Proceedings of the 1999 IEEE
International Conference on Robotics & Automation
Detroit, Michigan • May 1999

Nonlinear MBPC for Mobile Robot Navigation Using Genetic Algorithms

D.R. Ramírez, D. Limón, J. Gómez-Ortega and E.F. Camacho *
Departamento de Ingeniería de Sistemas y Automática, Universidad de Sevilla
Escuela Superior de Ingenieros, Camino de los Descubrimientos s/n 41092 Sevilla
Fax: +34 954487340, email: {danirr,limon,juango,eduardo}@cartuja.us.es

Abstract

This paper presents a way of implementing a model-based predictive controller (MBPC) for mobile robot navigation using genetic algorithms to perform online nonlinear optimization. Obstacle avoidance is considered by the controller. Constraints on the maximum attainable speeds are also considered by the algorithm. Experimental results obtained when applying the controller to a TRC LABMATE mobile platform are given in the paper.

1 Introduction

The work presented in this paper can be embedded in a research line followed in our Department in recent years, where predictive control techniques are applied to the mobile robot navigation problem. As is well known, navigation consists of driving a mobile robot through a working environment, from one start point to a final point. Usually, this environment is partially structured and then a previous path planning stage is quite useful, although unexpected objects can be found during the travel.

Predictive Control Strategies have been used for mobile robot navigation. In [2], [3] and [4] a non-linear model of the vehicle is used, and a neural-network based controller [2], [3] and a fuzzy based controller [4], both adequately trained from a set of training patterns, allowed the application of a real time predictive control strategy for mobile robot navigation with unforeseen obstacles.

In this paper, a genetic algorithm based predictive control strategy is used for mobile robot navigation in a partially structured environment. The use of genetic algorithms (GA) allows the minimisation of a nonlinear cost function in real time, avoiding the complex training process of the neural networks and fuzzy algorithms. A 2D laser sensor system has been used for obstacle detection and experimental results are presented in the paper.

The paper is organised as follows. In section 2, it is shown how a model based predictive control (MBPC) strategy can be applied to the mobile robot navigation problem. In section 3 the proposed GA based optimisation algorithm is presented. A comparison of the results obtained with this approach and with other optimisation methods is shown in section 4. Section 5 presents the experimental tests and section 6 includes the conclusions.

2 MBPC technique for mobile robot navigation

2.1 MBPC strategy

In general, most MBPC methods are based on a common performance scheme. 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:

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

The notation $\hat{Y}(k+i|k)$ indicates that $\hat{Y}(k+i)$ is calculated with the data known at sample time

**Acknowledgment*-The authors would like to acknowledge the CICYT for funding this work under grant TAP96-0884-CP and TAP98-0541.

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. 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 is that of driving a mobile robot to follow a previously calculated desired path avoiding the unexpected static obstacles. A predictive control technique is used as the robot navigator. The cost function used here is:

$$\begin{aligned} J(N_1, N_2, N_u) = & \sum_{i=N_1}^{N_2} \mu_1[\hat{x}(k+i|k) - x_d(k+i)]^2 \\ & + \sum_{i=N_1}^{N_2} \mu_2[\hat{y}(k+i|k) - y_d(k+i|k)]^2 \\ & + \sum_{i=N_1}^{N_2} \mu_3[\hat{\theta}(k+i|k) - \theta_d(k+i)]^2 \\ & + \sum_{i=1}^{N_u} \lambda[\omega_r(k+i-1) - \omega_l(k+i-1)]^2 \\ & + \sum_{j=1}^{NFO} (\sum_{i=N_1}^{N_2} \frac{\psi}{[\mathrm{dist}_f(\hat{Y}(k+i|k), FO_j)]^2}) \end{aligned}$$

where $[\hat{x}(k+i|k), \hat{y}(k+i|k), \hat{\theta}(k+i|k)]$ is an i-step prediction of the robot position and heading made at instant k, and $[x_d(k+i), y_d(k+i), \theta_d(k+i)]$ are the desired robot position and heading; ω_r and ω_l are the right and left angular velocities of the two driving wheels, which are the control variables, and μ_1, μ_2, μ_3, λ and ψ are constant weighting factors. The first three terms in J penalize the position error; the fourth term penalizes the robot's angular velocity. These terms ensure smooth robot guidance.

The last term penalizes the proximity between the robot and the obstacles, which are detected with a 2D laser range system placed on-board the mobile robot. This is a potential function term, where $dist_f(\cdot)$ is the distance between the predicted position of the robot in $k+i$ and a fixed obstacle FO_j, which is considered to have a circular geometry in the plane. NFO is the number of fixed obstacles detected in the environment. A more precise description of this function is presented below. A block diagram of the control system is shown in Fig. 1.

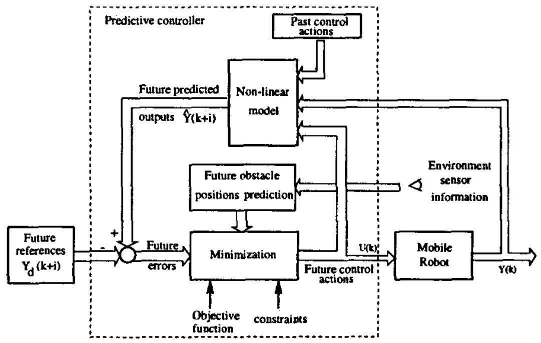

Figure 1: The predictive controller scheme.

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 $(\omega_r(k), \omega_l(k))$. In what follows, N_1 and N_2 will be considered to be $N_1 = d+1$ and $N_2 = N+d$, 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 horizons involved in the controller have only one free parameter, N.

2.2 Prediction model

For a MBPC formulation, a kinematic 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 (Fig. 2).

The following nonlinear model of the LABMATE mobile robot (which corresponds to a differential-drive vehicle), obtained by using kinematic equations and identification tests , is used for computing the predictions:

$$\begin{aligned} \theta(k+1) &= \theta(k) + \dot{\theta}(k-1)T \\ x(k+1) &= x(k) + \frac{V(k-1)}{\dot{\theta}(k-1)}\{\sin(\theta(k) + \dot{\theta}(k-1)T) \end{aligned}$$

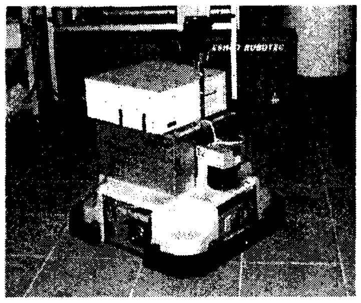

Figure 2: The LABMATE mobile robot.

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

where x, y, θ are the position and heading of the robot in a fixed reference frame (see Fig. 3), T is the sample interval and W is the half-distance between the wheels, which value has been estimated to be 168 mm. V is the linear velocity of the mobile robot, $\dot\theta$ is the steering speed, and ω_r, ω_l and R are the right and left wheel angular velocities 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:

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

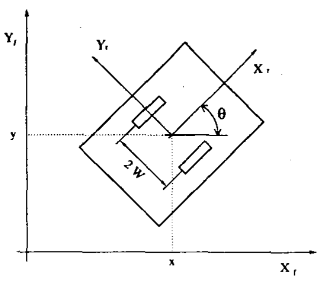

Figure 3: Reference frame

For this model the value of d is equal to 1. Using the maximum acceleration value, the velocities of both wheels have been considered to be constant for each sample period.

2.3 Potential function for fixed obstacles

As stated above, the fixed obstacles are considered to have a circular geometry in the plane. For this 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_{f_o}, defined as the center of the circumference. This distance is decreased by the obstacle radius R_{f_o} and the mobile robot size, in order to consider their dimensions:

$$dist_f = \|\hat{Y}(k+i|k) - Y_{f_o}\|_2 - R_{f_o}$$

This function is computed for each obstacle detected in the proximity of the mobile robot and is computed not only in the present but also in the future, using the predicted locations of the mobile robot calculated with the prediction model. The sensor data needed for the computation of this function is obtained from a 2D laser range sensor located onboard the robot. The details of the sensor data processing are described in [1].

3 Online Optimization with Genetic Algorithms

Genetic Algorithms (GAs) are stochastic algorithms whose search methods resemble the natural evolution process. They start with a *population* of randomly generated candidates and evolve towards better solutions according to the principles of natural selection and "survival of the fittest". Each individual represents a possible solution to a given problem and a *fitness score* is assigned according to how good a solution to the problem is. New individuals are generated by recombining the genetic material of the best individuals in every generation or by mutation. If the GA has been designed properly, the *population* will *converge* to an optimal solution of the problem.

When applying GAs to the online optimization in the MBPC problem it is compulsory to achieve good solutions in real time. This leads to small populations and fast evolution requirements. This can be partially accomplished if the initial population is randomly generated only for the first sample time. For the following sample times it can be used the best individuals of the resulting population of the previous optimisation problem, expecting that the optimal solution of the current problem will be near the previous one. The remaining individuals of the population are randomly generated to preserve the stochastic characteristics of the search. By doing this, local optima problems are avoided. This procedure takes some of the ideas of *serial selection*.

On the other hand, genetic operators should be chosen carefully. Whereas fast convergence is needed in early generations, *fine tuning* is required in the later as the accuracy of the solution is an important issue, especially dealing with nonlinear systems. For the *fine tuning* task two operators have been chosen: *non uniform mutation* [6] and *heuristic crossover* [7]. Non uniform mutation is a specialized mutation operator which has the form:

$$v'_k = \begin{cases} v_k + \Delta(t, u_k - v_k) & \text{if a random bit is 0} \\ v_k - \Delta(t, v_k - l_k) & \text{if a random bit is 1} \end{cases} \quad (1)$$

where v_k is the gene being mutated, (u_k, l_k) are the upper and lower bounds of v_k and the function $\Delta(t,y)$ is:

$$\Delta(t,y) = y \cdot r \cdot (1 - \frac{t}{T})^b \quad (2)$$

being r a random real number in the range $[0,1]$, b a constant (with a value of 2 here) and T the maximal generation number.

The heuristic crossover also plays an important role in fine tuning, and it works by creating an offspring s_3 from s_1 and s_2, where s_2 is better solution than s_1, according to the following rule:

$$s_3^{t+1} = r \cdot (s_2^t - s_1^t) + s_2^t \quad (3)$$

where r is a random number in the range $[0,1]$. This operator could produce such an offspring vector which is not a feasible solution. The implementation used here differs from the standard in the way it handles this situation. Instead of trying with another random number, the solution is saturated to the nearest boundary value. It makes faster the execution and helps the algorithm in searching near the boundaries which it has been reported as a difficult task for the standard genetic algorithms [5].

The remaining operators are standard genetic operators for real coded genetic algorithms: *random mutation* and *whole arithmetical crossover* which has the form:

$$\begin{aligned} s_v^{t+1} &= a \cdot s_w^t + (1-a) \cdot s_v^t \\ s_w^{t+1} &= a \cdot s_v^t + (1-a) \cdot s_w^t \end{aligned} \quad (4)$$

where a is constant or a random number in the range $[0,1]$. The latter option is used here as it has been reported to be more efficient [5].

Other aspects of the genetic algorithm are the sampling mechanism and the termination condition. The sampling mechanism adopted is selection based on fitness with an *elitist model* [5]. Finally the termination condition for the algorithm used is to do a fixed number of generations. Also the solutions have been coded using a floating point representation.

4 Performance analysis

To validate the performance of the proposed GA optimisation algorithm, a comparison has been carried out between its results and the ones obtained with a standard GA optimisation method with the values computed with the well known Powell algorithm . The reference path used for the tests is shown in figure 4. This path is unlikely to be generated by any trajectory planning method but is used here to make harder the optimisation procedure. The parameters of both GAs

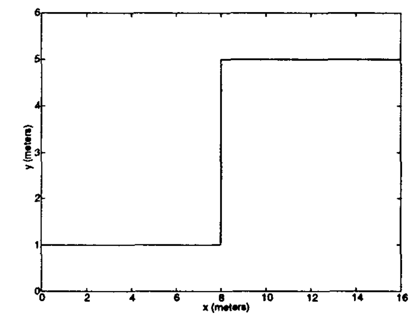

Figure 4: Path used for performance tests.

are the same, with a small population of 50 individuals and an evolution process limited to 50 generations. The crossover and mutation probabilities are 0.65 and 0.05 respectively. For the proposed algorithm two additional parameters are needed. The probability of use *non uniform mutation* instead of *random mutation* is set to 0.95. Similarly, the probability of use *heuristic crossover* instead of *whole arithmetical crossover* is set to 0.15.

Figure 5 shows the exact values of $\dot{\theta}$ and those calculated using the proposed algorithm (Figure 5a) and the SGA (Figure 5b) keeping as initial conditions for every sample time the exact values found in the previous sample. The proposed algorithm clearly outperforms the SGA especially when the reference remains unchanged or the change is smooth. This is due to the combined effect of population initialization scheme and fine tuning operators. When there is an abrupt change of the reference path the efficiency of the proposed algorithm decreases, but it is still better than the SGA. Again the fine tuning operators improve the accuracy of the solution found.

Figure 6 shows the absolute error values for both algorithms for two different set of horizons. Dotted lines correspond to SGA and solid lines to the proposed approach. This error is defined as the difference between the optimum values given by the GA and the ones given by the Powell algorithm. Figure 6a shows error values for the results of figure 5. The magnitude of the error is small enough in the case of the algorithm presented here whereas in the case of SGA the error evolution shows oscillations due to the early finished search process. To show how the accuracy of the solutions found can be affected by large control horizons, the same configuration of the genetic algorithm has been tested with (N1,N2,Nu) = (2,11,10). Although the magnitude of the error values is bigger the shape is almost the same, showing a particularly good behaviour when the change in the reference path is smooth. It can be expected that increasing the population or searching over more generations the results will be improved.

Finally, the mean minimum value of J found in the experiments shown in figures 4, 7 and 8 has been calculated to show how the errors in the solutions found affect the performance of the controller. Powell algorithm gives a mean minimum value of J for each path of 0.0712, 0.2328, and 0.2127 respectively. The proposed algorithm deviates from this values by 0.3%, 1.51%, and 4.92% respectively. However the position and heading error for the proposed algorithm deviates only by 0.05%, 0.9% and 1% from the Powell algorithm.

5 Experimental Results

The proposed control structure has been experimentally tested with the LABMATE mobile robot. The value of N chosen for the MBPC was experimentally made equal to 7; thus N_1, N_2 and N_u were given the values 2, 9 and 7, respectively. The population used in the genetic algorithm was 50 individuals and the maximal generation number was 100. The probabilities were the same as in section 4. Figures 7 and 8 show some of the experimental results obtained in the laboratory . In the figures, the dotted lines represent the reference paths and the solid lines the real paths.

Figure 7 shows the desired trajectory and the real trajectory for a path tracking experiment. For this

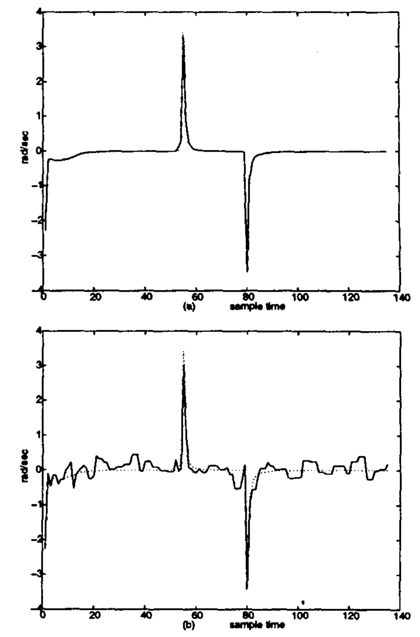

Figure 5: Values of $\dot{\theta}$ calculated (solid line) compared with the exact solution (dotted line) for (N1,N2,Nu) = (2,6,5). (a) Proposed algorithm. (b) SGA.

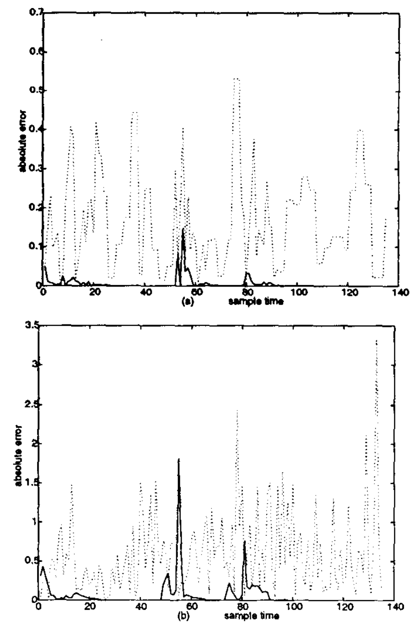

Figure 6: Absolute error values of both algorithms for (a) (N1,N2,Nu) = (2,6,5). (b) (N1,N2,Nu) = (2,11,10).

test, the following values of the parameters were chosen: $\mu_1 = 2.0$, $\mu_2 = 2.0$, $\mu_3 = 0.25$ $\lambda = 0.05$ and $\psi = 0.0$. The maximum and minimum angular velocities were given the following values respectively: 0.8 rad/s and -0.8 rad/s. The linear robot velocity was 0.6 m/s and the sample time 0.3 seconds.

Figure 8 shows an experiment where unexpected static obstacles have been positioned. For this test the following values of the parameters were chosen: $\mu_1 = 2.0$, $\mu_2 = 2.0$, $\mu_3 = 0.6$ $\lambda = 0.05$ and $\psi = 0.2$. The maximum and minimum angular velocities were 0.5 rad/s and -0.5 rad/s respectively. The linear robot velocity was 0.3 m/s and the sample time 0.6 seconds. It is important to notice how the mobile robot returns to the reference path after an unexpected obstacle has been avoided.

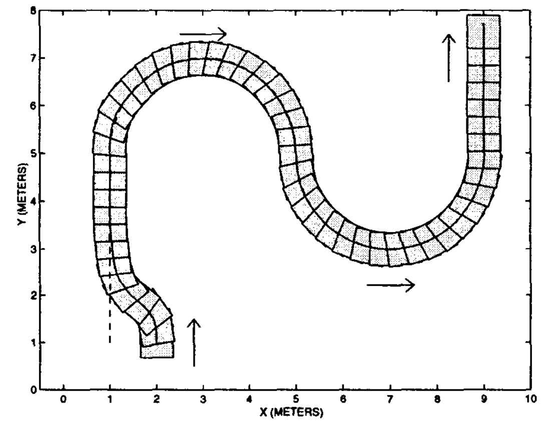

Figure 7: Experiment 1. Path tracking

6 Conclusions

A MBPC for mobile robots has been presented. Path tracking and obstacle avoidance is achieved by including the distance to obstacles in the objective function. Control signal saturations and nonlinearities of the model were considered in order to obtain accurate predictions of the robot trajectories. The nonlinear optimisation of the objective function is carried out by means of genetic algorithms. The computation time to obtain a solution near the optimal is small enough for real time. This approach has proved to be an effective way of implementing the predictive algorithm as shown by the simulation and experimental tests.

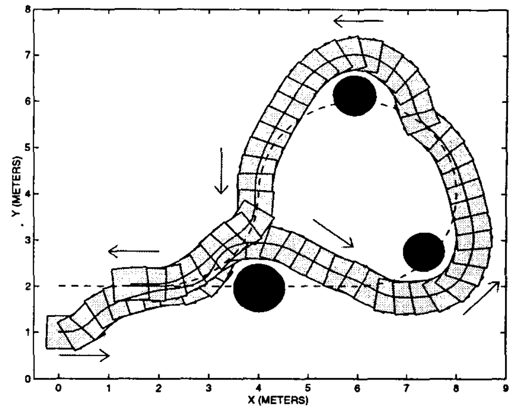

Figure 8: Experiment 2. Path tracking and obstacle avoidance

References

[1] Limón, D., Ramírez, D.R. and Gómez-Ortega, J., "Obstacle detection with a 2D laser range sensor," *Internal Report, Dpt. Ingeniería de Sistemas y Automática, Univ. de Sevilla (Spain), July 1998*

[2] Gómez-Ortega, J. and Camacho, E.F., "Neural Network MBPC for Mobile Robot Path Tracking," *Robotics & Computer-Integrated Manufacturing*, Vol. 11, pp. 271-278, 1994.

[3] Gómez-Ortega, J. and Camacho, E.F., "Mobile Robot Navigation in a Partially Structured Static Environment, using Neural Predictive Control" *Control Engineering Practice*, Vol. 4, pp. 1669-1679, 1996.

[4] Gómez-Ortega, J., Rubio, F.R. and Camacho, E.F., "Fuzzy Predictive Controller for Mobile Robot Path-Tracking," *Proc. SICICA'97, pp 283-288, Annecy (France), July 1997*

[5] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs,* Springer-Verlag, 1994

[6] Michalewicz, Z. and Schoenauer, M., "Evolutionary Algorithms for Constrained Parameter Optimization Problems" *Evolutionary Computation,* Vol. 4, pp. 1-32, 1996

[7] Wright, A., "Genetic algorithms for real parameter optimization" in J.G. Rawlins (Ed.) *Foundations of Genetic Algorithms*, pp. 205-218, Morgan Kauffman, 1991