INT. J. CONTROL, 2000, VOL. 73, NO. 10, 980–991

Learning optimal trajectories for non-holonomic systems

GIUSEPPE ORIOLO*†, STEFANO PANZIERI‡ and GIOVANNI ULIVI‡

Many advanced robotic systems are subject to non-holonomic constraints, e.g. wheeled mobile robots, space manipulators and multifingered robot hands. Steering these mechanisms between configurations in the presence of perturbations is a difficult problem. In fact, the *divide et impera* strategy (first plan a trajectory, then track it by feedback) has a fundamental drawback in this case: due to the peculiar control properties of non-holonomic systems, smooth feedback cannot provide tracking of the whole trajectory. As a result, it would be necessary to give up either accuracy in the final positioning or predictability of the actual motion. We pursue here a different approach which does not rely on a separation between planning and control. Based on the learning control paradigm, a robust steering scheme is devised for systems which can be put in chained form, a canonical structure for non-holonomic systems. By overparametrizing the control law, other performance goals can be met, typically expressed as cost functions to be minimized along the trajectory. As a case study, we consider the generation of robust optimal trajectories for a car-like mobile robot, with criteria such as total length, maximum steering angle, distance from workspace obstacles, or error with respect to an off-line planned trajectory.

1. Introduction

Iterative learning control (ILC) is a methodology for enhancing the performance of a control system through training: at each new experiment, the control law is updated on the basis of the results of the previous trial. In repetitive tasks, this approach allows the system to increase its robustness with respect to model perturbations without requiring large processing power, as the control is updated off-line.

Most ILC methods are devised to improve tracking of a trajectory to be executed over and over. This objective was pursued for different kinds of system using various design techniques; for an exhaustive overview, see Moore (1998). In robotics, the use of ILC for trajectory tracking was pioneered by Arimoto *et al.* (1984); other contributions are due to Bondi *et al.* (1988) and De Luca *et al.* (1992). In this context, the definition of the reference trajectory is left to a higher-level module—a trajectory planner—which should provide a feasible time history of the system outputs in the presence of constraints (actuator bounds, workspace obstacles).

Besides trajectory tracking, another fundamental task in robotic systems is steering between configurations. A solution approach to this problem cannot ignore the planning issue; in fact, the trajectory is not given in advance and must be generated on-line—a common situation in sensor-based applications. Interestingly, these applications often involve non-holonomic robots, i.e. robots subject to non-integrable velocity constraints. The most cited example in this class is that of wheeled mobile robots, but one should also mention space manipulators and multifingered robot hands. These systems bring up many challenging problems: for example, planning is complicated by the presence of the velocity constraints. Moreover, stabilization at a given configuration cannot be obtained via smooth time-invariant feedback; as a consequence, this kind of feedback can be used for trajectory tracking only if the trajectory never stops. We refer the reader to the works of Bloch *et al.* (1992) and De Luca *et al.* (1998) for a detailed discussion of these points.

In view of the above comments, it should be clear that steering a non-holonomic robot between given configurations in the presence of perturbations is quite difficult. In principle, one could adopt the usual *divide et impera* strategy: choose one of the many planning techniques for generating smooth point-to-point trajectories (Murray *et al.* 1994: Chapter 8), and then add a feedback action to obtain robust performance. However, as no smooth feedback can track the trajectory in proximity to the final configuration, it would be necessary to switch to non-smooth and/or time-varying feedback in the 'landing' phase, typically resulting in erratic, awkward motions as shown in De Luca *et al.* (1998).

ILC provides an alternative, effective solution to the above problem without forcing a separation between planning and control. This was shown in Oriolo *et al.* (1998) where—motivated by a general framework due to Lucibello (1994)—we proposed a learning controller for steering non-holonomic robots. The idea is to define the control input as the linear combination of a suitable set of generating functions, updating the coefficient vector on the basis of the positioning error obtained at the end

Received February 1999. Revised November 1999.

*Author for correspondence. e-mail: oriolo@labrob.ing.uniroma1.it

† Dipartimento di Informatica e Sistemistica, Università di Roma 'La Sapienza', Via Eudossiana 18, 00184 Roma, Italy.

‡ Dipartimento di Informatica e Automazione, Università di Roma Tre, Via della Vasca Navale 79, 00146 Roma, Italy.

of each experiment. Robustness with respect to model perturbations was theoretically proved and illustrated by experiments on a car-like robot. A drawback of this method is that, once a set of generating functions has been chosen, there is no direct command on the 'quality' of the generated trajectory.

In this paper, the above limitation is overcome by exploiting further possibilities offered by ILC. In particular, we show that our learning paradigm can be extended so as to generate trajectories that, in addition to being executable in perturbed conditions, meet other performance goals, typically expressed as cost criteria to be minimized. Essentially, this is obtained through the use of an overparametrized control: by augmenting the dimension of the coefficient vector with respect to the number of the state variables, we provide space for the optimization process. The main advantages of the resulting method are (i) the fact that it is not necessary to provide a trajectory connecting the initial with the final configuration, (ii) the robustness of the obtained trajectories against disturbances and model perturbations, and (iii) the flexibility in the choice of the trajectory cost criterion.

As a case study, we consider a car-like robot and attempt the synthesis of robust trajectories that minimize cost functions such as the total length, the maximum steering angle, the distance from workspace obstacles, or the error with respect to an off-line planned trajectory. Simulation results for a realistic vehicle model are reported in order to show the effectiveness of the proposed technique.

The paper is organized as follows. In § 2, we recall the properties of chained systems as canonical forms for non-holonomic systems. The proposed method is illustrated in § 3, first by introducing an optimal learning scheme for a special class of linear time-varying systems and then by working out its adaptation to chained forms; robustness and optimization properties are explicitly discussed. In § 4, we apply our method to a car-like non-holonomic robot. Work in progress and possible extensions are discussed in the concluding section.

2. Chained-form systems

Consider the two-input driftless non-linear system

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

called the $(2,n)$ *chained form* (Murray and Sastry 1993).

Controllability of the system is readily established by using the Lie algebra rank condition (Isidori 1995). Note that if a constant v_1 is chosen, the chained form becomes a linear time-invariant system and behaves like a chain of integrators from z_2 to z_n, driven by the input v_2. More generally, if v_1 is a given function of time, equations (1) represent a linear time-varying system.

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

$$\dot{q} = h_1(q)u_1 + h_2(q)u_2, \qquad q \in \mathbb{R}^n \tag{2}$$

into a $(2,n)$ chained form through a change of coordinates $z = \alpha(q)$ and an input transformation $u = \beta(q)v$. In particular, kinematic models of non-holonomic systems with two inputs and $n \leq 4$ state variables can be always put in chained form.

Suppose that we wish to steer the state of system (2) from an initial configuration $q(0) = q^0$ to a desired configuration $q(T) = q^{\mathrm{d}}$ 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. Correspondingly, the chained system (1) must move from $z^0 = \alpha(q^0)$ to $z^{\mathrm{d}} = \alpha(q^{\mathrm{d}})$.

It is relatively easy to plan trajectories for chained forms. Among the available techniques, we mention sinusoidal inputs (Murray and Sastry 1993), piecewise-constant inputs (Monaco and Normand-Cyrot 1992), and mixed piecewise-constant and polynomial inputs (Tilbury *et al.* 1995). Below, we summarize the latter method, that will later be adopted as a component of our learning controller. Let

$$v_1(t) = c_1(t) \tag{3}$$

$$v_2(t) = c_{21} + c_{22}t + \cdots + c_{2,n-1}t^{n-2} \tag{4}$$

where $c_1(t)$ is a piecewise-constant function and $c_{21}, \ldots, c_{2,n-1}$ are constants. To steer system (1) from z^0 to z^{d} in T seconds, $c_1(t)$ should satisfy

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

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

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

where $M(c_1, T)$ is a non-singular matrix, provided that $c_1(t) \neq 0$, for some $t \in [0, T]$.

The control law (3–4) produces 'natural' robot trajectories; nevertheless, being inherently open-loop, it does not provide any degree of robustness. As a consequence, it will not yield exact steering when applied to a system whose model is perturbed with respect to the nominal chained form (1).

The basic idea of our method is to achieve robustness with respect to repetitive disturbances and model perturbations by computing the control parameters (in this case, $c_1(t)$ and $c_{21},\ldots,c_{2n-1}$) through a learning scheme, improving at the same time the value of a given performance criterion. In the next section we show how this can be done with reference to a control law that generalizes equations (3–4).

3. Learning optimal trajectories

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

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

$$\dot{z}_a(t)=v_1(t) \tag{5}$$

$$\dot{z}_b(t)=A(t)z_b(t)+B\,v_2(t) \tag{6}$$

where

$$A(t)=\begin{bmatrix}0&0&0&\ldots&0\\ v_1(t)&0&0&\ldots&0\\ 0&v_1(t)&0&\ldots&0\\ \vdots&\vdots&\ddots&\ddots&\vdots\\ 0&\ldots&0&v_1(t)&0\end{bmatrix},\quad B=\begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix} \tag{7}$$

During each iteration of the learning algorithm, we make use of a control law which is a generalization of eqs. (3–4). 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 inputs during the kth iteration ($k=1,2,\ldots$) be expressed in the parametric form

$$v_1^{[k]}(t)=c_{1i}^{[k]},\qquad t\in[t_{i-1},t_i),\ i=1,\ldots,p \tag{8}$$

$$v_2^{[k]}(t)=\sum_{j=1}^{q}c_{2j}^{[k]}\lambda_j(t),\quad t\in[0,T],\ j=1,\ldots,q \tag{9}$$

where $c_{1i}^{[k]},c_{2j}^{[k]}\in\mathbb{R}$, while the $\lambda_j(t):[0,T]\mapsto\mathbb{R}$ are assigned piecewise-polynomial functions with discontinuities occurring only at the time instants t_i (see figure 1). For example, one may set

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

where $\pi_i(t)$ is 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 in 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.

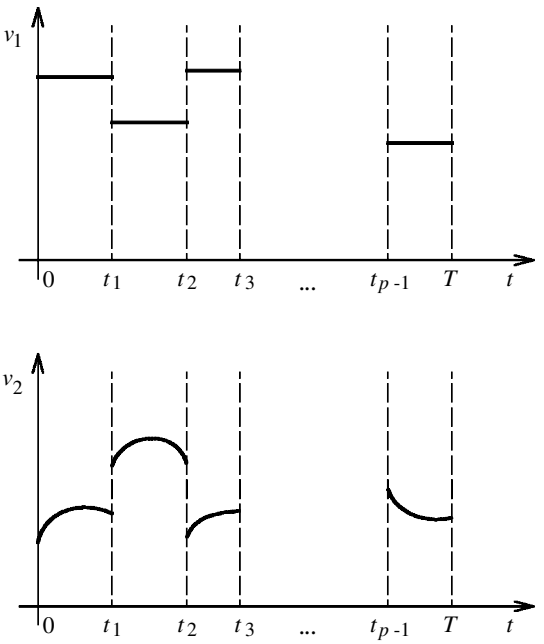

Figure 1. Profile of inputs v_i within each iteration of the learning algorithm.

As the first subsystem (5) is a simple integrator, a one-dimensional ($p=1$) control coefficient c_1 would be sufficient to steer z_1 to z_1^{d}. However, in view of the structure of matrix A in equation (7), to allow steering of the other variables $z_2,\ldots,z_n$ one must set $p\geq 2$ in general, whereas $p=1$ is admissible only if $z_1^0\neq z_1^{\mathrm{d}}$.

The dimension q of the control coefficient c_2 must be at least $n-1$. This is easily established by noting that the second subsystem (5) has dimension $n-1$ and is controllable with the choice (8) for v_1, provided that $c_{1i}^{[k]}\neq 0$ for some i; therefore, there exists a piecewise-polynomial control that transfers the system state z_2 from z_2^0 to z_2^{d} in time T (Sontag 1990: Proposition 3.7.1). In particular, a unique solution exists if $c_2\in\mathbb{R}^{n-1}$, whereas infinite solutions can be found if the dimension of c_2 is larger.

In the following, it is assumed that $r=p+q>n$, i.e. the number of control coefficients is strictly larger than

the number of state variables. By choosing an overparametrized control law, we provide additional degrees of freedom for improving a performance criterion during the learning process.

We next present a learning control algorithm for a class of linear time-varying systems that encompasses subsystem (6) 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 (5–6) or, equivalently, for system (1).

3.1. *Optimal learning control for linear systems*

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

$$\dot{x}(t) = A(t)x(t) + Bu(t), \qquad x \in \mathbb{R}^N, u \in \mathbb{R}^M, t \in [0,T] \tag{10}$$

such that

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

with $t_0 = 0$, $t_p = T$, and $\{A_1, \ldots, A_p\}$ a sequence of constant matrices. With the assumption that the pair (A_i, B) is controllable for all i, our objective is to learn a control that steers system (10) from $x(0) = x^0$ to a desired x^{d} in finite time T.

Assume that u is chosen as

$$u(t) = \sum_{j=1}^{q} c_j \lambda_j(t) \tag{11}$$

where $c_j \in \mathbb{R}$ and the $\lambda_j(t) : [0,T] \mapsto \mathbb{R}^M$ are linearly independent, piecewise-continuous vector functions with discontinuities occurring only at the time instants t_i. We assume $q > N$, i.e. the dimension of the coefficient vector $c = (c_1, \ldots, c_q)$ is strictly larger than the dimension of the state of system (10).

Once a suitable set of base functions has been chosen (e.g. piecewise-constant, piecewise-polynomial and piecewise-exponential functions), the controllability assumption guarantees that in the chosen class there exists ∞^{q-N} control inputs (each corresponding to a value of vector c) achieving the desired x^{d} in a finite time T.

Define $\delta_i = t_i - t_{i-1}$, for $i = 1, \ldots, p$. We have

$$\begin{aligned} x(t_i) &= \mathrm{e}^{A_i \delta_i} x(t_{i-1}) + \sum_{j=1}^{q} c_j \int_{t_{i-1}}^{t_i} \mathrm{e}^{A_i (t_i - \tau)} B \lambda_j(\tau)\, \mathrm{d}\tau \\ &= V_i\, x(t_{i-1}) + W_i\, c \end{aligned} \tag{12}$$

with V_i and W_i constant matrices of dimensions $N \times N$ and $N \times q$, respectively, given by

$$V_i = \mathrm{e}^{A_i \delta_i}$$

$$W_i = \int_0^{\delta_i} \mathrm{e}^{A_i (\delta_i - \tau)} B [\lambda_1(t_{i-1} + \tau) \ \cdots \ \lambda_q(t_{i-1} + \tau)]\, \mathrm{d}\tau$$

for $i = 1, \ldots, p$. Applying recursively equation (12), we obtain

$$x(T) = V x^0 + W c \tag{13}$$

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

$$V = V_p V_{p-1} \cdots V_1 \tag{14}$$

$$W = W_p + V_p W_{p-1} + \cdots + V_p \cdots V_2 W_1 \tag{15}$$

Note that $V x^0 = 0$ in equation (13) if x^0 is an equilibrium point.

Assume that, at each iteration, the control system (10) is exactly re-initialized at x^0. The control input (11) during the kth iteration ($k = 1, 2, \ldots$) is denoted by

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

while the final system positioning error is

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

Proposition 1: *Let the coefficient vector be updated according to*

$$c^{[k+1]} = c^{[k]} + F \varepsilon^{[k]} + n^{[k]} \tag{16}$$

where F is a $q \times N$ constant matrix such that the eigenvalues of matrix $I - WF$ lie in the open unitary disk of the complex plane, and $n^{[k]}$ is an arbitrary *q-vector in the null space of matrix W (i.e. such that $W n^{[k]} = 0$). 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

$$\varepsilon^{[k+1]} = (I - WF) \varepsilon^{[k]} \tag{17}$$

Hence, under the hypothesis of the proposition, we have that $\varepsilon^{[k]} \to 0$ as $k \to \infty$, uniformly and exponentially. The existence of (infinite) matrices F satisfying the hypothesis is guaranteed by the controllability assumption on system (10), which in turn implies the controllability of the pair (I, W). □

If all the eigenvalues of $I - WF$ are placed at zero, the learning algorithm produces a feasible solution (i.e. a coefficient vector c such that $\varepsilon = 0$) after one iteration. This can be done by letting $F = W^r$, where W^r is any *right inverse* of W (Rao and Mitra 1971). Since matrix W has full row rank, it admits infinite right inverses. A classical choice is the unique *pseudoinverse* of W, i.e.

$W^\dagger = W^{\mathrm{T}}(WW^{\mathrm{T}})^{-1}$. Interestingly, the solution obtained from (16) by choosing $F = W^\dagger$ and $n^{[k]} = 0$, namely

$$c^{[k+1]} = c^{[k]} + W^\dagger \varepsilon^{[k]}$$

provides the minimum-norm *update* $c^{[k+1]} - c^{[k]}$ of the coefficient vector that solves the steering problem. In particular, if the coefficient vector is initialized at $c^{[1]} = 0$, one gets $\varepsilon^{[1]} = x^{\mathrm{d}} - Vx^0$ and the choice

$$c^{[2]} = W^\dagger \varepsilon^{[1]}$$

gives the minimum-norm *coefficient* that solves the steering problem.

Placing the eigenvalues of $I - WF$ at zero in the learning algorithm yields a feasible solution after one iteration only in principle. In practice, the presence of perturbations on the nominal system (10) requires the execution of multiple iterations. We shall later analyse the behaviour of the proposed method in the presence of perturbations (see § 3.1.2).

3.1.1. *Choice of the null-space vector* $n^{[k]}$. Proposition 1 states that a learning algorithm based on the update (16) will produce a feasible solution regardless of the particular choice of the null-space vector $n^{[k]}$. Therefore, one can exploit this degree of freedom to pursue an additional objective.

Assume that we wish to learn a control input inside the chosen class (11) that minimizes a cost criterion $H(c)$ along the steering interval. To this end, we can choose the null-space vector as

$$n^{[k]} = -\alpha^{[k]}(I - W^\dagger W)\nabla_c H|_{c^{[k]}}, \quad \alpha^{[k]} > 0 \qquad (18)$$

where $(I - W^\dagger W)$ is the orthogonal projection matrix in the null space of W and $\nabla_c H$ is the gradient of H w.r.t. vector c. From equation (16), the resulting update for the coefficient vector is

$$c^{[k+1]} = c^{[k]} + W^\dagger \varepsilon^{[k]} - \alpha^{[k]}(I - W^\dagger W)\nabla_c H|_{c^{[k]}} \qquad (19)$$

As already discussed, use of equation (19) for the nominal system yields a feasible solution $c^{[2]}$ after one iteration. In successive iterations ($k = 2, \ldots$), it will be $\varepsilon^{[k]} = 0$ and $c^{[k]}$ will evolve as with the *gradient projection* method (Luenberger 1984). In fact, when $\alpha^{[k]} = 1$, the right-hand side of equation (18) provides the projected antigradient of H, the projection accounting for the necessity of performing a constrained optimization: in particular, at each iteration the value of H should decrease while satisfying $x(T) = x^{\mathrm{d}}$, or, according to (13), the linear constraint

$$Wc = x^{\mathrm{d}} - Vx^0$$

Note that, in view of the linearity of the constraint, the projection matrix $(I - W^\dagger W)$ is constant.

The following remarks are in order:

- While it is easily proven that the negative antigradient of H computed at $c^{[k]}$ provides a local descent direction for any cost criterion, a procedure for selecting the stepsize $\alpha^{[k]}$ is needed to guarantee that a non-linear H actually decreases at the new point $c^{[k+1]}$ given by equation (19). To this end, one can use a *line search* algorithm to identify the value of α that minimizes H over the feasible segment emanating from $c^{[k]}$. A common implementation in constrained optimization methods is a simple Armijo-like test (Luenberger 1984).
- The gradient projection method is guaranteed to converge to a point which satisfies the Kuhn–Tucker necessary conditions for constrained optimality. In practice this means that at least a local minimum is reached, where the projected gradient is zero. The characteristics of the cost function H as well as the initialization of the control coefficient c play a role in determining the actual outcome of the optimization process. For example, if H is a quadratic function of c, the gradient projection algorithm is guaranteed to converge to the true constrained minimum.
- We assumed above that a closed form is available for H as a function of the coefficient vector c. While this may be the case (see the case study of the next section), it may happen that the closed form is complicated or even unavailable. In this case, as is customary in non-linear optimization techniques, the gradient may be computed numerically by evaluating H in the neighbourhood of the current point $c^{[k]}$. These function evaluations may be performed through simulation of system (10).

3.1.2. *Robustness analysis.* As the error dynamics (17) is uniformly and exponentially stable, small non-persistent perturbations are completely rejected, whereas small persistent perturbations give limited errors (Hahn 1967). This fact is essential in establishing the following result.

Proposition 2: *Let system* (10) *be perturbed*† *as*

$$\dot{x}(t) = A(t)x(t) + Bu(t) + \gamma\,\eta(x(t), u(t), t) \qquad (20)$$

where η *is a continuous function of* x *and* u. *Then, for sufficiently small* γ, *the learning algorithm* (16) *designed on the basis of the nominal system* (10) *gives a final positioning error* ε *that converges to zero over the iterations, uniformly and exponentially, provided that*

† Although we restrict our attention to additive perturbations, it should be realised that that effect of multiplicative perturbations can be (at least approximately) expressed in the form (20) by using a Taylor expansion.

$$n^{[k]} = o\left(||\varepsilon^{[k]}||\right) \tag{21}$$

The proof is a straightforward extension of the proof of Proposition 2 in Oriolo *et al.* (1998), and is therefore omitted.

If, in particular, equation (18) is used, one may take $\alpha^{[k]} = o\left(||\varepsilon^{[k]}||\right)$ in order to achieve condition (21).

The above result characterizes the performance of the proposed method from a general viewpoint, and in fact no special assumption is made on the structure of the perturbation function. More specific results may be obtained by using the actual expression of η, and in some cases it may be possible to compute an upper bound for parameter γ in (20). In any case, Proposition 2 provides a firm basis for a robustness claim which is further supported by simulation and experimental results. In addition, if γ is so large that the resulting perturbation on the error dynamics is bounded but persistent, one can still prove ultimate boundedness of the error by means of classical stability arguments (Hahn 1967: Th. 56.4).

3.2. *Application to chained-form systems*

It is now possible to devise an iterative learning controller for the chained-form system (1) or, equivalently, for system (5–6) by interleaving learning phases on the two subsystems. In fact, having chosen v_1 in equation (8) as a piecewise-constant function, the second subsystem (6) has exactly the linear time-varying structure (10) and is controllable. The same is trivially true for the first subsystem (5), which is a simple integrator.

The learning algorithm proceeds as follows.

Step 0. Set $k = 1$. Use arbitrary values $c_{11}^{[1]}, \ldots, c_{1,p}^{[1]}$ and $c_{21}^{[1]}, \ldots, c_{2,q}^{[1]}$ to initialize the control coefficients $c_{11}, \ldots, c_{1,p}$ and $c_{21}, \ldots, c_{2,n-1}$, respectively. To speed up convergence, it is convenient to initialize the coefficients at the values computed for the nominal system through a procedure similar to the one outlined at the end of § 2, i.e. to start with an open-loop control.

Step 1. Perform the kth experiment using the control inputs given by equations (8) and (9), and measure the final positioning error $\varepsilon^{[k]} = \left(\varepsilon_a^{[k]}, \varepsilon_b^{[k]}\right) = z^{\mathrm{d}} - z^{[k]}(T)$.

Step 2a. Apply equation (19) to the first subsystem (5) in order to obtain the update rule for the first control coefficient vector $c_1 = (c_{11}, \ldots, c_{1,p})$. This gives

$$c_1^{[k+1]} = c_1^{[k]} + \delta^{\mathrm{T}^\dagger} \varepsilon_a^{[k]} - \alpha_1^{[k]} (I - \delta^{\mathrm{T}^\dagger} \delta^{\mathrm{T}}) \nabla_{c_1} H|_{c^{[k]}} \tag{22}$$

with $\delta^{\mathrm{T}} = (\delta_1 \ \ldots \ \delta_p)$. Through equation (8), this determines the control law $v_1^{[k+1]}(t)$ to be used in the next iteration.

Step 2b. Plug the value of $v_1^{[k+1]}(t)$ in equations (7), (14) and (15) to build the current matrices $A\left(v_1^{[k+1]}\right)$, $V\left(v_1^{[k+1]}\right)$ and $W\left(v_1^{[k+1]}\right)$ for the second subsystem (6). The second control coefficient c_2 is then updated as

$$c_2^{[k+1]} = c_2^{[k]} + W^{[k+1]^\dagger} \left[\varepsilon_b^{[k]} - \Delta V^{[k+1]} z_b^0 - \Delta W^{[k+1]} c_2^{[k]} \right] + \alpha_2^{[k]} \left(I - W^{[k+1]^\dagger} W^{[k+1]} \right) \nabla_{c_2} H|_{c^{[k]}} \tag{23}$$

having set $V^{[k+1]} = V\left(v_1^{[k+1]}\right)$, $W^{[k+1]} = W\left(v_1^{[k+1]}\right)$, $\Delta V^{[k+1]} = V^{[k+1]} - V^{[k]}$ and $\Delta W^{[k+1]} = W^{[k+1]} - W^{[k]}$. Through equation (9), this determines the control law $v_2^{[k+1]}(t)$ to be used in the next iteration.‡

Step 3. Set $k = k + 1$ and go to Step 1.

The correctness of the algorithm is proven next. For compactness, we drop the time dependence for quantities measured at the end of the time interval $[0, T]$; for example, $z^{[k]} = z^{[k]}(T)$.

Proposition 3: *Using the proposed learning controller, the final positioning error* $\varepsilon = z^{\mathrm{d}} - z(T)$ *of the nominal system* (5–6) *converges to zero in one iteration.*

Proof: Since the evolution of the first variable $z_a = z_1$ is not influenced by the second subsystem, it is easy to realize that $v_1^{[k]}(t)$ converges to a feasible value which steers z_1 to its desired value z_1^{d}. As for the second subsystem, we have at the end of the $(k+1)$th iteration

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

so that

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

Using the update law (23) for the control coefficient c_2 as well as the expressions of $\Delta V^{[k+1]}$ and $\Delta W^{[k+1]}$, one obtains easily $\varepsilon_b^{[k+1]} = 0$, which shows that the error dynamics for the second subsystem is also completely decoupled from that of the first subsystem. □

Note the following points

- The expressions of matrices $V^{[k+1]}$ and $W^{[k+1]}$ in (23) depend, through equations (14–15), on the choice of the piecewise constant function $v_1(t)$ in (8) and of the piecewise polynomial functions $\lambda_j(t)$ in (9). However, once the structure of the control is chosen, the closed-form derivation of $V^{[k+1]}$ and $W^{[k+1]}$ is usually quite simple.

‡ Note that two terms have been added in equation (23) with respect to the basis update law (16). Such terms implement a feedforward action that compensates the effect of v_1 changing over the iterations.

- Since both matrices δ^{T} and $W^{[k+1]}$ are of full row rank by construction, their pseudoinverses are computed in closed form as (Rao and Mitra 1971)

$$\delta^{\mathrm{T}\dagger} = \delta\left(\delta^{\mathrm{T}}\delta\right)^{-1}, \quad W^{[k+1]\dagger} = W^{[k+1]\mathrm{T}}\left(W^{[k+1]}W^{[k+1]\mathrm{T}}\right)^{-1}$$

  Note also that $I - \delta^{\mathrm{T}\dagger}\delta^{\mathrm{T}}$ and $I - W^{[k+1]\dagger}W^{[k+1]}$ in (22–23) are the orthogonal projection matrices in the null space of δ^{T} and $W^{[k+1]}$, respectively. Their presence guarantees that the constraint $z(T) = z^{\mathrm{d}}$ is satisfied for any value of the gradients $\nabla_{c_1} H|_{c^{[k]}}$ and $\nabla_{c_2} H|_{c^{[k]}}$.
- In view of the above two remarks, the computational load of the above algorithm is very limited. In fact, all matrices in the update laws (22–23) are closed-form computable. The only other burden is the selection of stepsizes $\alpha_1^{[k]}$ and $\alpha_2^{[k]}$, which however is extremely fast if the aforementioned Armijo-like test is adopted. In practice, the computation time between two experiments does not exceed a few milliseconds on a Pentium-class computer.

As for the robustness of the above learning controller, it is straightforward to prove that, provided that $\alpha_1^{[k]}$ is $o(\|\varepsilon_a\|)$ and $\alpha_2^{[k]}$ is $o(\|\varepsilon_b\|)$, the proof of Proposition 3 still holds for *perturbed* chained systems of the form

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \\ \vdots \\ \dot{z}_n \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ z_2 v_1 \\ \vdots \\ z_{n-1} v_1 \end{bmatrix} + \gamma \begin{bmatrix} \eta_1(z, v) \\ \eta_2(z, v) \\ \eta_3(z, v) \\ \vdots \\ \eta_n(z, v) \end{bmatrix} \tag{24}$$

in which the perturbation on z_1 depends neither on $z_2, \ldots, z_n$ nor on 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 the presence of sufficiently small perturbations.

However, one should note that, if the stepsizes α_1 and α_2 were weighted with the positioning error norm (as requested in order to guarantee robust steering of the perturbed model), the optimization process would slow down as the perturbed system gets closer to the exact steering condition. Therefore, it is convenient to apply the proposed learning scheme as follows:

(1) A preliminary learning process with unweighted optimization is performed on the nominal model, in order to compute a nominal value for the coefficient vector (*optimal learning phase*).

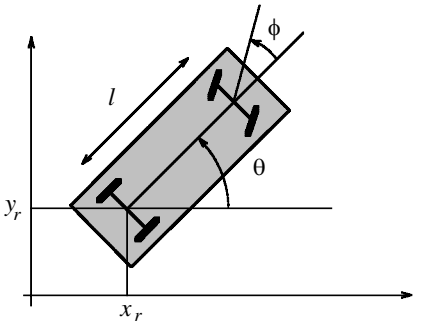

Figure 2. Generalized coordinates for a car-like robot.

(2) A phase of pure learning (i.e. with $\alpha_i^{[k]} = 0$ or $\alpha_i^{[k]} = o(\|\varepsilon_i^{[k]}\|)$, for $i = 1, 2$ and any k) is performed through experiments on the perturbed model so as to make the nominal trajectory robust (*robust learning phase*).

The effectiveness of the above approach is illustrated in the next section, using a car-like robot as a case study.

4. Case study: the car-like robot

Consider a non-holonomic mobile robot with the kinematics of an automobile (figure 2). 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 whereas the rear wheel orientation is fixed. The distance between the two wheel centres is l. The generalized coordinates are $q = (x_{\mathrm{r}}, y_{\mathrm{r}}, \theta, \phi)$, where $(x_{\mathrm{r}}, y_{\mathrm{r}})$ 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

$$\begin{bmatrix} \dot{x}_{\mathrm{r}} \\ \dot{y}_{\mathrm{r}} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{1}{l}\tan\phi \\ 0 \end{bmatrix} \rho u_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_2$$

where ρ is the driving wheel radius, u_1 is the angular velocity of the driving wheel and u_2 is the steering rate. A control singularity occurs when $\phi = \pm\pi/2$; its relevance is, however, limited, due to the restricted range of ϕ in practical cases.

To convert this system into chained form, use the change of coordinates

$$z_1 = x_{\mathrm{r}}$$

$$z_2 = \frac{1}{l}\frac{\tan\phi}{\cos^3\theta}$$

$$z_3 = \tan\theta$$

$$z_4 = y_{\mathrm{r}}$$

together with the input transformation

$$u_1 = \frac{v_1}{\rho \cos\theta} \tag{25}$$

$$u_2 = -\frac{3}{l}\frac{\sin\theta}{\cos^2\theta}\sin^2\phi\ v_1 + l\cos^3\theta\cos^2\phi\ v_2 \tag{26}$$

It is easy to verify that the transformed system is in the form (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 $\theta \in (-k\pi/2, k\pi/2)$ $(k = 1, 2, \ldots)$.

4.1. *Effects of perturbations*

Since the above transformation is model based, model perturbations perturb the chained structure for this robot. We consider two cases, namely perturbations on the vehicle length l and on the wheel radius ρ; similar derivations can be given for other perturbations. The reader is referred to Oriolo *et al.* (1998) for details.

Assume that the vehicle length is not exactly known. This means that, in performing the input transformation (25–26), we make use of a value $l + \Delta l$ in place of the true value l. The following perturbed chained-form system is obtained

$$\begin{bmatrix} \dot z_1 \\ \dot z_2 \\ \dot z_3 \\ \dot z_4 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ z_2 v_1 \\ z_3 v_1 \end{bmatrix} + \frac{\Delta l}{l}\begin{bmatrix} 0 \\ 3z_2^2 z_3 \cos^2(\arctan z_3) v_1 + v_2 \\ 0 \\ 0 \end{bmatrix} \tag{27}$$

Similarly, if the driving wheel radius is not exactly known, in performing the input transformation (25–26) a value $\rho + \Delta\rho$ is used in place of the true value ρ. The perturbed chained-form system is

$$\begin{bmatrix} \dot z_1 \\ \dot z_2 \\ \dot z_3 \\ \dot z_4 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ z_2 v_1 \\ z_3 v_1 \end{bmatrix} + \frac{\Delta\rho}{\rho}\begin{bmatrix} v_1 \\ \dfrac{3z_2 z_3}{l\cos(\arctan z_3)} v_1 \\ z_2 v_1 \\ z_3 v_1 \end{bmatrix} \tag{28}$$

Note that the perturbed models (27) and (28) are of the form (24); therefore, our algorithm should provide robust steering of the car-like robot in both cases.

4.2. *Cost criteria for the car-like robot*

The possibility of optimizing the learning process with respect to a given criterion can be exploited in order to achieve further objectives in addition to state steering. Among these, we mention:

(1) minimize the length of the trajectory;

(2) keep the steering angle ϕ inside a given interval $[-\phi_{\max}, \phi_{\max}]$ to avoid steering saturations;

(3) avoid obstacles that are present in the workspace;

(4) track as closely as possible a given trajectory connecting q^0 to q^{d} (e.g. generated by a collision-free motion planner).

The length of the trajectory is simply expressed as

$$H_1(c) = \int_0^{\mathrm{T}} \rho\, u_1(t)\, \mathrm{d}t$$

To pursue the second objective, one may use the following penalty function

$$H_2(c) = \max_{t\in[0,T]}\left(\frac{\phi(t)}{\phi_{\max}}\right)^{2p}, \qquad p \geq 1$$

which takes into account the maximum value of $|\phi|$ along the trajectory. By increasing p, values of $|\phi|$ above $\phi_{\max}$ are penalized more, whereas maximum values of $|\phi|$ below $\phi_{\max}$ give a smaller contribution.

As for the third objective, assume for simplicity the case of a circular robot of radius R among s circular obstacles $\mathcal{O}_1, \ldots, \mathcal{O}_s$, of radius $r_1, \ldots, r_s$, respectively. A cost criterion that penalizes collision with obstacles along the trajectory $\mathcal{T}$ is

$$H_3(c) = \max_{p\in\mathcal{T}}\left\{\max_{j\in\{1,\ldots,s\}}\left[\delta_{-2}(R + r_j - d_j)\right]\right\}$$

where p is the generic point of the trajectory, δ_{-2} is the unit ramp function and d_j is the distance between the centre of the robot and the centre of $\mathcal{O}_j$. A generalization of this function that applies to a polygonal robot among polygonal obstacles can be easily computed.

Finally, the trajectory tracking objective can be addressed by minimizing the cost function

$$H_4(c) = \int_0^T \left(z^{\mathrm{d}}(t) - z(t)\right)^{\mathrm{T}}\left(z^{\mathrm{d}}(t) - z(t)\right)\mathrm{d}t$$

where $z^{\mathrm{d}}(t)$, $t \in [0, T]$, is the desired trajectory connecting $z^0 = \alpha(q^0)$ to $z^{\mathrm{d}} = \alpha(q^{\mathrm{d}})$.

It is possible to show that, among the above cost functions, only H_1 and H_4 can be explicitly expressed as a function of the control coefficients, once the structure of the steering law has been chosen. Although, for compactness, details are not given here, this implies that both gradients $\nabla_c H_1$ and $\nabla_c H_4$ can be computed in closed form.

On the other hand, no closed form is available for H_2 and H_3 as functions of the control coefficients. This is essentially due to the presence of the 'max' operator in both functions. Therefore, the gradient must be computed numerically by function evaluations in the neighbourhood of the current coefficient vector $c^{[k]}$. These

may be performed either via simulation or, for the perturbed system, by means of experiments. The numerical evaluation of the gradients also overcomes the lack of differentiability of H_2 and H_3.

4.3. *Simulation results*

To test the performance of the proposed method, we have run four simulations, each involving a different cost criterion. In particular, in the first two the car-like robot is assigned a parallel parking task, in the third a given configuration must be reached across a cluttered workspace, while in the fourth a forward parking task must be executed.

For the robot considered, it is $l = 0.2\,\mathrm{m}$ and $\rho = 0.02\,\mathrm{m}$ in the nominal model. To obtain a realistic perturbed model, we have included the following non-idealities in simulation:

- perturbations of 10% on both the values of l and ρ;
- a digital version of the controller has been implemented with a sampling time of 0.025 s;
- a simulated encoder has been used to measure the wheel angular position.

In all the simulations, an optimal learning phase is performed on the nominal model to compute a trajectory that minimizes (at least locally) the cost criterion. Then, a robust learning phase is executed on the perturbed system to make the generated trajectory robust.

In the first simulation, the car-like robot must move from $q^0 = (0, 0.8, 0^\circ, 0^\circ)$ (start) to $q^\mathrm{d} = (0, 0, 0^\circ, 0^\circ)$ (goal) in $T = 10\,\mathrm{s}$, minimizing the trajectory length H_1. The structure of the control law has been chosen as follows

$$v_1(t) = c_{1i}$$

$$v_2(t) = c_{21,i}(t - t_{i-1})^2 + c_{22,i}(t - t_{i-1}) + c_{23,i}$$

$$t \in [t_{i-1}, t_i), \quad i = 1, \ldots, 3$$

where $t_0 = 0$, $t_1 = 3$, $t_2 = 7$, $t_3 = 10$. Correspondingly, it is $c_1 = (c_{11}, c_{12}, c_{13}) \in \mathbb{R}^3$ and $c_2 = (c_{21,1}, \ldots, c_{23,3}) \in \mathbb{R}^9$.

Details of the performance of the algorithm are reported in table 1. In the optimal learning phase, the positioning error is zeroed in one iteration (as expected), while the length of the trajectory is reduced to 1.005 m in the first iteration and is not appreciably reduced in the subsequent iteration. When the nominal control thus computed is applied to the perturbed system, a non-zero final error (0.08 m in norm) is experienced. Then, the robust learning phase takes over, reducing the error to zero in four iterations (see figure 3). The final value of H_1 is 1.10 m, very close to the minimum found during the optimal learning phase. Note that the robot undergoes a large variation of the steering angle, consistently with the objective of minimizing the trajectory length.

Optimal learning phase			Robust learning phase		
Iteration	H	$\|\varepsilon\|$	Iteration	$\|\varepsilon\|$	H
1	1.005	0.00	1	0.08	1.21
2	1.00	0.00	2	0.05	1.14
			3	0.01	1.09
			4	0.00	1.10

Table 1. First simulation: details of the learning process.

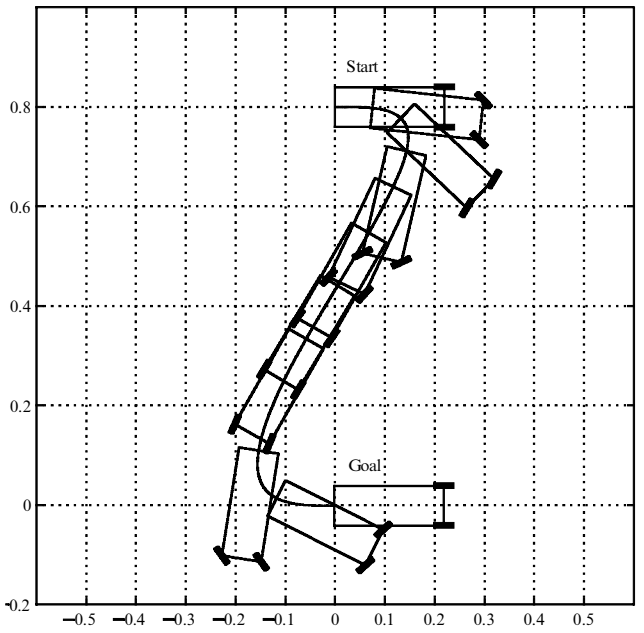

Figure 3. First simulation: minimization of the trajectory length. The robot trajectory during iteration 4 of the robust learning phase is shown.

In the second simulation, the robot is assigned the same steering task. However, a linear combination $H = H_1 + H_2/2$ is considered as a cost criterion in order to reduce the maximum value of the steering angle while still guaranteeing a reasonable length of the trajectory. The value of $\phi_{\max}$ has been set to 30°. The same control structure of the first simulation has been chosen. Details on the performance of the algorithm are reported in table 2. Again, in the optimal learning phase, the positioning error is zeroed in one iteration, while subsequent iterations only improve the performance criterion. The robust learning phase then robustifies the nominal trajectory, at the expense of a small increase of the cost function. The final trajectory is shown in figure 4. Note that it is considerably longer than in the previous simulation, on account of the presence of function H_2 in the cost criterion. The final value of H is 6.73, with $H_1 = 5.96\,\mathrm{m}$ and $H_2 = 18.5^\circ$.

Optimal learning phase					Robust learning phase		
Iteration	$H = H_1 + H_2/2$	H_1	max $\phi(t)$	$\|\varepsilon\|$	Iteration	$\|\varepsilon\|$	H
1	8.52	5.31	38.02	0.00	1	0.08	6.94
2	6.47	5.08	25.02	0.00	3	0.01	6.68
3	6.39	5.41	20.99	0.00	4	0.00	6.73

Table 2. Second simulation: details of the learning process.

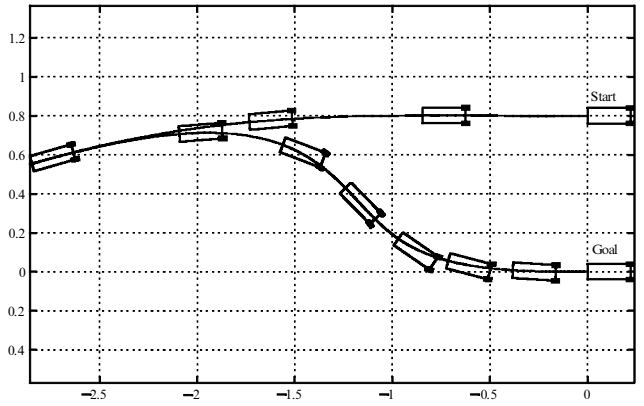

Figure 4. Second simulation: minimization of the trajectory length in the presence of a steering saturation. The robot trajectory during iteration 4 of the robust learning phase is shown.

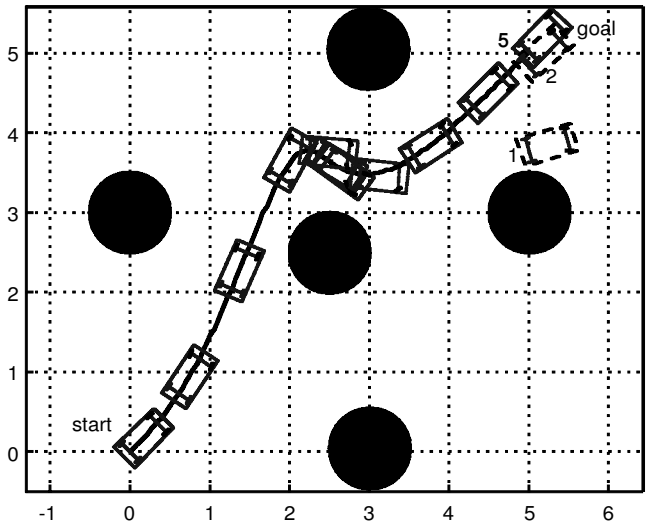

Figure 5. Third simulation: generation of a collision-free trajectory. The robot trajectory during iteration 5 of the robust learning phase is shown. Also, the robot configurations at the end of iterations 1 and 2 of the robust learning phase are shown dashed.

Optimal learning phase		Robust learning phase	
Iteration	H_3	Iteration	$\|\varepsilon\|$
1	0.6746	1	1.4198
2	0.2154	2	0.2071
3	0.0589	3	0.0356
5	0.0386	4	0.0566
6	0.0000	5	0.0229

Table 3. Third simulation: details of the learning process.

An environment with five circular obstacles is considered in the third simulation. The car-like robot must move from $q^0 = (0,0,45^\circ,0^\circ)$ (start) to $q^{\mathrm{d}} = (5,5,45^\circ,0^\circ)$ (goal) in $T = 10\,\mathrm{s}$. The control structure is similar to that of the previous cases, with the difference that a third-degree polynomial is used in the second time subinterval to allow for a larger number of parameters. The results are shown in figure 5. Details of the performance of the algorithm are reported in table 3. In the optimal learning phase, a collision-free nominal trajectory is generated in six iterations by minimizing the cost criterion H_3. In the robust learning phase, five iterations are needed to achieve exact steering on the perturbed model.

In the fourth simulation, the car-like robot must move from $q^0 = (-2,1,0^\circ,0^\circ)$ (start) to $q^{\mathrm{d}} = (0,0,0^\circ,0^\circ)$ (goal) in $T = 10\,\mathrm{s}$, tracking as closely as possible a given trajectory $q^{\mathrm{d}}(t)$, $t \in [0,T]$, generated off-line. Hence, the cost function $H_4(c)$ is used in the learning process. The control structure is the same as in the first two simulations. The results after ten iterations of optimal learning and ten iterations of robust learning are shown in figure 6 and 7. As expected, the learning paradigm allows us to obtain robust steering, guaranteeing at the same time that the generated trajectory conforms as much as possible to the desired one.

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 $|\theta^0 - \theta^{\mathrm{d}}| < \pi$. 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 $|\theta^0 - \theta^{\mathrm{d}}| = \pi$, it is necessary to introduce an *intermediate* desired configuration q^i, e.g. a configuration such that $\theta^i = (\theta^0 + \theta^{\mathrm{d}})/2$. 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

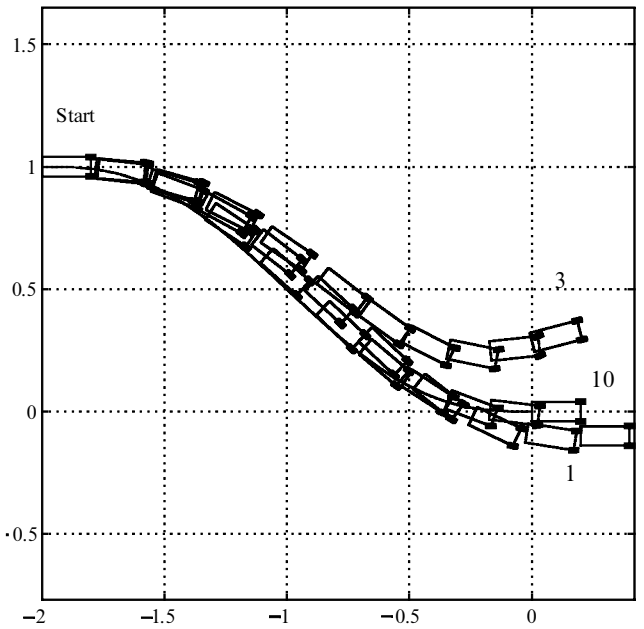

Figure 6. Fourth simulation: tracking a given trajectory. The robot trajectories during iterations 1, 3 and 10 of the robust learning phase are shown.

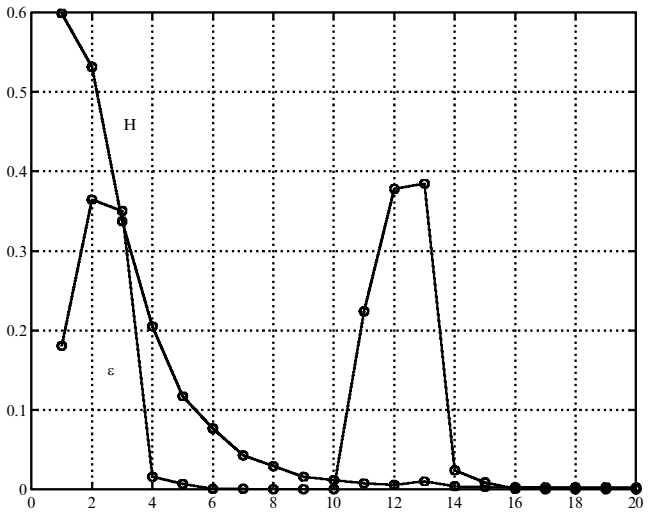

Figure 7. Fourth simulation: tracking a given trajectory. Evolution of the error norm $\|\varepsilon\|$ and of the cost function H along the iterations.

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. Conclusion

We have presented a solution to the problem of planning robust trajectories which steer a non-holonomic robot between given configurations in the presence of perturbations, at the same time optimizing (at least locally) a given performance criterion. The proposed approach makes use of an overparametrized iterative process in order to learn the optimal control input that guarantees exact steering. To this end, a finite-dimensional class of control function is chosen. At the end of each iteration, the control coefficient parameter is updated on the basis of the final positioning error as well as of the current value of the performance criterion.

As a case study, we have considered the car-like robot, explicitly deriving the effect of typical model uncertainties on the chained-form representation in order to prove that the proposed learning algorithm can reject such perturbations. Simulation results have been reported to show the performance of the method. In particular, we have addressed the synthesis of trajectories that minimize the trajectory length while avoiding steering saturations. Moreover, we have proved that the same approach can be used to plan robust trajectories in the presence of obstacles or to track pre-planned trajectories that join the start with the goal.

Work in progress includes the implementation of this approach on the car-like robot MARIO available in our laboratory. This low-cost prototype displays the typical problems of practical vehicles, such as friction, backlash and wheel deformation, and therefore provides an ideal benchmark for testing the robustness of our controller. While preliminary results of the proposed learning method are satisfactory, the application of the proposed technique has required the solution of a number of relevant problems, indicating at the same time further directions of research:

- In the presence of slippage, use of dead reckoning (i.e. integration of the wheel encoders' data) for measuring of the state variables needed for the feedback transformation invariably results in the accumulation of large errors, which may hinder the convergence of the algorithm. Therefore, it is necessary to endow the robot with a localization system, which may use on-board or off-board sensors. To this end, we have realized a vision system based on a fixed camera.
- The kinematic model (chained forms) adopted for devising our controller fully accounts for the non-holonomic constraints, which are the main source of non-linearity, but ignores pure dynamic effects and, more importantly, the actuator models. In particular, the piecewise-constant input profile (8) cannot be realized in practice. The latter problem can be solved by choosing a piecewise-trapezoidal law for v_1, because the corresponding velocity profile for u_1 is continuous and can be tracked much more accurately by the driving velocity servo. The side effect of this modification is that

matrix $A(t)$ in equation (7) becomes piecewise-linear in t, so that the expressions of matrices V_i and W_i must be obtained through formulae for linear time-varying systems. In any case, it would be appropriate to investigate the possibility of applying the present approach to dynamic models of non-holonomic systems using suitable canonical forms (e.g. dynamically extended chained forms).

- Exact system re-initialization at each iteration—a common requirement of learning controllers—would require the availability of an accurate *homing* procedure, limiting the applicability of the proposed technique in practical applications. 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 over time. If the sequence consists of the start and the goal configuration, the robot simultaneously learns exact steering and re-initialization. The inclusion of this feature in the method presented is straightforward and can be achieved following Oriolo *et al.* (1998).

On the theoretical side, another avenue of research is the use of different optimization techniques during the learning process. Among the possible choices, we mention the reduced gradient method or a constrained version of the conjugate gradient technique (Luenberger 1984); these alternative methods may lead to an improvement in the convergence rate.

The approach presented can be applied to other robotic systems. For example, the optimal learning controller for linear systems described in § 3.1 was successfully used for steering a flexible robot arm (Oriolo *et al.* 1996). As for non-holonomic systems, the extension to more complicated wheeled mobile robots, such as a car-like robot towing an arbitrary number of trailers, can be worked out following very closely the derivations in this paper. Moreover, 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 of the naturalness of the learning paradigm for these systems.

Acknowledgments

This work was partially supported by MURST within the *RAMSETE* Project.

References

ARIMOTO, S., KAWAMURA, S., and MIYAZAKI, F., 1984, Bettering operation of robots by learning. *Journal of Robotic Systems*, **1**, 123–140.

BLOCH, A. M., REYHANOGLU, M., and MCCLAMROCH, N. H., 1992, Control and stabilization of non-holonomic mechanical systems. *IEEE Transactions on Automatic Control*, **37**, 1746–1757.

BONDI, P., CASALINO, G., and GAMBARDELLA, L., 1988, On the iterative learning control theory of robotic manipulators. *IEEE Journal of Robotics and Automation*, **4**, 14–22.

DE LUCA, A., PAESANO, G., and ULIVI, G., 1992, A frequency-domain approach to learning control: Implementation for a robot manipulator. *IEEE Transactions on Industrial Electronics*, **39**, 1–10.

DE LUCA, A., ORIOLO, G., and SAMSON, C., 1998, Feedback control of a non-holonomic car-like robot. In J.-P. Laumond (Ed), *Robot Motion Planning and Control*, Lecture Notes in Control and Information Sciences (London, Springer-Verlag), **229**, 171–253. Also available at http://www.laas.fr/~jpl/book.html.

HAHN, W., 1967, *Stability of Motion* (Berlin, Springer-Verlag).

ISIDORI, A., 1995, *Nonlinear Control Systems* (London, Springer-Verlag).

LUCIBELLO, P., 1994, State steering by learning for a class of non-linear control systems. *Automatica*, **30**, 1463–1468.

LUENBERGER, D. G., 1984, *Linear and Nonlinear Programming* (Reading, MA: Addison-Wesley).

MONACO, S., and NORMAND-CYROT, D., 1992, An introduction to motion planning under multirate digital control. *Proceedings of the 31st IEEE Conference on Decision and Control*, Tucson, AZ, pp. 1780–1785.

MOORE, K. L., 1998, Iterative learning control — An expository overview. *Applied and Computational Controls, Signal Processing, and Circuits*, **1**, 425–488.

MURRAY, R. M., 1993, Control of non-holonomic systems using chained forms. *Fields Institute Communications*, **1**, 219–245.

MURRAY, R. M., LI, Z., and SASTRY, S. S., 1994, *A Mathematical Introduction to Robotic Manipulation* (Boca Raton, CRC Press).

MURRAY, R. M., and SASTRY, S. S., 1993, Nonholonomic motion planning: Steering using sinusoids. *IEEE Transactions on Automatic Control*, **38**, 700–716.

ORIOLO, G., PANZIERI, S., and ULIVI, G., 1996, Finite-dimensional optimal learning control: application to a flexible link. *Proceedings of the 4th IEEE Mediterranean Symposium on New Directions in Control and Automation*, Chania, Greece, pp. 687–692.

ORIOLO, G., PANZIERI, S., and ULIVI, G., 1998, An iterative learning controller for non-holonomic mobile robots. *International Journal of Robotics Research*, **17**, 954–970.

RAO, C. R., and MITRA, S. K., 1971, *Generalized Inverse of Matrices and its Applications* (New York, John Wiley and Sons).

SONTAG, E. D., 1990, *Mathematical Control Theory* (New York, Springer-Verlag).

TILBURY, D., MURRAY, R. M., and SASTRY, S. S., 1995, Trajectory generation for the N-trailer problem using Goursat normal form. *IEEE Transactions on Automatic Control*, **40**, 802–819.