[1] This work has been partially funded by *CICYT*.

accomplished by the minimization of a cost function J with respect to the control variables. Often, the cost function J is the sum of the squared future errors in reference tracking, $\hat{e}(t+i|t) = \hat{Y}(t+i|t) - Y_d(t+i)$, predicted over a *prediction horizon* $N = N_2 - N_1$, and also of the sum of the squared control actions foreseen for the future, $U(t+i|t)$, over a *control horizon* N_u

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

The future system outputs, $\hat{Y}(t+i|t)$ for $i = N_1, ..., N_2$, are predicted from a model of the process, from past inputs and outputs, and from the control actions foreseen for the future, $U(t+i|t)$ 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.

After this sequence is obtained, a *receding horizon* approach is considered. This consists of only applying the first calculated control action $U(t)$. 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 is that of driving a mobile robot to follow a previously calculated desired path avoiding unexpected static obstacles.

The cost function used here is:

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

where $\hat{Y}(t+i|t) = \{\hat{x}(t+i|t), \hat{y}(t+i|t)\}$ is an i-step prediction of the robot position made at instant t, 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 angular velocity. These last two terms ensure smooth robot guidance. The last term penalizes the proximity between the robot and the obstacles, which are detected with an ultrasound proximity system placed on board the mobile robot. This is a potential function term, where $dist_f(\cdot)$ is a measure in $t+i$ of the distance between the robot and a fixed obstacle FO_j, which is considered to have a poligonal geometry in the plane. A more precise description of this function is presented below. A block diagram of the system is shown in figure 1.

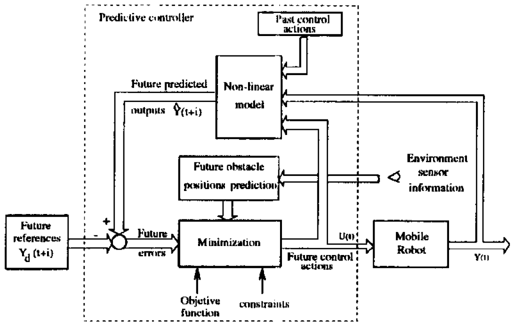


Fig. 1. The predictive controller scheme.

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 $(\omega_r(t), \omega_l(t))$. In the following 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.

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.

2.2 Prediction Model

For an MBPC formulation, a 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 has been used (figure 2).

A model of the LABMATE mobile robot, which takes into account the dead time produced by communications with the host processor, was obtained by using kinematic equations and identification tests. The following kinematic model (which corresponds to a differential-drive vehicle) is used for computing the predictions:

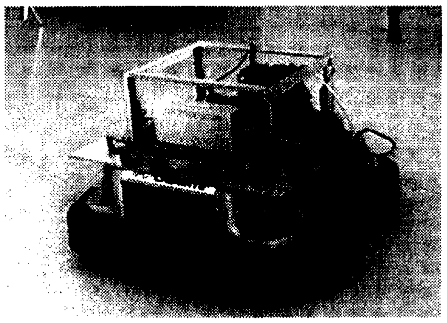


Fig. 2. The LABMATE mobile robot.

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

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

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

$$\mathcal{A} = R\frac{\omega_r(k-1) - \omega_l(k-1)}{2W}$$

$$V = R\frac{\omega_r(k-1) + \omega_l(k-1)}{2}$$

where x, y, θ are the position and heading of the robot in a fixed reference frame (see figure 3), T is the sample interval and W is the half-distance between wheels, which value has been estimated to be 168 mm (figure 3). V is the linear velocity of the mobile robot, $\mathcal{A}$ 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. In the case of a linear trajectory ($\mathcal{A} = 0$), the equations of motion are given by:

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

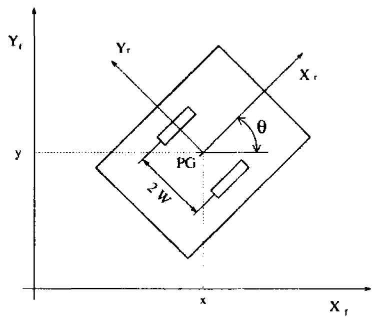


Fig. 3. Reference Frame

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

2.3 Parametrization of the desired path.

The reference path is given to the MBPC 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 figure 4, 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 $(X_r(k), Y_r(k))$. The next N points are spaced equally on the path, with a separation between them of ΔS, which is a design parameter.

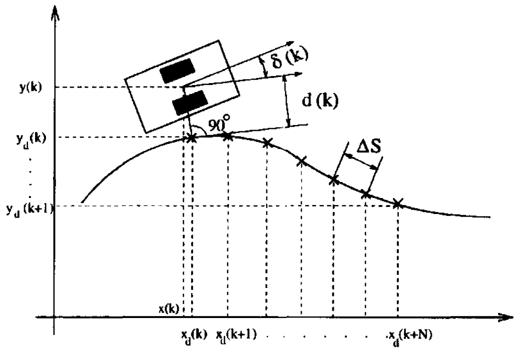


Fig. 4. Parametrization of the desired path.

2.4 Potential function for considering fixed obstacles

As stated above, the fixed obstacles are considered to have polygonal geometry in the plane, and the surfaces of the obstacles are considered to be perpendicular to the moving plane of the mobile robot.

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

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 out of it, as the distance to the frontier is augmented. NSF is the number of segments of the obstacle frontier. The following potential function is used:

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

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 figure 5, where two rectangular shape static obstacles are present in the proximity of the robot.

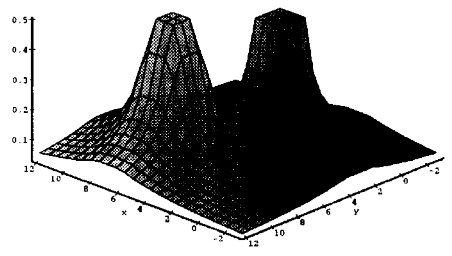


Fig. 5. Convex regions potential function.

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 frontier. This function has the same characteristics as $p_{ccx}(x)$.

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 figure 6. In this figure 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 non linear prediction model can be observed.

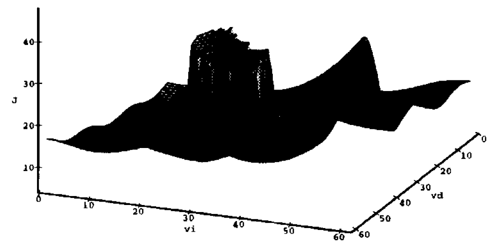
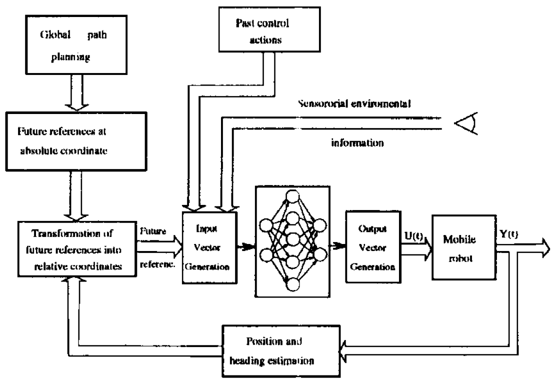


Fig. 6. Objetive function J.

3. THE NEURAL NETWORK 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 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 (Pomerlau, 1990), (Meng and Kak, 1993).

The modules of the control scheme used in this work (see figure 7) are:



Global path planning
Past control actions
Sensororial enviromental information
Future references at absolute coordinate
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. 7. Predictive neural network scheme for mobile robot navigation.

- **Artificial neural network controller**. The ANN architecture chosen here is a Multilayer Perceptron, with one hidden layer (see figure 8).

  The input layer consists of twelve neurons (see figure 8). The first two inputs correspond to the previous linear and angular velocities of the robot. The next three inputs are associated to 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 curva-

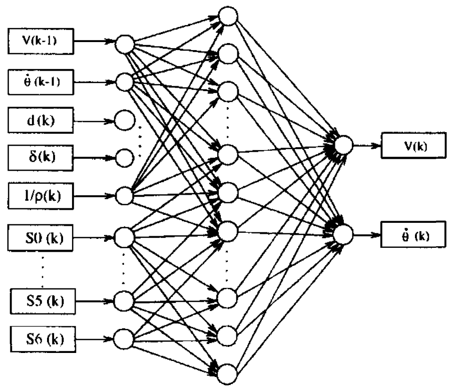


Fig. 8. Neural Network Scheme

ture of the future desired points ($1/\rho$) (see figure 4). The last seven inputs correspond to the distances measured directly by the ring of sonar sensors. This fact avoids the high level process that usually has to be carried out with sensor data in order to provide useful environment information, which is a difficult and high computation time consuming phase.

The output layer consists of two nodes which correspond to the control commands (the linear and angular robot velocities).

- **Inputs vector generation module**. A symetry 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). Also a normalization is made which leads to better performance at the ANN training stage.
- **Outputs vector generation module**. This performs the inverse transformation of that made at the symetry input module when required.
- **Reference path coordinates 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 ANN to consider the delay time of the robot system.
- **Sonar range measurements**. Their measurements are directly used as inputs for ANN. ANN learns from the input patterns set, where different situations of static obstacles are present. Thus, there is no need for a high level sonar measurements preprocess, which guarantees real time.

3.1 Training phase

The ANN controller has been trained using a classic supervised training scheme as the backpropagation algorithm (see figure 9). The training patterns set have

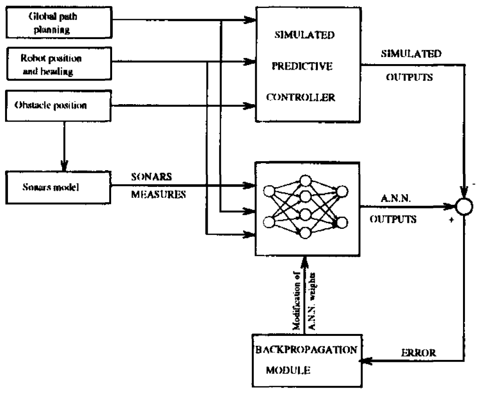


Fig. 9. ANN supervised training scheme.

been obtained from an off-line simulation system. For the minimization phase a *Powell iterative algorithm* has been used, where constraints on the control variables are considered. Also, the sonar system measurements have been simulated using a sonar model where the objects in the environment are described as a set of geometric primitives such as planes, cylinders, edges and corners (Leonard and Durrant-Whyte, 1992).

4. RESULTS

The proposed control structure has been tested with the LABMATE mobile robot. The ANN used consisted of twelve input neurons, a hidden layer with eighteen neurons and the output layer with two neurons (see figure 8).

The ANN was trained in a supervised manner, as described previously. The sampling interval T was given a value of 2 s. The value of N chosen for the MBPC was 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.

Figure 10 shows some of the experimental results obtained in the laboratory when applying the proposed algorithm to the LABMATE mobile robot. Although it is drawn as straight lines in the figures, the real environment includes laboratory objects such as chairs, tables,

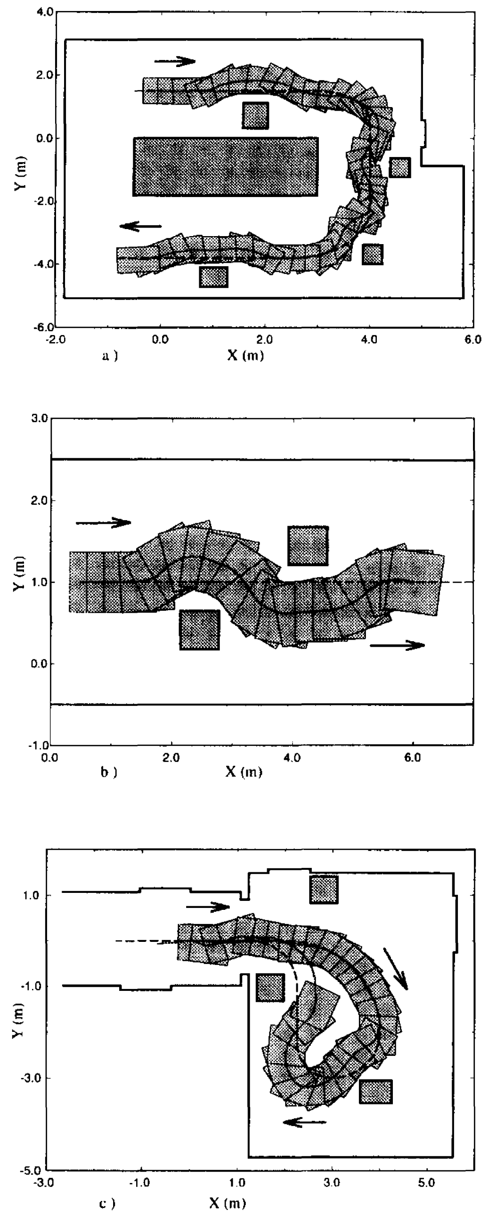


Fig. 10. Experimental results of the Neural predictive controller for mobile robot navigation.

etc. This shows the robustness of the neural controller as it has been trained with only a set of geometric primitives such as planes, corners, etc.

Figure 10(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 10(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, figure 10(c) shows another test for a path where small curvature radii are specified. The tracking error observed in figure 10(c) is due to saturation in the angular velocity and to the penalization chosen for the control actions.

5. CONCLUSIONS

A predictive ANN controller for mobile robot navigation in partially structured static environments has been presented. The ANN has been trained in a supervised way with a backpropagation algorithm. The desired ANN 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 with numerical methods would be prohibitive for real time. The ANN approach has proved to be an effective way of implementing the path tracking predictive algorithm as shown by experimental tests.

6. REFERENCES

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.

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

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

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

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.