An Iterative Learning Controller for Nonholonomic Mobile Robots

Giuseppe Oriolo, Stefano Panzieri and Giovanni Ulivi

The International Journal of Robotics Research 1998; 17; 954
DOI: 10.1177/027836499801700904

Giuseppe Oriolo
Dipartimento di Informatica e Sistemistica
Università di Roma, La Sapienza, Italy

Stefano Panzieri
Giovanni Ulivi
Dipartimento di Informatica e Automazione
Terza Università di Roma, Italy

An Iterative Learning Controller for Nonholonomic Mobile Robots

Abstract

We present an iterative learning controller that applies to nonholonomic mobile robots, as well as other systems that can be put in chained form. The learning algorithm exploits the fact that chained-form systems are linear under piecewise-constant inputs. The proposed control scheme requires the execution of a small number of experiments to drive the system to the desired state in finite time, with nice convergence and robustness properties with respect to modeling inaccuracies as well as disturbances. To avoid the necessity of exactly reinitializing the system at each iteration, the basic method is modified so as to obtain a cyclic controller, by which the system is cyclically steered through an arbitrary sequence of states. As a case study, a carlike mobile robot is considered. Both simulation and experimental results are reported to show the performance of the method.

1. Introduction

Many robotic systems for advanced applications are subject to nonholonomic constraints. Examples include wheeled mobile robots (Latombe 1991), space manipulators (Xu and Kanade 1993), and multifingered robot hands (Murray, Li, and Sastry 1994). This situation is particularly interesting, because it implies that the mechanism can be controlled with a reduced number of actuators. On the other hand, both planning and control require special techniques. In fact, nonholonomic robots cannot follow arbitrary trajectories, due to their kinematic constraints, and they cannot be stabilized at a given equilibrium point by smooth static feedback, as inferred from a celebrated result by Brockett (1983).

The problem of steering a nonholonomic robot between two configurations can be tackled with the following two approaches:

1. Use open-loop control to generate a trajectory that connects the two configurations and complies with the nonholonomic constraints (Lafferriere and Sussmann 1991; Monaco and Normand-Cyrot 1992; Murray and Sastry 1993;Tilbury, Murray, and Sastry 1993). Although the obtained trajectories are naturally appealing, such a solution is not robust with respect to disturbances or modeling inaccuracies. To emphasize this fact, such an approach is often referred to as *motion planning*.
2. Use a feedback-stabilizing control that is not ruled out by the aforesaid limitation, namely, a nonsmooth (Bloch, McClamroch, and Reyhanoglu 1990; Canudas de Wit and Sørdalen 1992) and/or time-varying (Samson 1993) law. The design of these types of controllers is more difficult, and they often give rise to erratic, awkward motions. However, their use provides a certain degree of robustness.

One possible way to combine the advantages of both approaches is represented by *learning control*, a methodology for improving the performance of a control system through iterative training (Arimoto, Kawamura, and Miyazaki 1984; Bondi, Casalino, and Gambardella 1988; Moore 1993; Lucibello 1994). At each iteration, a new control law is devised, based on the previous law and the value of the performance error. Robustness properties with respect to many kinds of perturbations have been analytically demonstrated and experimentally verified. Such technique has proved to be reliable in practical applications, both for trajectory tracking and for state steering.

To appreciate the practical relevance of this approach in mobile robot applications, consider the following example. A wheeled mobile robot must repeatedly perform the same task, namely, move from a start configuration q^0 to a desired configuration q^d in a specified time (this may be the case of a transportation task). An open-loop controller is used to generate a feasible trajectory from q^0 to q^d. However, due to possible errors in initial conditions or to the presence of model perturbations and disturbances, the robot may end its motion in a different configuration $q^f \neq q^d$. A straightforward way to "close the loop" would be to replan a path from q^f to q^d, but

The International Journal of Robotics Research
Vol. 17, No. 9, September 1998, pp. 954–970,

proving its effectiveness is not trivial (Lucibello and Oriolo 1996). Moreover, with this approach, the robot would not gain any information from experience, and would be subject to the same error at the next execution of the task. Instead, the learning paradigm requires the robot to go back to the starting point, to compute a new control taking into account the final error, and to perform a second trial. After a few experiments, the robot will be able to reach the desired position with a negligible error.

Here, we address the learning control problem for nonholonomic robots which can be put in chained form—a fairly general canonical structure for such systems (Murray and Sastry 1993). The adoption of the chained-form representation is particularly convenient from our viewpoint, for it allows the synthesis of efficient open-loop controllers, as well as the design of a simple learning algorithm.

To establish the effectiveness of the proposed method, one must assume that the system is exactly reinitialized at each iteration. To overcome this drawback, we shall also present a *cyclic* learning controller, i.e., an extension of the basic learning strategy in which the system is cyclically steered among an arbitrary number of equilibrium points.

The paper is organized as follows. In Section 2, we briefly recall the properties of chained systems as canonical forms for nonholonomic systems, and introduce the car-like robot system. Then, we present in Section 3 a learning-control method for a particular class of linear time-varying systems, and discuss its application to chained systems. Simulation and experimental results on a car-like robot are given in Sections 4 and 5, respectively. The extension to cyclic learning control is presented in Section 6. A discussion on possible extensions concludes the paper.

2. Chained-Form Systems

Consider the two-input driftless nonlinear control system

$$\begin{aligned} \dot{z}_1 &= v_1, \\ \dot{z}_2 &= v_2, \\ \dot{z}_3 &= z_2 v_1, \\ &\vdots \\ \dot{z}_n &= z_{n-1} v_1, \end{aligned} \tag{1}$$

a special case of the *chained form* called a *one-chain* system (Murray and Sastry 1993). Its structure is particularly interesting.

First, note that if we choose a constant value for v_1, the system becomes linear, time-invariant, and behaves like a chain of integrators from z_2 to z_n, driven by the input v_2. More in general, if v_1 is a given time-dependent function, eq. (1) represent a linear time-varying system. Second, controllability may be readily established by using the Lie algebra rank condition (Isidori 1995). Finally, all the Lie brackets above a certain degree are identically zero; this property of the system is called *nilpotency*.

Murray (1993) has established necessary and sufficient conditions for converting a generic two-input driftless nonlinear system,

$$\dot{q} = h_1(q)u_1 + h_2(q)u_2, \tag{2}$$

into chained form by means of a change of coordinates $z = \alpha(q)$ and an input transformation $u = \beta(q)v$. In particular, nonholonomic systems with two inputs and $n \leq 4$ state variables can *always* be put in chained form.

2.1. Example: The Car-like Robot

Consider a robot having the same kinematic model of an automobile (Fig. 1). As is customary, we assume that the two wheels of each axis (front and rear) collapse into a single wheel located at the midpoint of the axis (*bicycle* model). The front wheel can be steered while the rear-wheel orientation is fixed. The distance between the two wheel centers is ℓ.

The generalized coordinates are $q = (x_r, y_r, \theta, \phi)$, where (x_r, y_r) are the Cartesian coordinates of the rear-axle midpoint, θ is the orientation of the car body with respect to the x-axis, and ϕ is the steering angle.

For a rear-wheel drive car, the kinematic model is

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{1}{\ell}\tan\phi \\ 0 \end{bmatrix} \rho u_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_2, \tag{3}$$

where ρ is the driving-wheel radius, u_1 is the angular velocity of the driving wheel, and u_2 is the steering rate. Note the presence of a control singularity at $\phi = \pm\pi/2$, namely, when

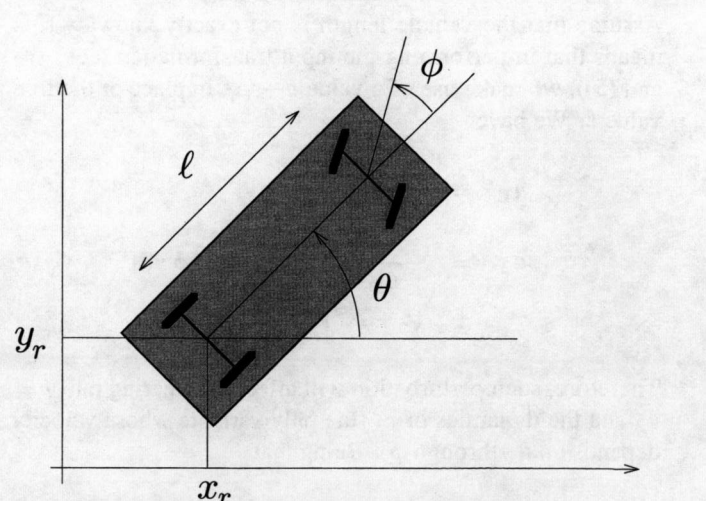

Fig. 1. Generalized coordinates for a car-like robot.

the front wheel is normal to the longitudinal axis of the vehicle. The relevance of this singularity is limited, due to the restricted range of ϕ in practical cases.

To convert this system into chained form, use the change of coordinates $z = \alpha(q)$, given by

$$\begin{aligned} z_1 &= x_r, \\ z_2 &= \frac{1}{\ell}\frac{\tan\phi}{\cos^3\theta}, \\ z_3 &= \tan\theta, \\ z_4 &= y_r, \end{aligned}$$

together with the input transformation $u = \beta(q)v$ expressed as

$$u_1 = \frac{v_1}{\rho\cos\theta}, \tag{4}$$

$$u_2 = -\frac{3}{\ell}\frac{\sin\theta}{\cos^2\theta}\sin^2\phi\, v_1 + \ell\cos^3\theta\cos^2\phi\, v_2. \tag{5}$$

It is easy to verify that the transformed system is in the form of eq. (1), with $n = 4$. Note that the change of coordinates is only locally defined, because $\theta \neq \pm\pi/2$ must hold. Hence, there is a one-to-one correspondence between the q- and the z-coordinates only if $\theta \in (-k\pi/2, k\pi/2)$ for $k = 1, 2, \ldots$.

Since the above transformation is model-based, an interesting issue is the effect of model perturbations on the chained structure for this robot. Below, we analyze in particular two cases; namely, perturbations on the vehicle length ℓ, and on the wheel radius ρ. Similar derivations can be given for other possible perturbations, such as an uncertainty in the measure of some state variable.

2.1.1. *Effect of a Perturbation on the Vehicle Length*

Assume that the vehicle length is not exactly known. This means that in performing the input transformation (eqs. (4) and (5)), we make use of a value $\ell + \Delta\ell$ in place of the true value ℓ. We have

$$\begin{aligned} u_1 &= \frac{v_1}{\rho\cos\theta}, \\ u_2 &= -\frac{3}{\ell+\Delta\ell}\frac{\sin\theta}{\cos^2\theta}\sin^2\phi\, v_1 \\ &\quad + (\ell+\Delta\ell)\cos^3\theta\cos^2\phi\, v_2. \end{aligned}$$

Therefore, such perturbation will affect the steering rate $\dot{\phi} = u_2$ and the dynamics of z_2 (the only variable whose velocity depends on u_2 through $\dot{\phi}$). Being that

$$\frac{1}{\ell+\Delta\ell} \approx \frac{1}{\ell}\left(1 - \frac{\Delta\ell}{\ell}\right)$$

for sufficiently small $\Delta\ell$, one obtains

$$\begin{aligned} \dot{z}_2 &= \frac{1}{\ell\cos^3\theta}\left(3\tan\theta\tan\phi\,\dot{\theta} + \frac{\dot{\phi}}{\cos^2\phi}\right) \\ &\approx v_2 + \frac{\Delta\ell}{\ell}\left(\frac{3\tan\theta\tan^2\phi}{\ell^2\cos^4\theta}v_1 + v_2\right). \end{aligned}$$

The perturbed model in the z-coordinates is then

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \\ \dot{z}_4 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ z_2 v_1 \\ z_3 v_1 \end{bmatrix} + \frac{\Delta\ell}{\ell}\begin{bmatrix} 0 \\ 3z_2^2 z_3\cos^2(\arctan z_3)v_1 + v_2 \\ 0 \\ 0 \end{bmatrix}. \tag{6}$$

2.1.2. *Effect of a Perturbation on the Wheel Radius*

Assume now that the driving-wheel radius is not exactly known. As a consequence, in performing the input transformation (eqs. (4) and (5)), we make use of a value $\rho + \Delta\rho$ in place of the true value ρ. We have

$$\begin{aligned} u_1 &= \frac{v_1}{(\rho+\Delta\rho)\cos\theta}, \\ u_2 &= -\frac{3}{\ell}\frac{\sin\theta}{\cos^2\theta}\sin^2\phi\, v_1 + \ell\cos^3\theta\cos^2\phi\, v_2. \end{aligned}$$

Different from the previous case, such a perturbation will affect the forward velocity u_1 and, hence, the dynamics of all z-variables. Being that

$$\frac{\rho}{\rho+\Delta\rho} \approx 1 - \frac{\Delta\rho}{\rho}$$

for sufficiently small $\Delta\rho$, simple computations give the perturbed model in the z-coordinates as

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \\ \dot{z}_4 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ z_2 v_1 \\ z_3 v_1 \end{bmatrix} + \frac{\Delta\rho}{\rho}\begin{bmatrix} v_1 \\ \frac{3z_2 z_3}{\ell\cos(\arctan z_3)}v_1 \\ z_2 v_1 \\ z_3 v_1 \end{bmatrix}. \tag{7}$$

2.2. *Steering Chained-Form Systems*

Suppose that we wish to steer the state of system described by eq. (2) from an initial configuration $q(0) = q^0$ to a desired configuration q^d in a finite time T. Henceforth, it is assumed that both q^0 and q^d belong to a region of the configuration space where the chained-form transformation is one-to-one.[1]

1. This assumption does not imply any loss of generality; see the remark at the end of Section 4.

Correspondingly, the chained-form system of eq. (1) must move from $z^0 = \alpha(q^0)$ to $z^d = \alpha(q^d)$.

As already mentioned, the adoption of open-loop control laws is convenient because their design is relatively simple and the obtained trajectories are predictable. Among the proposed techniques, we cite the use of the following:

- sinusoidal inputs (Murray and Sastry 1993), inspired by Brockett's work on optimal control for a class of systems (Brockett 1981);
- piecewise-constant inputs, as in multirate digital control (Lafferriere and Sussmann 1991; Monaco and Normand-Cyrot 1992); and
- mixed piecewise-constant and polynomial inputs (Tilbury, Murray, and Sastry 1993), to obtain a smoother behavior.

Below, we recall the latter method, which will be adopted to illustrate our learning controller.

Let

$$v_1(t) = c_1(t), \tag{8}$$

$$v_2(t) = c_{21} + c_{22}t + \ldots + c_{2,n-1}t^{n-2}, \tag{9}$$

where $c_1(t)$ is a piecewise-constant function and $c_{21}, \ldots, c_{2,n-1}$ are constants. It is always possible to choose the function $c_1(t)$ and the coefficients c_{2i} $(i = 1, \ldots, n-1)$ so as to steer system (1) from z^0 to z^d in a finite time T. In particular, $c_1(t)$ should satisfy

$$\int_0^T c_1(t)dt = z_1^d - z_1^0, \tag{10}$$

while the remaining $n-1$ unknown coefficients c_{2i} $(i = 1, \ldots, n-1)$ can be found by imposing the desired reconfiguration on the variables $z_2, \ldots, z_n$. By forward integration of eq. (1), a linear relationship of the following form is derived:

$$M(c_1, T)\begin{bmatrix} c_{21} \\ c_{22} \\ \vdots \\ c_{2,n-1} \end{bmatrix} + m(z^0, c_1, T) = \begin{bmatrix} z_2^d \\ z_3^d \\ \vdots \\ z_n^d \end{bmatrix},$$

where $M(c_1, T)$ is a nonsingular matrix, provided that

$$c_1(t) \not\equiv 0, \qquad \text{for some} \quad t \in [0, T]. \tag{11}$$

The control law (eqs. (8) and (9)) produces smooth robot trajectories (Tilbury, Murray, and Sastry 1993), but has a fundamental drawback: being computed in open loop, it is not robust. As a consequence, it will not yield accurate steering when applied to a system whose model is different from the nominal chained form of eq. (1) (e.g., the perturbed chained forms of eqs. (6) and (7)).

However, it is possible to achieve robustness with respect to repetitive disturbances and model perturbations by computing the control parameters (in this case, $c_1(t)$ and $c_{21}, \ldots, c_{2,n-1}$) through a learning scheme. In the next section, we show how this can be done with reference to a control law that slightly generalizes eqs. (8) and (9).

3. The Proposed Learning Method

Assume that a parameterized class of inputs $v(t, c) : [0, T] \times \mathbb{R}^r \mapsto \mathbb{R}^2$ has been chosen, where $c \in \mathbb{R}^r$ is the control parameter vector. The objective of the learning algorithm is to perform an iterative, robust inversion of the relationship that associates the control parameters c with the final state $z(T)$. To this end, repeated experiments of duration T are executed: at the end of the kth experiment, the positioning error $z^d - z^{[k]}(T)$ is computed, and a new control parameter vector $c^{[k+1]}$ is generated accordingly.

For convenience, partition the state vector z as (z_a, z_b), with $z_a = z_1 \in \mathbb{R}^1$ and $z_b = (z_2, \ldots, z_{n-1}) \in \mathbb{R}^{n-1}$. System (1) is conveniently rewritten as

$$\dot{z}_a(t) = v_1(t), \tag{12}$$

$$\dot{z}_b(t) = A(t)z_b(t) + B\,v_2(t), \tag{13}$$

with

$$A(t) = \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 \\ v_1(t) & 0 & 0 & \ldots & 0 \\ 0 & v_1(t) & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \ldots & 0 & v_1(t) & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{14}$$

During each iteration of the learning algorithm, we make use of a control law that is a generalization of eqs. (8) and (9). Divide the steering-time interval $[0, T]$ into p subintervals $[t_{i-1}, t_i]$, $i = 1, \ldots, p$, with $t_0 = 0$ and $t_p = T$. Let the velocity inputs during the kth iteration be expressed in the parametric form

$$v_1^{[k]}(t) = c_{1i}^{[k]}, \quad t \in [t_{i-1}, t_i),\ i = 1, \ldots, p, \tag{15}$$

$$v_2^{[k]}(t) = \sum_{j=1}^{q} c_{2j}^{[k]}\lambda_j(t), \quad t \in [0, T], \tag{16}$$

where $c_{1i}^{[k]}, c_{2j}^{[k]} \in \mathbb{R}$, $i = 1, \ldots, p$, $j = 1, \ldots, q$, while $\lambda_j(t)$, $j = 1, \ldots, q$ are assigned piecewise-polynomial functions with discontinuities occurring only at the time instants t_i (see Fig. 2). For example, one may set

$$\lambda_j(t) = \sum_{i=1}^{p} \pi_i(t), \qquad j = 1, \ldots, q,$$

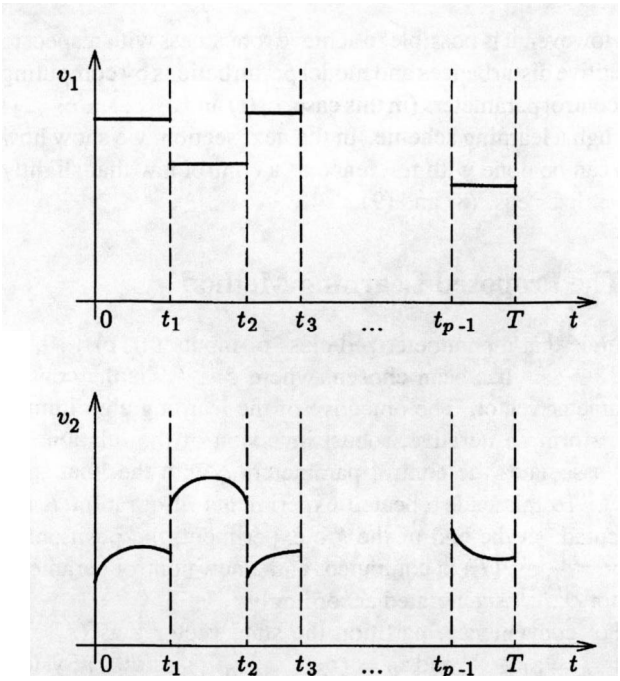

Fig. 2. The profile of the inputs v_i within each iteration of the learning algorithm.

with $\pi_i(t)$ as a polynomial function such that $\pi_i(t) \neq 0$ for $t \in [t_{i-1}, t_i)$ and $\pi_i(t) = 0$ elsewhere. We shall assume that the $\lambda_j(t)$ functions (also called *base* functions) are linearly independent over $[0, T]$, as elements of the linear space of piecewise-polynomial functions. The learning process consists of the computation of the two vectors of control coefficients $c_1 \in \mathbb{R}^p$ and $c_2 \in \mathbb{R}^q$ by repeated trials.

Some comments are in order with reference to the above structure of inputs:

- As the first subsystem (eq. (12)) is a simple integrator, a single parameter is sufficient to obtain convergence of z_1 to its desired value z_1^d by satisfying eq. (10). The value of v_1 in the other subintervals can therefore be set to arbitrary constant values which are *not* updated during the learning process. Hence, without loss of generality, we may set

$$v_1^{[k]}(t) = \begin{cases} c_{11}^{[k]} & t \in [0, t_1), \\ c_{12} & t \in [t_1, t_2), \\ \vdots & \\ c_{1p} & t \in [t_{p-1}, T]. \end{cases} \tag{17}$$

  Note, however, that to meet condition (11), at least one of the coefficients $c_{12}, \ldots, c_{1p}$ should be nonzero in general, even if no reconfiguration is required on the first variable z_a. In view of the structure of matrix A in eq. (14), this guarantees that the other variables z_b can be reconfigured as desired. As a consequence, it must be $p \geq 2$ in general, while $p = 1$ is admissible if $z_1^0 \neq z_1^d$.
- With reference to the previous remark, it should be evident that although the values of the constants $c_{12}, \ldots, c_{1p}$ are arbitrary, their choice will affect the shape of the obtained system trajectories. The same is true for the choice of the piecewise-polynomial functions. However, simple considerations may be used to help in the choice. In Section 4, it will be shown how this can be done for the car-like robot.
- The dimension q of the control coefficient c_2 must be at least $n - 1$. This is easily established by noting that (1) the second subsystem (eq. (12)) is controllable with the choice of eq. (17) for $v_1(t)$; and (2) its dimension is $n - 1$. This implies (Sontag 1990, Proposition 3.7.1) that there exists some piecewise-polynomial control that transfers the system state z_2 from z_2^0 to z_2^d in a finite time T. In particular, a unique solution exists if $c_2 \in \mathbb{R}^{n-1}$, while infinite solutions can be found if the dimension of c_2 is larger.

In the following, we shall assume that the minimum possible number of control coefficients has been chosen; i.e., $q = n-1$. The possibility of overparameterizing the controller will be discussed in the concluding section.

We now present a learning control algorithm for a special class of linear time-varying systems that encompasses subsystem (13) as a particular case. Later, it will be shown how such an algorithm can be embedded in a global learning structure for the whole system (eqs. (12) and (13)), or equivalently, for system (1).

3.1. *Learning Control for Linear Systems*

Consider the special class of linear time-varying systems of the form

$$\dot{x}(t) = A(t)x(t) + Bu(t), \quad x \in \mathbb{R}^N, u \in \mathbb{R}^M, t \in [0, T], \tag{18}$$

such that

$$A(t) = A_i, \qquad t \in [t_{i-1}, t_i), i = 1, \ldots, p,$$

where $t_0 = 0$, $t_p = T$, and $\{A_1, A_2, \ldots, A_p\}$ is a given sequence of constant matrices. We shall assume that the pair (A_i, B) is controllable for all i. Our objective is to steer system (18) from an initial state $x(0) = x^0$ to a desired state x^d in a finite time T. Note that within each time interval, the control system (eq. (18)) is linear time-invariant.

Assume that u belongs to a finite-dimensional subspace of the linear space of piecewise-continuous functions:

$$u(t) = \sum_{j=1}^{N} c_j \lambda_j(t), \tag{19}$$

where $c_j \in \mathbb{R}$, $j = 1, \ldots, N$, and $\lambda_j(t) : [0, T] \mapsto \mathbb{R}^M$, $j = 1, \ldots, N$ are linearly independent, piecewise-continuous vector functions with discontinuities occurring only at the time instants t_i. Typical choices for the base function λ_j's include piecewise-constant, piecewise-polynomial, and piecewise-exponential functions. Note that the dimension of the coefficient vector $c = (c_1, \ldots, c_N)$ is exactly the dimension of the state space of system (18).

As previously noted, the controllability assumption guarantees that in the chosen class there exists a unique control $u(t)$ achieving the desired state x^d in a finite time T. In turn, this identifies a unique value for the control coefficient vector c, which we intend to compute via learning.

Defining $\Delta_i = t_i - t_{i-1}$, for $i = 1, \ldots, p$, we have

$$\begin{aligned} x(t_i) &= e^{A_i \Delta_i} x(t_{i-1}) + \sum_{j=1}^{N} c_j \int_{t_{i-1}}^{t_i} e^{A_i (t_i - \tau)} B \lambda_j(\tau) d\tau \\ &= V_i\, x(t_{i-1}) + W_i\, c, \end{aligned} \tag{20}$$

with V_i and W_i two constant $N \times N$ matrices given by

$$V_i = e^{A_i \Delta_i}, \tag{21}$$

$$W_i = \int_0^{\Delta_i} e^{A_i (\Delta_i - \tau)} B [\lambda_1(t_{i-1} + \tau) \ \ldots \ \lambda_q(t_{i-1} + \tau)] d\tau, \tag{22}$$

for $i = 1, \ldots, p$. Applying recursively eq. (20), we obtain at time $t_p = T$

$$x(T) = V x^0 + W c, \tag{23}$$

where the expression of the constant matrices V and W is easily obtained as

$$V = V_p V_{p-1} \ldots V_1, \tag{24}$$

$$W = W_p + V_p W_{p-1} + \ldots + V_p \ldots V_2 W_1. \tag{25}$$

Equation (23) provides the basis for our learning algorithm. Denote by x^d the final desired state at time T, and assume that at each iteration the control system (eq. (18)) is exactly reinitialized at x^0. For the kth iteration, the control input (eq. (19)) is written as

$$u^{[k]}(t) = \sum_{j=1}^{N} c_j^{[k]} \lambda_j(t), \tag{26}$$

while the final system-positioning error is

$$\varepsilon^{[k]} = x^d - x^{[k]}(T).$$

Henceforth, we shall drop the time dependence for quantities measured at the end of the time interval $[0, T]$. For example, $x^{[k]} = x^{[k]}(T)$.

PROPOSITION 1. Let the coefficient vector be updated according to

$$c^{[k+1]} = c^{[k]} + F \varepsilon^{[k]}, \tag{27}$$

where F is an $N \times N$ constant matrix such that the eigenvalues of matrix $I - WF$ lie in the open unitary disk of the complex plane. Then, the final positioning error ε converges to zero over the iterations, uniformly and exponentially.

Proof. Simple computations show that the error dynamics over successive iterations takes the form

$$\varepsilon^{[k+1]} = (I - WF) \varepsilon^{[k]}. \tag{28}$$

Hence, under the hypothesis of the proposition we have that $\varepsilon^{[k]} \to 0$ as $k \to \infty$, uniformly and exponentially. The existence of a matrix F satisfying the hypothesis—and in particular, the invertibility of matrix W—is guaranteed by the controllability assumption on system (18). □

Since matrix W is invertible, one may set $F = \nu W^{-1}$, with $0 < \nu < 2$ to achieve an arbitrary rate of convergence. In particular, the choice $\nu = 1$ yields algorithm convergence in two iterations. This is true also in the presence of *repetitive* disturbances, i.e., disturbances that act on the system in an identical fashion during each iteration. For example, this is the case of a constant error on the initial condition x^0. In practice, however, the presence of model perturbations on the nominal system (eq. (18)) requires the execution of multiple iterations.

The robustness of our learning controller with respect to model perturbations is assessed in the following proposition. The proof is based on a well-known stability result, namely, the fact that, being that the discrete-time error dynamics (eq. (28)) are uniformly and exponentially stable, small nonpersistent perturbations are completely rejected. During the proof we make use of a technical lemma that is reported in the Appendix for compactness.

PROPOSITION 2. Assume that system (18) is perturbed as

$$\dot{x}(t) = A(t)x(t) + Bu(t) + \gamma\, \eta(x(t), u(t), t), \quad t \in [0, T], \tag{29}$$

where $\gamma > 0$ is a small parameter and η is a continuously differentiable function of x, u, and t. Then, for sufficiently small γ, the learning algorithm (eq. (27)) designed on the basis of the nominal system (eq. (18)) gives a final positioning error ε that converges to zero over the iterations, uniformly and exponentially.

Proof. Owing to the presence of the perturbation, the system state at the end of iteration $k + 1$ can be written as

$$x^{[k+1]}(T) = V x^0 + W c^{[k+1]} + d^{[k+1]},$$

where $d^{[k+1]}$ represents the mismatch introduced at time T with respect to the unperturbed final state as expressed by eq. (23). Note that such a mismatch depends on the iteration through the particular state trajectory that the system follows.

The final system positioning error is

$$\varepsilon^{[k+1]} = x^d - x^{[k+1]}(T) = x^d - Vx^0 - W(c^{[k]} + F\varepsilon^{[k]}) - d^{[k+1]},$$

where eq. (27) has been used. Being that

$$\varepsilon^{[k]} = x^d - Vx^0 - Wc^{[k]} - d^{[k]},$$

we have

$$\varepsilon^{[k+1]} = (I - WF)\varepsilon^{[k]} + d^{[k]} - d^{[k+1]}. \tag{30}$$

Hence, the perturbation on the error dynamics at iteration $(k+1)$ is

$$\tilde{\eta}^{[k]} = d^{[k]} - d^{[k+1]}. \tag{31}$$

We now invoke Lemma 1 (see the appendix) according to which $\tilde{\eta}^{[k]}$ satisfies

$$||\tilde{\eta}^{[k]}|| \leq g||\varepsilon^{[k]}||, \tag{32}$$

with $g \leq 1$ for γ sufficiently small. Therefore, $\tilde{\eta}$ is a *non-persistent* perturbation on the error dynamics (Hahn 1967, Theorem 56.2). As the origin of the unperturbed error dynamics (eq. (28)) is uniformly exponentially stable, the same is true for the perturbed error dynamics (eq. (30)). □

At this point, one could wonder what happens when the parameter γ in eq. (29) is so large that the resulting perturbation $\tilde{\eta}$ on the error dynamics is *persistent*, i.e., it does *not* satisfy the estimate (eq. (32)). Again, classical stability results (Hahn 1967, Theorem 56.4) can be used to show that bounded persistent perturbations give rise to bounded errors (*total stability*). In conclusion, the proposed learning algorithm provides robust convergence in the presence of small model perturbations.

3.2. Application to Chained-Form Systems

We can build an iterative learning controller for the chained-form system (1), or, equivalently, for system (12-13), by interleaving learning phases on the two subsystems. In fact, having chosen v_1 in eq. (15) (or, eq. (17)) as a piecewise-constant function, the second subsystem (eq. (13)) has exactly the linear time-varying structure of eq. (18) and is controllable. The same is trivially true for the first subsystem (eq. (12)), which is a simple integrator.

The learning algorithm proceeds as follows.

STEP 0

Set $k = 1$. Use arbitrary values $c_{11}^{[1]}$ and $c_{21}^{[1]}, \ldots, c_{2,n-1}^{[1]}$ to initialize the control coefficients c_{11} (remember that $c_{12}, \ldots, c_{1p}$ are kept constant) and $c_{21}, \ldots, c_{2,n-1}$, respectively. A reasonable choice is to use the values computed for the nominal system through a procedure similar to the one outlined at the end of Section 2.2.

STEP 1

Perform the kth experiment using the control inputs given by eqs. (17) and (16), and measure the final positioning error $\varepsilon^{[k]} = (\varepsilon_a^{[k]}, \varepsilon_b^{[k]}) = z^d - z^{[k]}(T)$.

STEP 2a

Apply eq. (27) to the first subsystem (eq. (12)) to obtain the update rule for the control coefficient c_{11}. This gives

$$c_{11}^{[k+1]} = c_{11}^{[k]} + \nu_a \varepsilon_a^{[k]}/\Delta_1, \tag{33}$$

with $0 < \nu_a < 2$. Through eq. (17), this determines the control law $v_1^{[k+1]}(t)$ to be used in the next iteration.

STEP 2b

Use the value of $v_1^{[k+1]}(t)$ to build the "current" matrices $A(v_1^{[k+1]})$, $V(v_1^{[k+1]})$, and $W(v_1^{[k+1]})$ (given, respectively, by eq. (14), eq. (24), and eq. (25)) for the second subsystem (eq. (13)). For compactness, let $V^{[k+1]} = V(v_1^{[k+1]})$ and $W^{[k+1]} = W(v_1^{[k+1]})$. The second control coefficient c_2 is then updated as

$$c_2^{[k+1]} = c_2^{[k]} + \left(W^{[k+1]}\right)^{-1} \nu_b \varepsilon_b^{[k]} - \left(W^{[k+1]}\right)^{-1} \cdot \left[\Delta V^{[k+1]} z_b^0 + \Delta W^{[k+1]} c_2^{[k]}\right], \tag{34}$$

with $0 < \nu_b < 2$, $\Delta V^{[k+1]} = V^{[k+1]} - V^{[k]}$, and $\Delta W^{[k+1]} = W^{[k+1]} - W^{[k]}$. Through eq. (16), this determines the control law $v_2^{[k+1]}(t)$ to be used in the next iteration. Note that a third term has been added in eq. (34) with respect to the basic update law (eq. (27)). Such a term implements a feedforward action that compensates the effect of v_1 changing from iteration to iteration.

STEP 3

Set $k = k + 1$ and go to Step 1.

The actual expressions of matrices $V^{[k+1]}$ and $W^{[k+1]}$ in eq. (34) depend on the choices of the piecewise-constant function $v_1(t)$ in eq. (17) and the piecewise-polynomial functions $\lambda_j(t)$ in eq. (16). However, the involved computations are usually very simple. An example is provided in Section 4.

The correctness of the algorithm is proven next.

PROPOSITION 3. Using the above learning controller, the final positioning error $\varepsilon = z^d - z(T)$ of the chained form system (eqs. (12) and (13)) converges asymptotically to zero over the iterations, uniformly and exponentially.

Proof. Since the evolution of the first variable $z_a = z_1$ is not influenced by the second subsystem, $v_1^{[k]}(t)$ converges to the "correct" $v_1^*(t)$, which steers z_1 to its desired value z_1^d. Hence, ε_a converges uniformly and exponentially to zero. As

for the second subsystem, we have at the end of iteration $k+1$

$$z_b^{[k+1]} = V^{[k+1]} z_b^0 + W^{[k+1]} c_2^{[k+1]},$$

so that

$$\varepsilon_b^{[k+1]} = z_b^d - z_b^{[k+1]} = z_b^d - V^{[k+1]} z_b^0 - W^{[k+1]} c_2^{[k+1]}.$$

Hence, using the update law (eq. (34)) for the control coefficient c_2, we obtain

$$\begin{aligned}\varepsilon_b^{[k+1]} = {} & z_b^d - V^{[k+1]} z_b^0 - W^{[k+1]} c_2^{[k]} - \nu_b \varepsilon_b^{[k]} \\ & + \Delta V^{[k+1]} z_b^0 + \Delta W^{[k+1]} c_2^{[k]}.\end{aligned} \tag{35}$$

Recalling that $V^{[k+1]} = V^{[k]} + \Delta V^{[k+1]}$ and $W^{[k+1]} = W^{[k]} + \Delta W^{[k+1]}$, eq. (35) yields

$$\varepsilon_b^{[k+1]} = z_b^d - V^{[k]} z_b^0 - W^{[k]} c_2^{[k]} - \nu_b \varepsilon_b^{[k]},$$

and finally,

$$\varepsilon_b^{[k+1]} = (1 - \nu_b) \varepsilon_b^{[k]}. \tag{36}$$

This shows that the positioning-error dynamics for the second subsystem are completely decoupled from those of the first subsystem. Since $0 < \nu_b < 2$, we conclude that ε_b also converges uniformly and exponentially to zero. □

As for the robustness of the above learning controller, it is easy to realize that the proof of Proposition 3 still holds for *perturbed* chained systems of the form

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \\ \vdots \\ \dot{z}_n \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ z_2 v_1 \\ \vdots \\ z_{n-1} v_1 \end{bmatrix} + \gamma \begin{bmatrix} \eta_1(z, v) \\ \eta_2(z, v) \\ \eta_3(z, v) \\ \vdots \\ \eta_n(z, v) \end{bmatrix}, \tag{37}$$

in which the perturbation on the first variable z_1 depends neither on the other variables $z_2, \ldots, z_n$ nor on the second control input v_2. In fact, in this case the triangular structure of the system is preserved and Proposition 2 may be applied to each subsystem. Therefore, the proposed learning algorithm yields convergence to zero of the whole positioning error in spite of the presence of small perturbations. Note that for the car-like robot, both the perturbed models (eqs. (6) and (7)) are of the form of eq. (37); therefore, the presented algorithm provides robust steering in such cases. We have verified that many other perturbations of the car-like robot satisfy the above requirement.

4. Simulation Results

To test the performance of the proposed learning algorithm, the latter was applied to the car-like robot (eq. (3)) with nominal parameters $\ell = 0.2$ m and $\rho = 0.02$ m. In simulation, we have used two perturbed parameters, $\tilde{\ell} = 0.22$ m and $\tilde{\rho} = 0.022$ m, i.e., values 10% larger than the nominal values, and we have set $\nu_a = \nu_b = 1$ in eqs. (33) and (34). Both of these perturbations should be rejected by the learning algorithm designed for the nominal system. In all simulations, the initial values of the control coefficients to be used in the first iteration have been computed by simply performing a fictitious "zero" iteration with c_1 and c_2 completely null; this implies $\varepsilon^{[1]} = z^d - z^0$ in eqs. (33) and (34).

We have run three simulations with $T = 5$ sec. In the first, we chose $q^0 = (-1, 0, \pi/4, 0)$ and $q^d = (0, 0, \pi/4, 0)$ (throughout the paper, distances and angles are respectively expressed in meters and radians). This motion task requires a net displacement in the x-direction, and is therefore called a *forward parking*. Since $z_1^0 \neq z_1^d$, a single constant value c_1 can be used for v_1 in eq. (17) (see the related remark in Section 3.1); this is consistent with the fact that no inversion of motion is needed to perform the task. The structure of the steering-control law is

$$\begin{aligned} v_1(t) &= c_{11}, \\ v_2(t) &= c_{21} + c_{22} t + c_{23} t^2, \end{aligned}$$

for $t \in [0, 5]$. A comparison with the general form of v_2 in eq. (16) shows that the base functions are $\lambda_1 = 1, \lambda_2 = t$, and $\lambda_3 = t^2$. These functions have been chosen to be continuous over $[0, T]$ in view of the continuity of v_1 (see Fig. 2).

The update of the control coefficients c_{11} and c_{21}, c_{22}, c_{23} is performed according to eqs. (33) and (34). In the former, we have $\Delta_1 = T = 5$. For the latter, by using eqs. (24) and (25), it is easy to derive the following expressions:

$$V^{[k]} = \begin{bmatrix} 1 & 0 & 0 \\ c_{11}^{[k]} T & 1 & 0 \\ (c_{11}^{[k]})^2 T^2/2 & c_{11}^{[k]} T & 0 \end{bmatrix},$$

$$W^{[k]} = \begin{bmatrix} T & T^2/2 & T^3/3 \\ c_{11}^{[k]} T^2/2 & c_{11}^{[k]} T^3/6 & c_{11}^{[k]} T^4/12 \\ (c_{11}^{[k]})^2 T^3/6 & (c_{11}^{[k]})^2 T^4/24 & (c_{11}^{[k]})^2 T^5/60 \end{bmatrix},$$

which are valid for the kth iteration.

The obtained results are summarized in Figures 3–6. Four iterations are needed to achieve exact steering. Figure 3 is a stroboscopic view of the robot motion during the final iteration; for comparison, the configuration at the end of the first iteration is also shown. The evolution of the generalized coordinates during the fourth iteration is reported in Figure 4, while the profiles of the actual control inputs u_1 and u_2 (which are related to the chained-form control inputs through eqs. (4) and (5)) are given in Figure 5. Finally, Figure 6 shows the evolution of the Cartesian positioning error $\sqrt{(x_r(T) - x_r^d)^2 + (y_r(T) - y_r^d)^2}$ and the orientation error $\theta(T) - \theta^d$ over the iterations. Note the exponential convergence to the desired configuration.

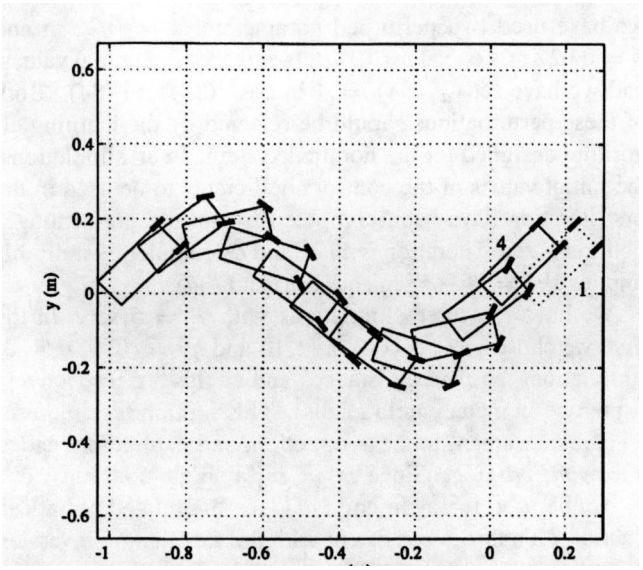

Fig. 3. First simulation: trajectory during the fourth iteration. The configuration at the end of the first iteration is also shown (dotted).

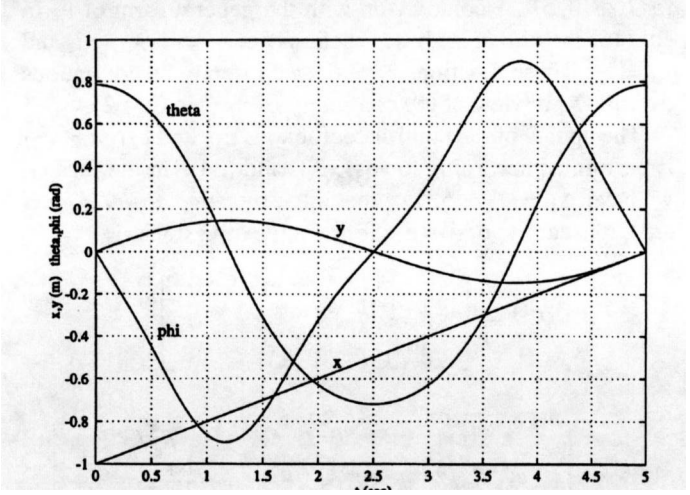

Fig. 4. First simulation: generalized coordinates during the fourth iteration.

In the second simulation, the robot is assigned the same motion task, but a steering-angle saturation (at 45°) has been introduced to account for a typical limitation of actual vehicles. The results of the learning algorithm after seven iterations are shown in Figures 7–10. In particular, note the saturation of ϕ in Figure 8. The successful outcome shows that the learning algorithm is also able to cope with state constraints.

For the third case, we have chosen $q^0 = (0, 0.5, 0, 0)$ and $q^d = (0, 0, 0, 0)$. This motion task requires a net displacement in a direction that is orthogonal to the vehicle's

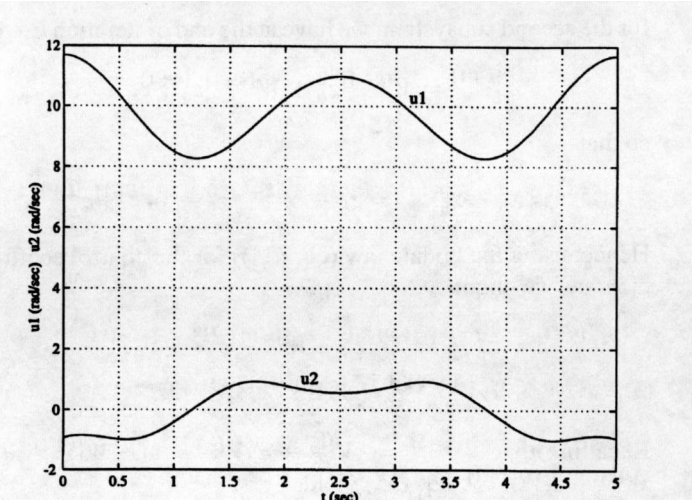

Fig. 5. First simulation: control inputs u_1 and u_2 during the fourth iteration.

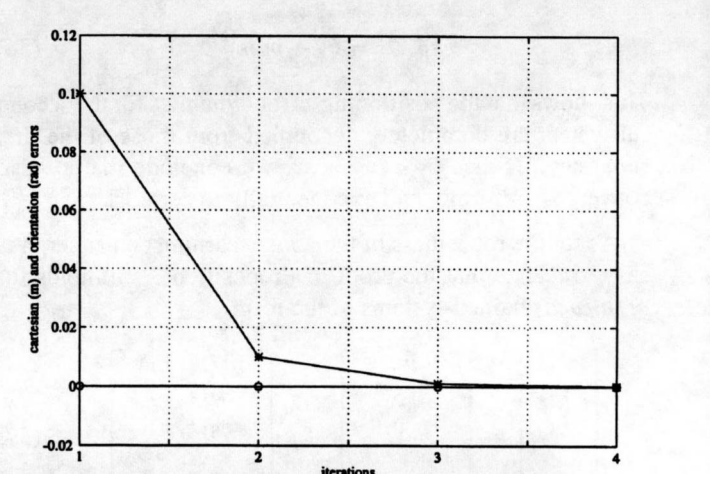

Fig. 6. First simulation: evolution of the Cartesian ($*$) and the orientation ($\circ$) errors over the iterations.

initial orientation, and is therefore called a *parallel parking*. Since $z_1^0 = z_1^d$, $p \geq 2$ coefficients must be used in eq. (17), one of which is computed through the learning procedure, namely, c_{11}. In particular, we have set $p = 2$ and $\Delta_1 = \Delta_2 = 2.5$, thereby obtaining the following structure for the steering-control law:

$$v_1(t) = \begin{cases} c_{11} & t \in [0, 2.5), \\ c_{12} & t \in [2.5, 5], \end{cases}$$

$$v_2(t) = \begin{cases} c_{21} + c_{22}t + c_{23}t^2 & t \in [0, 2.5), \\ c_{21} + c_{22}(t - 2.5) + c_{23}(t - 2.5)^2 & \\ & t \in [2.5, 5]. \end{cases}$$

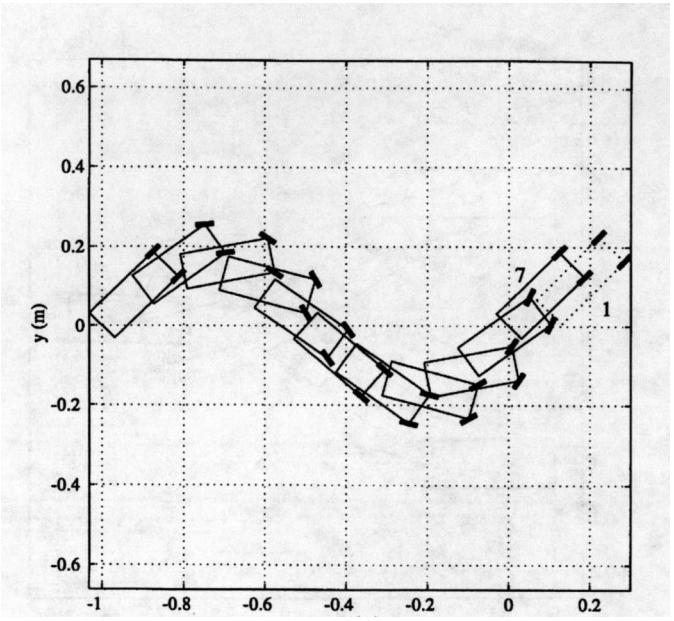

Fig. 7. Second simulation: trajectory during the seventh iteration. The configuration at the end of the first iteration is also shown (dotted).

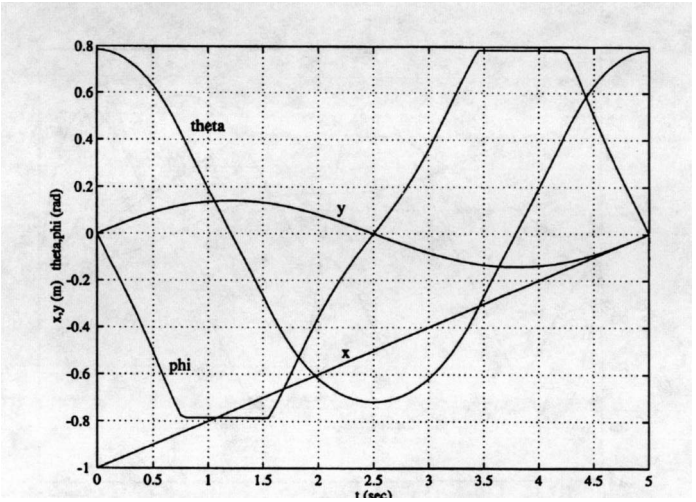

Fig. 8. Second simulation: generalized coordinates during the seventh iteration.

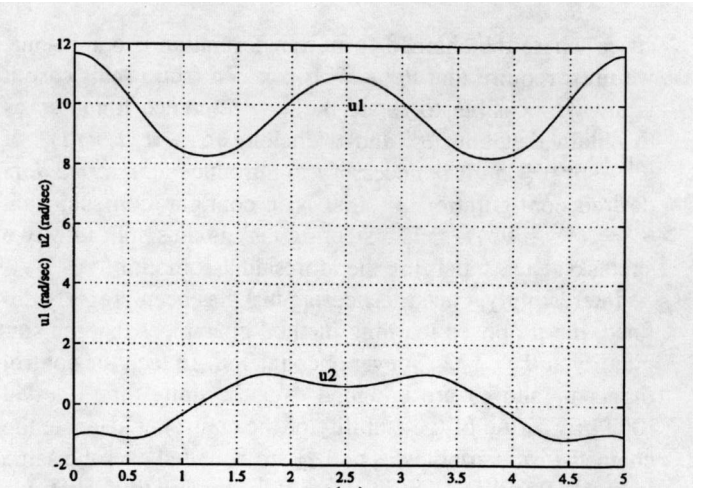

Fig. 9. Second simulation: control inputs u_1 and u_2 during the seventh iteration.

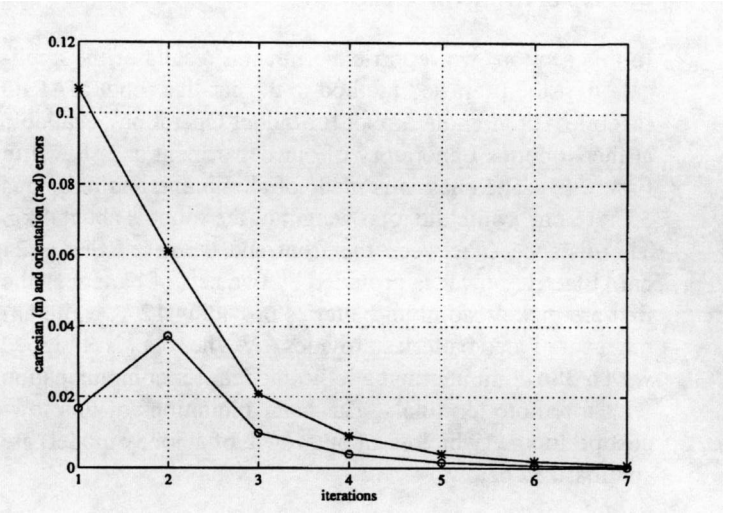

Fig. 10. Second simulation: evolution of the Cartesian (∗) and the orientation (o) errors over the iterations.

The above choice for $v_1(t)$ will produce a natural forward/backward maneuver. Accordingly, $v_2(t)$ has been built by patching together two second-order polynomials, with a single discontinuity at $t_1 = 2.5$.

The effect of a particular choice for the constant coefficient c_{12} can be predicted with the aid of eq. (12). In fact, being $z_1 = x_r$, such an equation implies that the *nominal* distance traveled along the x-axis during the second part of the movement (i.e., the backward motion) is $|c_{12}| \cdot \Delta_2$. Therefore, by changing the value of c_{12}, one directly controls the amplitude of the parking maneuver. Here, we have used $c_{12} = -0.24$, so as to obtain a nominal length of $0.24 \cdot 2.5 = 0.6$ m for the backward path. The actual length will differ slightly from the nominal value, due to the presence of a perturbation on the wheel's radius.

The update of the control coefficients c_{11} and c_{21}, c_{22}, c_{23} is performed according to eqs. (33) and (34). As for the first simulation, the expression of $V^{[k]}$ and $W^{[k]}$ to be used in eq. (34) can be easily obtained. The results for this simulation are reported in Figures 11–14. Five iterations are needed to achieve exact steering.

We conclude this section by pointing out that in view of the use of the chained-form transformation, some care must be taken in the definition of the steering task. To guarantee that both q^0 and q^d belong to a region of the configuration

space where the chained-form transformation is one-to-one, we must require that $|\theta^0 - \theta^d| < \pi$. In fact, in this case it is always possible to rotate the x–y reference frame so as to obtain that both θ^0 and θ^d belong to $(-\pi/2, \pi/2)$. If $|\theta^0 - \theta^d| = \pi$, it is necessary to introduce an *intermediate* desired configuration q^i, that is, a configuration such that $\theta^i = (\theta^0 + \theta^d)/2$. The steering task is thus split into two subtasks, each satisfying the aforesaid assumption.

Interestingly, once the steering task has been properly defined, the proposed learning method guarantees that the singularity at $\theta = \pi/2$ is never encountered. In fact, the control inputs v_1 and v_2 are bounded over the finite time interval $[0, T]$ by virtue of the stability of the controller. Hence, the chained-form variables z_2 and z_3 are bounded over the same interval. Recalling that $z_3 = \tan\theta$ for the car-like robot, we conclude that $|\theta|$ is bounded below $\pi/2$.

5. Experimental Results

In this section, we report experimental results of the application of the proposed method to the car-like robot MARIO (Mobile Autonomous Robot for Indoor Operation), available at the Robotics Laboratory of our department and shown in Figure 15. The chassis is made of aluminum, and measures 35 ×16 cm, while the total weight of the robot is about 4 kg. The distance ℓ between the front and the rear axles is 24 cm. Electric power is provided by two sets of batteries: the first are nickel-cadmium batteries providing 12 V, while another set of lead batteries provides 6 V. The robot is equipped with a 386 computer and a radio-modem for communication with a remote terminal. The main limitations of this low-cost prototype (which is an outgrowth of a hobby model) are summarized below.

- Two electric motors are used to drive the rear wheels and to steer the front wheels. The first has an external velocity-regulation loop, while only the built-in feedback loop is available for the second, whose operation is therefore less accurate. Also, the existing linkage allows a maximum steering angle of 25°.
- Most mechanical components (gearbox, shaft, joints) are realized in plastic material without ball bearings; as a consequence, effects such as friction and backlash are quite relevant.
- The wheels, having a radius of 3.65 cm and a width of 2 cm, are made of a particularly soft rubber, which is subject to deformations under the weight of the vehicle.
- The measure of the configuration's variables (x_r, y_r, θ, ϕ) (which is required to perform the input transformation, eqs. (4) and (5)), is reconstructed via dead reckoning on the basis of the information provided by the rear wheels' encoders. These introduce a quantization effect (40 pulses per turn), and do not account for slippage effects.

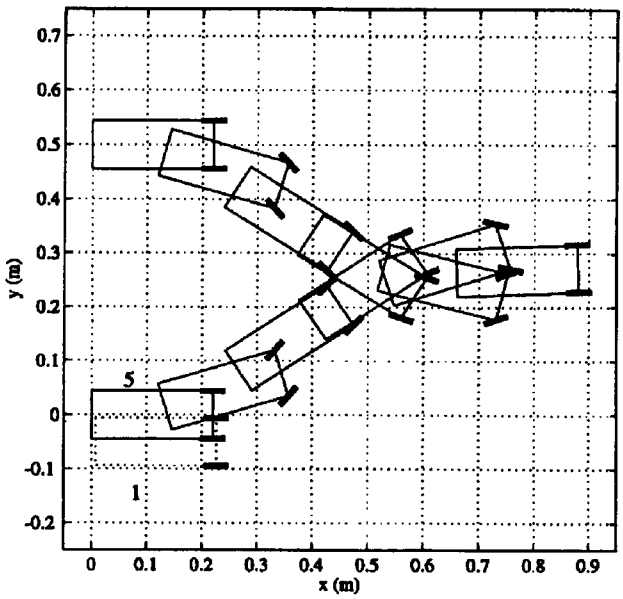

Fig. 11. Third simulation: trajectory during the fifth iteration. The configuration at the end of the first iteration is also shown (dotted).

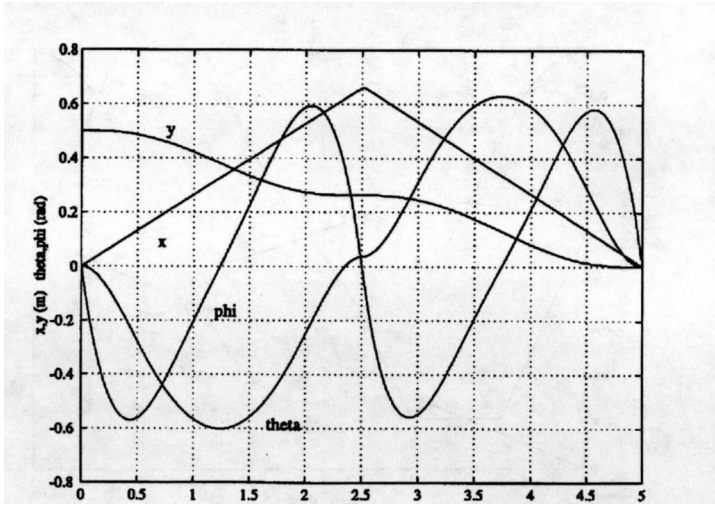

Fig. 12. Third simulation: generalized coordinates during the fifth iteration.

As will be shown below, the performance of the proposed learning method is satisfactory in spite of these nonideal conditions.

At the end of each iteration, the configuration of MARIO is measured by means of an external localization system. In fact, although the learning algorithm can reject the perturbation deriving from erroneous estimates of the configuration's variables during the motion, an accurate measure of the final positioning error $q(T) - q^d$ is required.

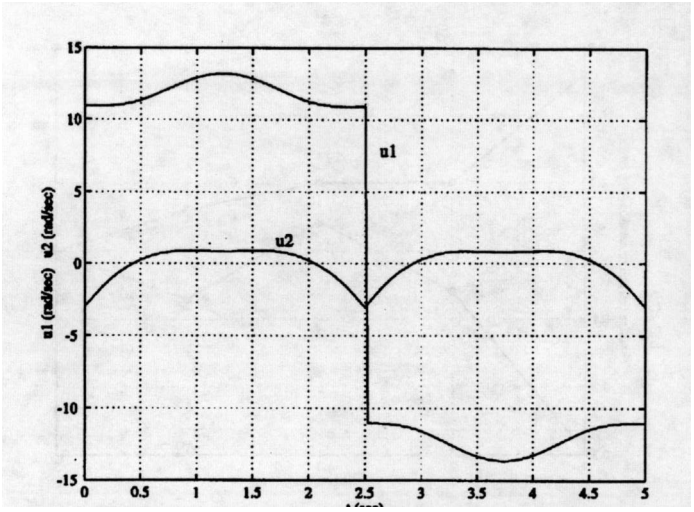

Fig. 13. Third simulation: control inputs u_1 and u_2 during the fifth iteration.

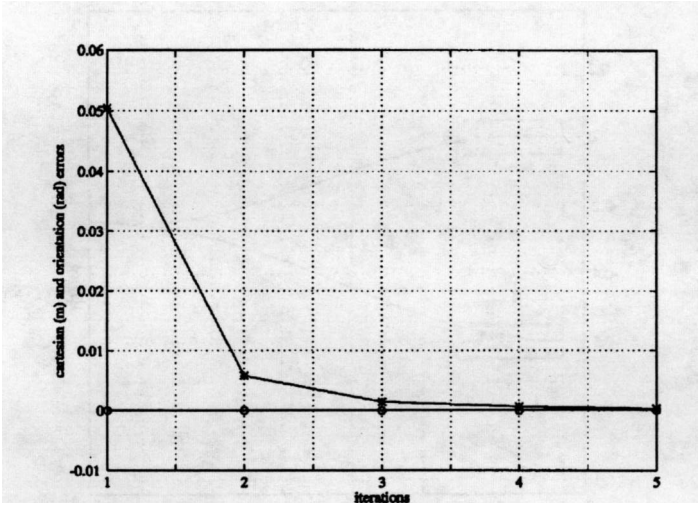

Fig. 14. Third simulation: evolution of the Cartesian (∗) and the orientation (∘) errors over the iterations.

To improve the behavior of the proposed control scheme, we have introduced a slight modification with respect to the algorithm so far discussed. In particular, piecewise-trapezoidal profiles have been used for the first control input v_1 in place of the piecewise-constant law (eq. (17)), the rationale being simply that the corresponding velocity profiles for u_1 are continuous over time and hence can be tracked much more accurately by the driving velocity servo. The main implication of this modification for the learning algorithm is that matrix $A(t)$ in eq. (14) becomes piecewise linear in t for $t \in [0, T]$; therefore, the structures of V_i and W_i are more complicated than in eqs. (21) and (22). Although we shall not provide explicit expressions here, they can be easily obtained by using elementary formulas for linear time-varying systems.

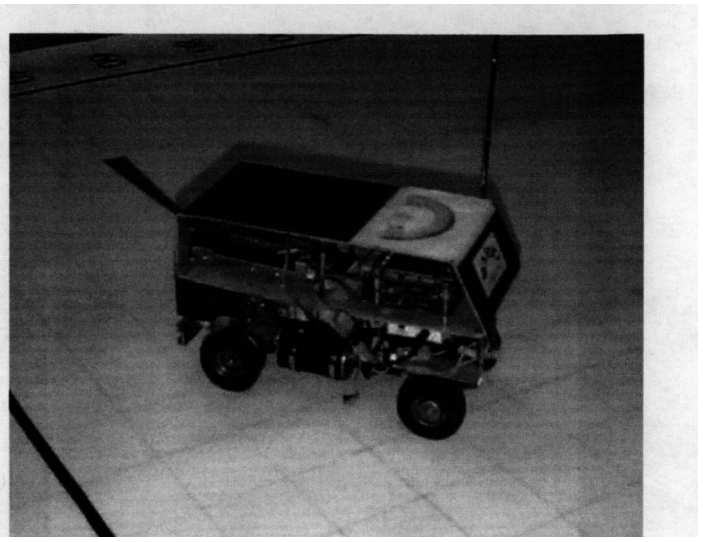

Fig. 15. The prototype car-like robot, MARIO.

In the first experiment, MARIO must perform a forward-parking task in $T = 12$ sec from the initial configuration $(0, 0, 0, 0)$ to the desired configuration $(1.80, 0.80, 0, 0)$. Similar to the first simulation of the previous section, the first control input v_1 is a *single* trapezoid. Its area is computed through the learning algorithm, while the slope in the initial and final tracts is fixed. The second control input v_2 is a simple quadratic polynomial. Five iterations are needed to achieve accurate steering. Figure 16 shows the robot motion during the final iteration, while Figure 17 displays the actual velocity control inputs u_1 and u_2. The convergence of the Cartesian and the orientation errors over the iterations is reported in Figure 18.

The second experiment is a parallel-parking task from the initial configuration $(0, 0, 0, 0)$ to the desired configuration $(0, -0.60, 0, 0)$, to be executed in $T = 10$ sec. Here, the first control input v_1 is a succession of two trapezoids, while v_2 is obtained by patching together two quadratic polynomials (compare with the third simulation in Section 4). Again, exact steering was achieved in five iterations, as shown in Figures 19–21. Note that the transition from forward to backward velocity in Figure 20 is now continuous, due to the continuity of v_1 over time.

6. Extension to Cyclic Learning Control

A possible drawback of the learning-control strategy presented thus far is that the system must be exactly reinitialized at each iteration. In practical applications, this can be done if a reliable *homing* procedure is available to bring the system—specifically, the wheeled mobile robot—back to its starting configuration q^0, before performing the next experiment.

To overcome this problem, one may devise a *cyclic* learning controller, i.e., an iterative control strategy that steers the state of the system along a sequence of desired states to be repeated

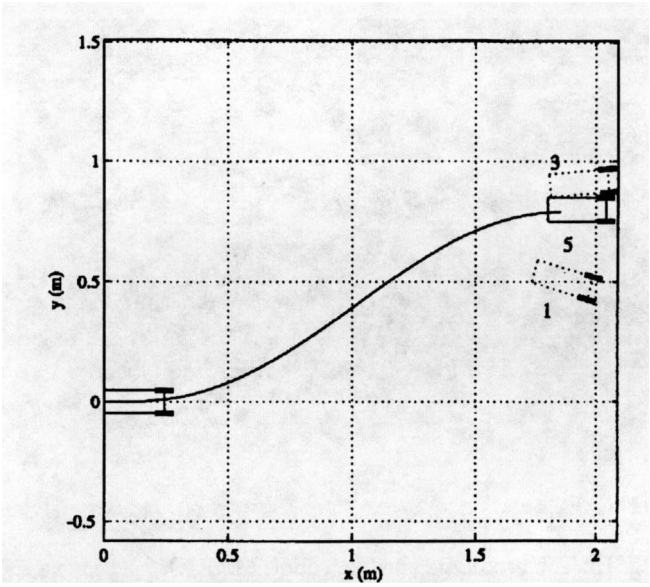

Fig. 16. First experiment: trajectory during the fifth iteration. The configurations at the end of the first and the third iterations are also shown (dotted).

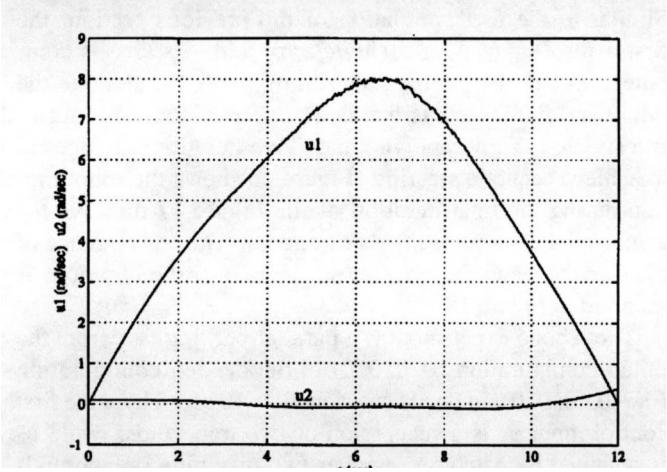

Fig. 17. First experiment: control inputs u_1 and u_2 during the fifth iteration.

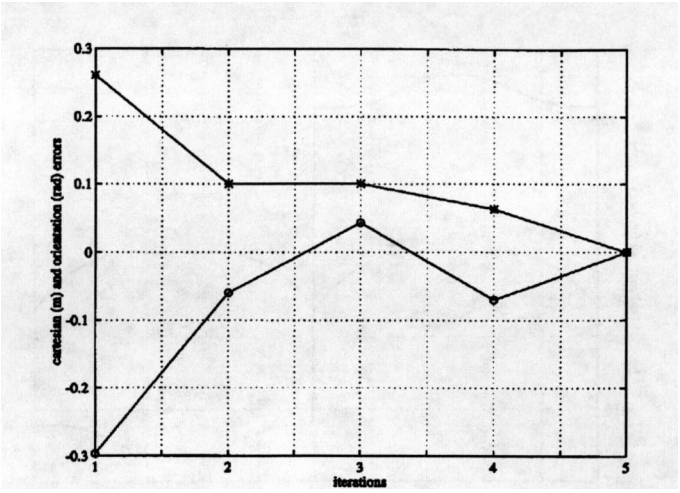

Fig. 18. First experiment: evolution of the Cartesian (∗) and the orientation (o) errors over the iterations.

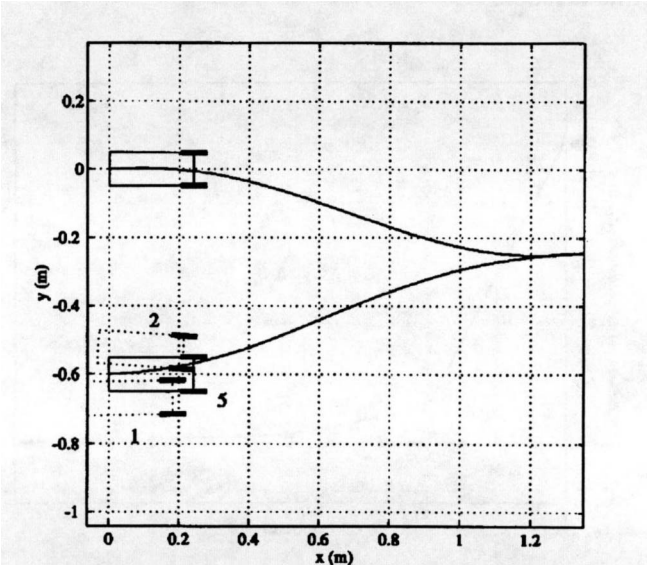

Fig. 19. Second experiment: trajectory during the fifth iteration. The configurations at the end of the first and the second iterations are also shown (dotted).

over time (Lucibello and Panzieri 1994). For compactness, we shall explain the basic idea with reference to the linear time-varying system (eq. (18)). The application to the chained-form system (eqs. (12) and (13)), or, equivalently, eq. (1), is straightforward and can be derived in a manner similar to that in Section 3.2. To show the performance of the cyclic controller, we present simulation results for the car-like robot.

6.1. Cyclic Control for Linear Systems

Assume that we wish to steer system (18) iteratively along the cyclic sequence of $r+1$ states $\{x_0^d, x_1^d, \ldots, x_{r-1}^d, x_r^d \equiv x_0^d\}$ to be attained at instants $\{\tau_0 \equiv 0, \tau_1, \ldots, \tau_{r-1}, \tau_r \equiv T\}$, with $\tau_l - \tau_{l-1} = T_l$, for $l = 1, \ldots, r$ (see Fig. 22). Analogous to Section 3.1, we suppose that each interval T_l is composed of one or more subintervals $\Delta_{l1}, \ldots, \Delta_{lp}$, with $A(t)$ constant within each subinterval.

Let

$$x_l^{[k]} = x^{[k]}(t_l), \quad l = 0, \ldots, r$$

be the system state assumed at $t = \tau_l$ during the kth iteration, and choose the steering-control input in each interval T_l as

$$u_l^{[k]}(t) = \sum_{j=1}^{N} c_{jl}^{[k]} \lambda_{jl}(t), \qquad t \in [\tau_{l-1}, \tau_l), \quad l = 1, \ldots, r$$

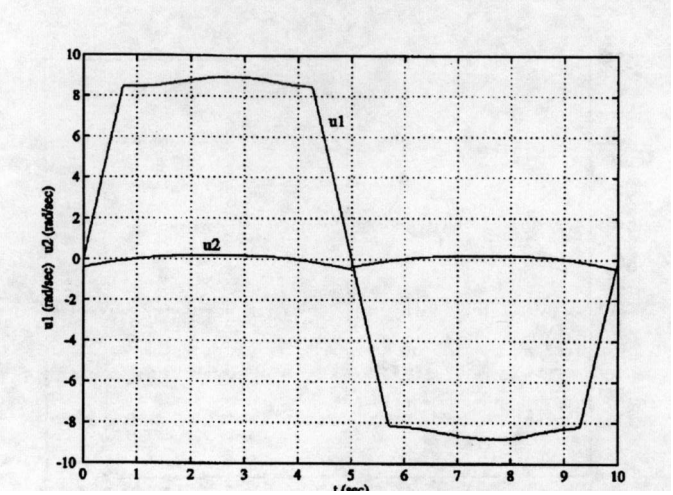

Fig. 20. Second experiment: control inputs u_1 and u_2 during the fifth iteration.

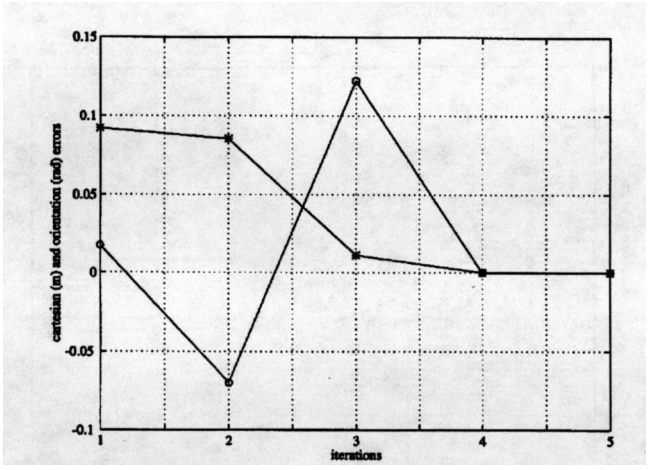

Fig. 21. Second experiment: evolution of the Cartesian ($*$) and the orientation ($\circ$) errors over the iterations.

where $c_{jl}^{[k]} \in \mathbb{R}$ and $\lambda_{jl} : [\tau_{l-1}, \tau_l] \mapsto \mathbb{R}^M$ are linearly independent, piecewise-continuous vector functions with discontinuities occurring only in correspondence with discontinuities of $A(t)$ (compare with eq. (26)).

Similar to eq. (20), one gets

$$x_l^{[k]} = \tilde{V}_l x_{l-1}^{[k]} + \tilde{W}_l c_l^{[k]}, \qquad l = 1, \ldots, r, \tag{38}$$

where $c_l^{[k]} = (c_{1,l}^{[k]}, \ldots, c_{N,l}^{[k]})$, and the expressions of matrices $\tilde{V}_l$ and $\tilde{W}_l$ can be easily obtained as in Section 3.1. Note that in eq. (38) it is $x_0^{[k]} = x_r^{[k-1]}$ in view of the cyclic nature of the task.

Define the positioning errors during the kth iteration as

$$\varepsilon_l^{[k]} = x_l^d - x_l^{[k]}, \qquad l = 1, \ldots, r.$$

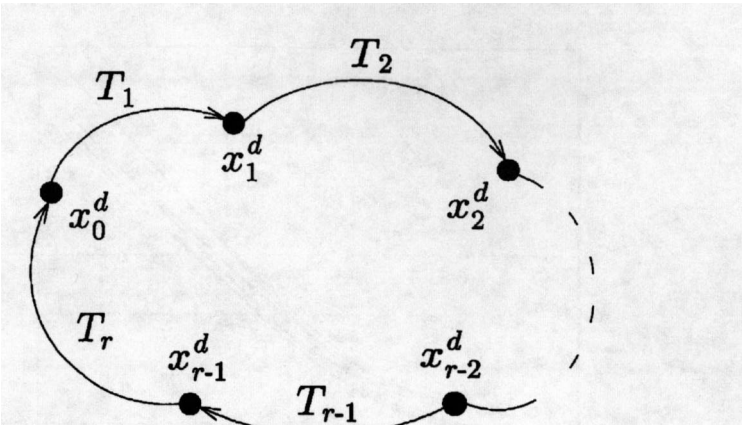

Fig. 22. Mode of operation of a cyclic controller.

We have the following result.

PROPOSITION 4. Let each control coefficient vector $c_l = (c_{1,l}, \ldots, c_{N,l})$ be obtained as the sum of a learned term and a feedforward term:

$$c_l^{[k+1]} = \hat{c}_l^{[k+1]} - \tilde{W}_l^{-1} \tilde{V}_l x_{l-1}^{[k+1]}, \qquad l = 1, \ldots, r, \tag{39}$$

where the learned term $\hat{c}_l$ is updated as

$$\hat{c}_l^{[k+1]} = \hat{c}_l^{[k]} + \tilde{F}_l \varepsilon_l^{[k]}, \tag{40}$$

and $\tilde{F}_l$ is an $N \times N$ constant matrix such that the eigenvalues of $(I - \tilde{W}_l \tilde{F}_l)$ lie in the open unitary disk of the complex plane. Then, the final positioning errors $\varepsilon_l^{[k]}$ converge to zero as $k \to \infty$, uniformly and exponentially.

Proof. By virtue of eqs. (38) and (39), the state of the closed-loop system at the time instant t_l is computed as

$$x_l^{[k+1]} = \tilde{W}_l \hat{c}_l^{[k+1]}, \qquad l = 1, \ldots, r. \tag{41}$$

Note that the role of the feedforward term is to compensate the effect of the last position, $x_{l-1}^{[k]}$. Being that

$$\varepsilon_l^{[k+1]} = x_l^d - x_l^{[k+1]},$$

eq. (41) yields

$$\varepsilon_l^{[k+1]} = x_l^d - \tilde{W}_l \hat{c}_l^{[k+1]} = x_l^d - \tilde{W}_l \hat{c}_l^{[k]} - \tilde{W}_l \tilde{F}_l \varepsilon_l^{[k]},$$

where eq. (40) has been used. Hence, the positioning-error dynamics is obtained as

$$\varepsilon_l^{[k+1]} = (I - \tilde{W}_l \tilde{F}_l) \varepsilon_l^{[k]}, \quad l = 1, \ldots, r,$$

which shows that $\varepsilon_l^{[k]} \to 0$ as $k \to \infty$, uniformly and exponentially. The existence of a matrix $\tilde{F}_l$ satisfying the hypothesis for $l = 1, \ldots, r$, is guaranteed by the controllability assumption on system (18). □

The robustness of the above cyclic controller with respect to small model perturbations can be straightforwardly established along the same lines of Proposition 2 and its proof.

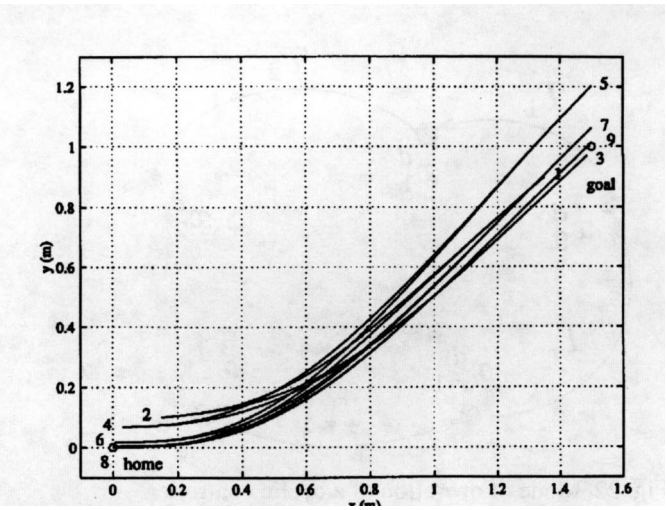

Fig. 23. Fourth simulation: path traced by the vehicle after five iterations of the cyclic controller.

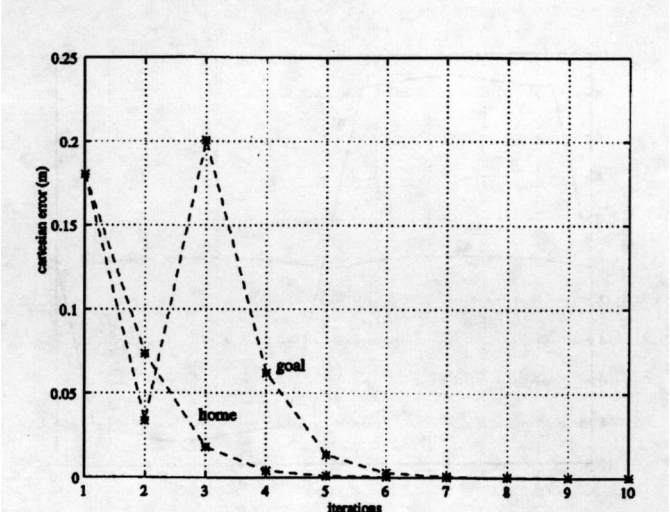

Fig. 24. Fourth simulation: evolution of the Cartesian error at the home and goal configurations over the iterations.

6.2. Simulation Results

To show how the proposed cyclic controller can be used to avoid the necessity of exact reinitialization, we present a fourth simulation for the same car-like robot of Section 4. In particular, the robot model is subject to the same perturbations on the vehicle length and the wheel radius. The cycle is specified by only three configurations: $q_0^d = q^0 = (0, 0, 0, 0)$ (*home*), $q_1^d = q^d = (1.5, 1, \pi/4, 0)$ (*goal*), and $q_2^d = q^0$ again, to be attained at $\tau_0 = 0$ sec, $\tau_1 = T_1 = 5$ sec, and $\tau_2 = T = 10$ sec, respectively. Inside each of the two time intervals $[0, 5)$ and $[5, 10]$, the steering control law has the same structure of the first simulation in Section 4.

The obtained results are shown in Figures 23–25. In particular, Figure 23 shows the resulting Cartesian path from q^0 to q^d. The numbers in the plot refer to successive positions reached by the robot. For the sake of clarity, only five iterations are depicted (each iteration is composed of a forward/backward motion). Note that, although the motion is started exactly at the home configuration q^0, at the end of the first iteration, the robot is displaced to a different configuration (the point marked "2" in the plot). However, as the learning algorithm proceeds, the goal and the home configurations are attained with increasing precision (e.g., the points marked "8" and "9" during the fourth and fifth iterations, respectively). This convergence is confirmed by the plots in Figures 24 and 25. The first shows the evolution over the iterations of the Cartesian positioning error at the home and the goal configurations. The second is the same plot, for the orientation error.

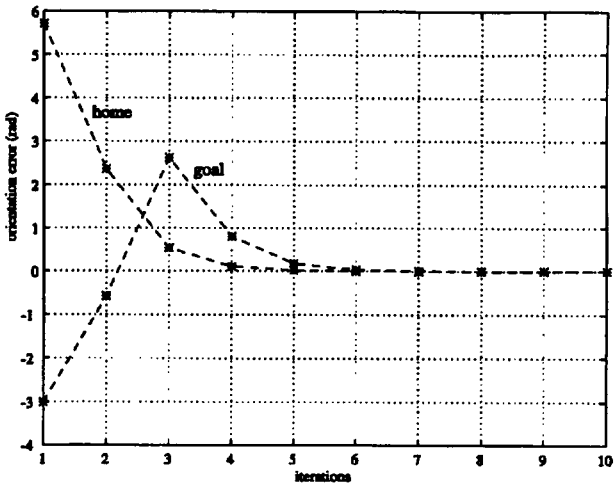

Fig. 25. Fourth simulation: evolution of the orientation error at the home and goal configurations over the iterations.

7. Conclusion

We have presented an iterative learning control method for robust steering of nonholonomic mobile robots. The controller design takes advantage of the possibility of transforming such systems in chained form via feedback. The application of the proposed method requires the execution of repeated trials to achieve a final desired state in finite time: at the end of each trial, an updated control law is computed on the basis of its previous expression and the final state error. While the necessity to perform multiple experiments represents an additional cost, this approach allows one to obtain smooth, natural motions even in the presence of model and input perturbations. The proposed method has also been modified so as to devise a cyclic learning controller, avoiding the necessity of exact system reinitialization between successive trials.

As a case study, we have considered the car-like robot. In particular, we have explicitly derived the effect of typical model uncertainties on the chained-form representation

to prove that the proposed learning algorithm can reject such perturbations. Both simulation and experimental results on a laboratory prototype have been presented to support this theoretical prediction. We mention that comparable, satisfactory results have been obtained over a wide range of steering tasks as well as model perturbations. The application to more complicated wheeled mobile robots, such as a car-like robot towing an arbitrary number of trailers, can be similarly worked out.

One direction of further development that we are currently pursuing is the optimization of performance criteria during the robust execution of a steering task. This can be realized by augmenting the dimension of the coefficient vector $c = (c_1, c_2)$ with respect to the number of the state variables n, and by properly exploiting the extra degrees of freedom to improve the value of the criterion. Typical performance criteria of interest are the length of the trajectory, the maximum curvature along the path, or the distance from obstacles in the workspace. Preliminary encouraging results have been obtained (Ferretti et al. 1996). Interestingly, a similar approach may allow the application of our learning method to the trajectory-tracking case, by defining the integral trajectory error as a cost criterion.

We conclude by mentioning that the presented approach may be successfully applied to other nonholonomic robotic systems. In particular, the application to space robots and multifingered hands—which may be transformed in chained form via feedback—may be of interest in view of the robustness properties of the controller and the naturalness of the learning paradigm for these systems.

Appendix: A Technical Lemma

LEMMA 1. The perturbation $\tilde{\eta}$ on the perturbed error dynamics (eq. (30)) satisfies the following estimate:

$$\|\tilde{\eta}^{[k]}\| \leq g\|\varepsilon^{[k]}\|, \quad g < 1,$$

for sufficiently small γ in the perturbed system (eq. (29)).

Proof. We shall prove the lemma for the perturbed time-invariant system:

$$\dot{x} = Ax(t) + Bu(t) + \gamma\,\eta(x(t), u(t), t), \quad t \in [0, T], \quad (42)$$

the extension to the time-varying system (eq. (29)) being straightforward in view of the piecewise-constant form of matrix $A(t)$.

Denote by ξ the state of the unperturbed system (i.e., the solution obtained for $\gamma = 0$ in eq. (42)). Recall that by eq. (31) we have

$$\tilde{\eta}^{[k]} = d^{[k]} - d^{[k+1]};$$

$d^{[i]}$ being the difference between the perturbed final state $x^{[i]}$ and the unperturbed final state $\xi^{[i]}$, for $i = k, k+1$. According to the formula of variation of constants (Hale 1980, Theorem 1.1, p. 81), we may write

$$\begin{aligned} x^{[i]} &= \xi^{[i]} + \gamma \int_0^T e^{A(T-\tau)}\eta(x^{[i]}(\tau), u^{[i]}(\tau), \tau)d\tau \\ &= \xi^{[i]} + \gamma\tilde{d}^{[i]}, \end{aligned} \quad (43)$$

having set

$$\tilde{d}^{[i]} = \int_0^T e^{A(T-\tau)}\eta(x^{[i]}(\tau), u^{[i]}(\tau), \tau)d\tau.$$

Note that $\tilde{d}^{[i]}$ is part of the solution at time $t = T$ of the differential equation (eq. (42)), in which $u = u^{[i]} = u(c^{[i]})$. The control coefficient vector c then acts as a parameter on which the closed-loop dynamics depend. A general property of differential equations (Hale 1980, Theorem 3.3, p. 21) states that the solution is continuously differentiable (and hence, locally Lipschitzian) with respect to c in its domain of definition.

From eq. (43), we get

$$d^{[i]} = \gamma\tilde{d}^{[i]},$$

so that

$$\tilde{\eta}^{[k]} = \gamma(\tilde{d}^{[k]} - \tilde{d}^{[k+1]}).$$

Denoting by L_c the Lipschitz constant of $\tilde{d}$ with respect to c, we may write

$$\|\tilde{\eta}^{[k]}\| = \gamma\|\tilde{d}^{[k]} - \tilde{d}^{[k+1]}\| \leq \gamma L_c\|c^{[k]} - c^{[k+1]}\|.$$

Using the update law (eq. (27)), with $F = \nu W^{-1}$, one obtains

$$\|\tilde{\eta}^{[k]}\| \leq \gamma L_c\,\nu\,\omega\|\varepsilon^{[k]}\|, \quad (44)$$

where

$$\omega = \|W^{-1}\|.$$

Equation (44) shows that $\|\tilde{\eta}^{[k]}\| \leq g\|\varepsilon^{[k]}\|$, with $g < 1$ for γ sufficiently small. □

Acknowledgments

This work was partially supported by MURST under 40% and 60% funds and by ESPRIT BR project 6546 (PROMotion).

References

Arimoto, S., Kawamura, S., and Miyazaki, F. 1984. Bettering operations of robots by learning. *J. Robot. Sys.* 1:123–140.

Bloch, A. M., McClamroch, N. H., and Reyhanoglu, M. 1990 (Honolulu, HI). Controllability and stabilizability properties of a nonholonomic control system. *Proc. of the 29th IEEE Conf. on Decision and Control.* Los Alamitos, CA: IEEE, pp. 1312–1314.

Bondi, P., Casalino, G., and Gambardella, L. 1988. On the iterative learning-control theory of robotic manipulators. *IEEE J. of Robot. Automat.* 4:14–22.

Brockett, R. W. 1981. Control theory and singular Riemannian geometry. *New Directions in Applied Mathematics*, eds. P. Hinton and G. Young. New York: Springer-Verlag, pp. 11–27.

Brockett, R. W. 1983. Asymptotic stability and feedback stabilization. *Differential Geometric Control Theory*, eds. R. W. Brockett, R. S. Millman, and H. J. Sussmann. Boston, MA: Birkhäuser, pp. 181–191.

Canudas de Wit, C., and Sørdalen, O. J. 1992. Exponential stabilization of mobile robots with nonholonomic constraints. *IEEE Trans. Automat. Control* 37(11):1791–1797.

Ferretti, E., Oriolo, G., Panzieri, S., and Ulivi, G. 1996 (Lisbon, Portugal). Learning nice robust trajectories for a car-like robot. *Proc. of the 4th Int. Symp. on Intell. Robot. Sys.*, pp. 123–130.

Hahn, W. 1967. *Stability of Motion*. Berlin: Springer-Verlag.

Hale, J. K. 1980. *Ordinary Differential Equations*. Malabar, FL: Krieger.

Isidori, A. 1995. *Nonlinear Control Systems*. London: Springer-Verlag.

Lafferriere, G., and Sussmann, H. J. 1991 (Sacramento, CA). Motion planning for controllable systems without drift. *Proc. of the 1991 IEEE Int. Conf. on Robot. and Automat.* Los Alamitos, CA: IEEE, pp. 1148–1153.

Latombe, J.-C. 1991. *Robot Motion Planning*. Norwell, MA: Kluwer.

Lucibello, P. 1994. State steering by learning for a class of nonlinear control systems. *Automatica* 30(9):1463–1468.

Lucibello, P., and Oriolo, G. 1996 (Kobe, Japan). Stabilization via iterative state steering with application to chained-form systems. *Proc. of the 35th IEEE Conf. on Decision and Control.* Washington, DC: IEEE, pp. 2614–2619.

Lucibello, P., and Panzieri, S. 1994 (Lake Buena Vista, FL). Cyclic control of linear systems: Theory and experimental implementation on a flexible arm. *Proc. of the 33th IEEE Conf. on Decision and Control.* Los Alamitos, CA: IEEE, pp. 369–372.

Monaco, S., and Normand-Cyrot, D. 1992 (Tucson, AZ). An introduction to motion planning under multirate digital control. *Proc. of the 31st IEEE Conf. on Decision and Control.* Los Alamitos, CA: IEEE, pp. 1780–1785.

Moore, K. L. 1993. *Iterative Learning Control for Deterministic Systems*. London: Springer-Verlag.

Murray, R. M. 1993. Control of nonholonomic systems using chained forms. *Fields Institute Comm.* 1:219–245.

Murray, R. M., Li, Z., and Sastry, S. S. 1994. *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL: CRC Press.

Murray, R. M., and Sastry, S. S. 1993. Nonholonomic motion planning: Steering using sinusoids. *IEEE Trans. Automat. Control* 38(5):700–716.

Samson, C. 1993. Time-varying feedback stabilization of a car-like wheeled mobile robot. *Int. J. Robot. Res.* 12(1):55–64.

Sontag, E. D. 1990. *Mathematical Control Theory*. New York: Springer-Verlag.

Tilbury, D., Murray, R. M., and Sastry, S. S. 1993. Trajectory generation for the N-trailer problem using Goursat normal form. Memo UCB/ERL M93/12, University of California at Berkeley.

Xu, Y., and Kanade, T. (eds.) 1993. *Space Robotics: Dynamics and Control*. Norwell, MA: Kluwer.