Learning Nice Robust Trajectories for a Car-Like Robot

Enrico Ferretti*, **Giuseppe Oriolo***, **Stefano Panzieri**†, **Giovanni Ulivi**†

* Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza"
Via Eudossiana 18, 00184 Roma, Italy

† Dipartimento di Discipline Scientifiche, Terza Università di Roma
Via della Vasca Navale, 84, 00146 Roma, Italy

{ferretti,oriolo,panzieri,ulivi}@labrob.ing.uniroma1.it

Abstract

A finite-dimensional iterative learning controller is presented that can steer a car-like robot in finite time between given configurations while optimizing a given performance criterion. The robustness property of the controller guarantees that the obtained trajectory achieves exact steering even in the presence of perturbations with respect to the nominal model. By properly choosing the performance criterion, it is possible to minimize the trajectory length, to avoid saturations of the steering angle as well as to generate collision-free trajectories among workspace obstacles. Simulation results are reported to show the satisfactory performance of the method.

1 Introduction

The problem of moving a a car-like robot between given configurations with trajectories that are optimal in some sense but at the same time can be executed in perturbed conditions is quite entangled. On the one hand, a number of planning algorithms have been proposed for generating trajectories that give rise to natural and smooth motions (e.g., see [1]). On the other hand, these trajectories are computed in open-loop, and a feedback action is mandatory to obtain real-time accurate tracking. However, the peculiar features of the car-like robot, and namely its being a nonholonomic system, make the synthesis of trajectory tracking controllers difficult. In particular, to prove asymptotic stability one is forced to make the somewhat unrealistic assumption that the trajectory never stops [2].

In this paper, we propose a solution approach based on the use of a learning controller. Learning is a methodology for improving the performance of a control system through iterative training [3]. A general framework for solving the finite-time steering problem through learning has been proposed in [4]; specific applications include nonholonomic mobile robots [5] and flexible structures [6, 7].

Our learning algorithm makes use of a *parameterized* control chosen in a finite-dimensional class. In particular, the input vector is the linear combination of a suitable set of generating functions. By augmenting the dimension of the coefficient vector with respect to the number of the state variables n, we obtain an overparameterized controller that can meet other performance goals, typically expressed as cost criteria to be minimized. The main advantages of this approach are *(i)* the fact that it is not necessary to provide a trajectory connecting the initial and final configurations, *(ii)* the flexibility in the choice of the trajectory cost criterion, and *(iii)* the robustness properties of the obtained trajectories against disturbances and model perturbations.

In particular, we shall apply the above approach in order to plan robust trajectories that minimize the total length, the maximum steering angle or the distance from workspace obstacles inside the chosen control class. This is obtained by expressing these cost criteria as functions of the control parameters and performing a constrained optimization process in the parameter space. Simulations results for a laboratory prototype are reported in order to show the effectiveness of the approach.

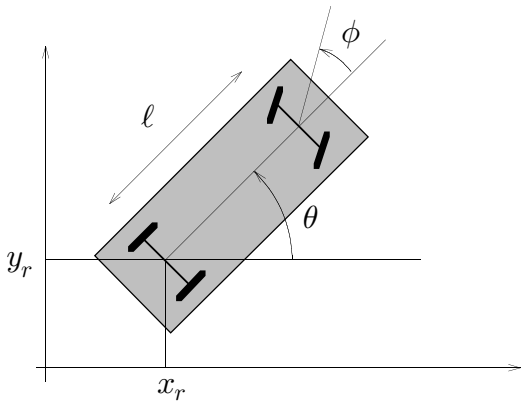

Fig. 1.: Generalized coordinates for a car-like robot

2 Steering the car-like robot in chained form

Consider a robot with the kinematic model of an automobile (Fig. 1). 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 w.r.t. the x axis, and ϕ is the steering angle.

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

$$\begin{aligned} \dot{x}_r(t) &= \cos\theta(t)\rho\, u_1(t), \\ \dot{y}_r(t) &= \sin\theta(t)\rho\, u_1(t), \\ \dot{\theta}(t) &= \tfrac{1}{\ell}\tan\phi(t)\rho\, u_1(t), \\ \dot{\phi}(t) &= u_2(t), \end{aligned} \tag{1}$$

where ρ is the wheel radius, u_1 is the angular velocity of the driving wheel, and u_2 is the steering rate.

Using the following change of coordinates

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

together with the input transformation

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

the transformed system is in *chained form* [8]

$$\begin{aligned} \dot{z}_1(t) &= v_1(t), \\ \dot{z}_2(t) &= v_2(t), \\ \dot{z}_3(t) &= z_2(t)v_1(t), \\ \dot{z}_4(t) &= z_3(t)v_1(t). \end{aligned} \tag{2}$$

Note that if we assign v_1 as a signal $v_1(t)$, the system becomes linear and behaves like a chain of integrators with time-varying gain from z_2 to z_n, driven by the input v_2.

Assume we wish to steer the car-like robot (1) from q^0 to q^d in a finite time T. Correspondingly, the chained form (2) must move from $z^0 = z(q^0)$ to $z^d = z(q^d)$. A possible control law [9] is

$$v_1(t) = \alpha(t), \tag{3}$$

$$v_2(t) = \beta_1 + \beta_2 t + \beta_3 t^2, \tag{4}$$

with $\alpha(t)$ a piecewise-constant function and $\beta_1, \beta_2, \beta_3$ constants. In fact, if $\alpha(t)$ satisfies

$$\int_0^T \alpha(t)dt = z_1^d - z_1^0,$$

the β_i's can be found by imposing the desired reconfiguration on the variables z_2, z_3, z_4. To this end, one may use the linear relationship obtained by forward integration of eq. (2):

$$M(\alpha(t), T) \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + m(z^0, \alpha(t), T) = \begin{bmatrix} z_2^d \\ z_3^d \\ z_4^d \end{bmatrix},$$

where $M(\alpha(t), T)$ is a nonsingular matrix provided that $\alpha(t) \neq 0$, for some $t \in [0, T]$.

The mixed piecewise-constant/polynomial control law (3–4) produces smooth robot trajectories, but has a main drawback: being essentially open-loop, it provides no robustness properties. As a consequence, it will not produce exact steering when applied to an actual car-like robot whose model is different from the ideal model (1). In the next section, we show how it is possible to achieve robustness with respect to repetitive disturbances and model perturbations through an iterative learning scheme, improving at the same time the value of a given performance criterion.

3 Computing robust trajectories via learning

Partition the state vector z as (z_a, z_b), with $z_a = z_1$ and $z_b = (z_2, z_3, z_4)$. System (2) is conveniently rewritten as

$$\dot{z}_a(t) = v_1(t), \tag{5}$$
$$\dot{z}_b(t) = A(v_1(t))z_b(t) + Bv_2(t), \tag{6}$$

with

$$A(v_1(t)) = \begin{bmatrix} 0 & 0 & 0 \\ v_1(t) & 0 & 0 \\ 0 & v_1(t) & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

Assume that system (5–6) is perturbed as

$$\dot{z}_a(t) = v_1(t) + \eta_a g_a(z(t), v(t), t), \tag{7}$$
$$\dot{z}_b(t) = A(v_1(t))z_b(t) + Bv_2(t) + \eta_b g_b(z(t), v(t), t), \tag{8}$$

where $\eta = (\eta_a, \eta_b) \in I\!R^n$ is a small parameter vector and $g = (g_a, g_b) \in I\!R^n$ is a continuous function of z and v. It is relatively easy to show that many nonidealities of the kinematic model (1) result in a perturbed chained-form of this kind; e.g., perturbations on the wheel radius, the vehicle length, and others.

Divide the time interval $[0, T]$ in $p \geq 1$ subintervals $[t_{i-1}, t_i)$, with $t_0 = 0$ and $t_p = T$, and let $\delta_i = t_i - t_{i-1}$ be the duration of the i-th subinterval. The control law (3–4) may be generalized as

$$v_1(t) = c_{1i}, \qquad \forall t \in [t_{i-1}, t_i),\ i = 1, \ldots, p,$$
$$v_2(t) = \sum_{j=1}^{q} c_{2j}\lambda_j(t) \quad \forall t \in [0, T],$$

where $\lambda_j(t)$, $j = 1, \ldots, q$ are piecewise-polynomial functions with discontinuities occurring only at the time instants t_i, $i = 1, \ldots, p$.

Once the set of generating functions $\{\lambda_j(t)\}$ has been chosen, the structure of the steering control law is assigned and the learning process consists in the computation of the two control coefficients $c_1 \in I\!R^p$ and $c_2 \in I\!R^q$ by repeated trials. Assume that, at the end of each iteration, the system is exactly re-initialized at z^0. The k-th final positioning error is $\varepsilon = (\varepsilon_a, \varepsilon_b)$, with

$$\varepsilon_a^{[k]} = z_a^d - z_a^{[k]}(T),$$
$$\varepsilon_b^{[k]} = z_b^d - z_b^{[k]}(T).$$

Define the matrices

$$V_i^{[k]} = e^{A(c_{1i}^{[k]})\delta_i}, \qquad W_i^{[k]} = \int_0^{\delta_i} e^{A(c_{1i}^{[k]})(\delta_i - \tau)} B[\lambda_1(\tau + t_{i-1}) \ldots \lambda_q(\tau + t_{i-1})]d\tau, \qquad i = 1, \ldots, p,$$
$$V^{[k]} = V_p^{[k]} \ldots V_1^{[k]}, \qquad W^{[k]} = W_p^{[k]} + V_p^{[k]} W_{p-1}^{[k]} + \ldots + V_p^{[k]} \ldots V_2^{[k]} W_1^{[k]},$$

and note that they all depend on the current value $c_1^{[k]}$ of the first coefficient vector. We have the following result.

Proposition 1. *Let the coefficient vector* $c^{[k+1]} = (c_1^{[k+1]}, c_2^{[k+1]})$ *be updated according to*

$$c_1^{[k+1]} = c_1^{[k]} + \delta^{T\dagger}\varepsilon_a^{[k]} + \left(I - \delta^{T\dagger}\delta^T\right)\tilde{c}_1^{[k]}, \tag{9}$$

$$c_2^{[k+1]} = c_2^{[k]} + W^{[k+1]\dagger}\left[\varepsilon_b^{[k]} + \left(V^{[k]} - V^{[k+1]}\right)z_b^0 + \left(W^{[k]} - W^{[k+1]}\right)c_2^{[k]}\right] + \left(I - W^{[k+1]\dagger}W^{[k+1]}\right)\tilde{c}_2^{[k]}, \tag{10}$$

where $\delta^{T\dagger}$ *and* $W^{[k+1]\dagger}$ *denote the pseudoinverse of* δ^T *and* $W^{[k+1]}$*, respectively, and* $\tilde{c}^{[k]} = (\tilde{c}_1^{[k]}, \tilde{c}_2^{[k]}) \in I\!R^{p+q}$ *is an arbitrary vector. Then:*

1. *For the nominal system (5–6), it is* $\varepsilon^{[k+1]} = 0$*.*
2. *For the perturbed system (7–8) and sufficiently small* η*,* ε *converges to zero uniformly and exponentially over the iterations, as long as* $\tilde{c}_1^{[k]}$ *and* $\tilde{c}_2^{[k]}$ *are both* $o(||\varepsilon^{[k]}||)$*.*

For the proof, the reader is referred to [5, 7]. The following remarks are in order.

– Since both matrices δ^T and $W^{[k+1]}$ are of full row rank by construction, their pseudoinverses are computed in closed form as [10]

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

Note also that $I - \delta^{T\dagger}\delta^T$ and $I - W^{[k+1]\dagger}W^{[k+1]}$ in eqs. (9–10) are the orthogonal projection matrices in the null space of δ^T and $W^{[k+1]}$, respectively. Their presence guarantees that the constraint $z(T) = z^d$ is satisfied for any choice of the null space vectors $\tilde{c}_1^{[k]}$ and $\tilde{c}_2^{[k]}$.

– Setting $\tilde{c}^{[k]} = 0$ in eqs. (9–10), one obtains the minimum-norm *update* $c^{[k+1]} - c^{[k]}$ of the coefficient vector that solves the steering problem for the nominal system. In particular, if the coefficient vector is initialized at $c^{[0]} = 0$, the choice

$$c_1^{[1]} = \delta^{T\dagger}(z_a^d - z_a^0), \qquad c_2^{[1]} = W^{[1]\dagger}\left(z_b^d - V^{[1]}z_b^0\right)$$

yields directly the minimum-norm coefficient vector that solves the nominal steering problem.

– The proof of the robustness property entailed by Prop. 1 relies on the positioning error dynamics being uniformly and exponentially stable. This implies that small non-persistent perturbations of the error dynamics are rejected, while small persistent perturbations give limited errors [11]. Thus, in order to obtain the result, it suffices to show that the model perturbations considered in eqs. (7–8) give rise to non-persistent perturbations of the error dynamics.

4 Optimization of performance criteria

The null-space vector $\tilde{c}^{[k]}$ in eqs. (9–10) may be easily selected so as to optimize a given performance criterion. In particular, assume that we wish to learn a control input—inside the chosen class—that minimizes a cost criterion $H(c)$ along the steering interval. Possible choices for H in robotic systems include actuator effort (e.g., maximum or integral force/torque [7]), trajectory optimality criteria (e.g., curvature), avoidance of mechanical joint limits and obstacles, and others. Specific cost criteria of interest for the car-like robot are presented later in this section.

4.1 Choice of the null-space vector

An effective choice for the null-space vector is

$$\tilde{c}_1^{[k]} = -\alpha_1\nabla_{c_1}H|_{c^{[k]}}, \qquad \tilde{c}_2^{[k]} = -\alpha_2\nabla_{c_2}H|_{c^{[k]}},$$

where $\alpha = (\alpha_1, \alpha_2) \in I\!R^n$ is a vector of positive real numbers and $\nabla_{c_j}H$ is the gradient of H w.r.t. the coefficient subvector c_j, $j = 1, 2$.

Proposition 1 states that, in order to guarantee exact steering for the perturbed system, $\tilde{c}_1^{[k]}$ and $\tilde{c}_2^{[k]}$ should be $o(||\varepsilon^{[k]}||)$. This can be realized by weighting the stepsizes α_1 and α_2 with the positioning error norm. By doing so, however, 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*).
2. A phase of pure learning (i.e., with $\tilde{c}^{[k]} = 0$ or with $\tilde{c}^{[k]} = o(||\varepsilon^{[k]}||)$, for any k) is performed through experiments on the perturbed model so as to make the nominal trajectory robust (*robust learning phase*).

In the optimal learning phase, the use of eqs. (9-10) yields zero positioning error for the nominal system in one iteration. In successive iterations, it will be $\varepsilon^{[k]} = 0$ and c will basically evolve as prescribed by the *projected gradient* optimization method [12]. The latter is not guaranteed to reach the true optimal solution, for it may stop in a local constrained minima (where the projected gradient is zero). Hence, the characteristics of the cost function H play a crucial role for the effectiveness of the optimization process. In any case, the inclusion of a *line search* procedure for the choice of the stepsizes α_1 and α_2 is mandatory to guarantee nice convergence properties. For example, an Armijo-like test may be adopted [12].

4.2 Cost criteria for the car-like robot

The possibility to optimize the learning process with respect to a given criterion has been exploited in order to achieve three further alternative objectives beside the steering one:

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.

The length of the trajectory is simply expressed as

$$H_1(c) = \int_0^T \rho\, u_1(t) dt.$$

To reach the second objective, we have chosen the following penalty function

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

that takes into account only the maximum value of $|\phi|$ along the trajectory. By increasing p, maximum values of $|\phi|$ above $\phi_{\max}$ are penalized more, while maximum values of $|\phi|$ below $\phi_{\max}$ give a smaller contribute. Note that H_2 is continuous but not differentiable. However, this is not a problem, because its gradient is computed numerically.

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 suitable cost criterion that penalizes collision with obstacles along the trajectory $\mathcal{T}$ is

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

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

No closed form is available for H_1, H_2 and H_3 as a function of the control coefficient vector c. This is due to the presence of the input transformation between the auxiliary control input v (that depends directly on c) and the actual control input u. Therefore, the gradient must be computed numerically by function evaluations in the neighborhood of the current coefficient vector $c^{[k]}$. In turn, these may be performed either via simulation or, for the perturbed system, by means of experiments.

5 Simulation results

To test the performance of the proposed method, we have considered three simulations with different trajectory cost criteria. In particular, in the first two the car-like robot is assigned a parallel parking task, while in the third it must reach a given configuration across a cluttered workspace.

For the considered robot, it is $\ell = 0.5$ m and $\rho = 0.05$ m in the nominal model. To obtain a realistic perturbed model, we have included the following nonidealities:

- perturbations of 5% and 10% on the values of ℓ and ρ, respectively;
- a digital version of the controller has been implemented with a sampling time of 0.025 sec;
- a simulated encoder has been used to measure the wheels' angular position.

In all the simulations, an optimal learning phase is performed on the nominal model to compute a trajectory that minimizes 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 $q^0 = (0, 1, 0°, 0°)$ (`start`) to $q^d = (0, 0, 0°, 0°)$ (`goal`) in $T = 10$ sec, minimizing the trajectory length H_1. The structure of the control law has been chosen as follows:

$$v_1(t) = c_{1i}, \qquad v_2(t) = c_{21}^i t^2 + c_{22}^i t + c_{23}^i, \quad \forall t \in [t_{i-1}, t_i), \;\; i = 1, \ldots, 3,$$

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

In the optimal learning phase, the length of the trajectory is reduced to 1.43 m in the first iteration and does not change appreciably in subsequent iterations. When the nominal control thus computed is applied to the perturbed system, a large final error is experienced. Then, the robust learning phase takes over, reducing the error to zero in 7 iterations (see Fig. 2). The final value of H_1 is 1.76 m. Note that the robot undergoes a large variation of the steering angle, consistently with the objective of minimizing the trajectory length.

In the second simulation, the robot is assigned the same steering task. However, a linear combination $H = H_1 + 2H_2$ 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 15°. The same control structure of the first simulation has been chosen. Details on the performance of the algorithm are reported in Table 1. The final trajectory is shown in Fig. 3. 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 9.5, with $H_1 = 8.57$ m and $H_2 = 12.5°$.

An environment with five circular obstacles is considered in the third simulation. The car-like robot must move from $q^0 = (0, 0, 45°, 0°)$ (`start`) to $q^d = (5, 5, 45°, 0°)$ (`goal`) in $T = 10$ sec. The control structure is similar to 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. Details on the performance of the algorithm are reported in Table 2. In the optimal learning phase, a collision-free nominal trajectory is generated in 6 iterations by minimizing the cost criterion H_3. In the robust learning phase, 5 iterations are needed to achieve exact steering on the perturbed model.

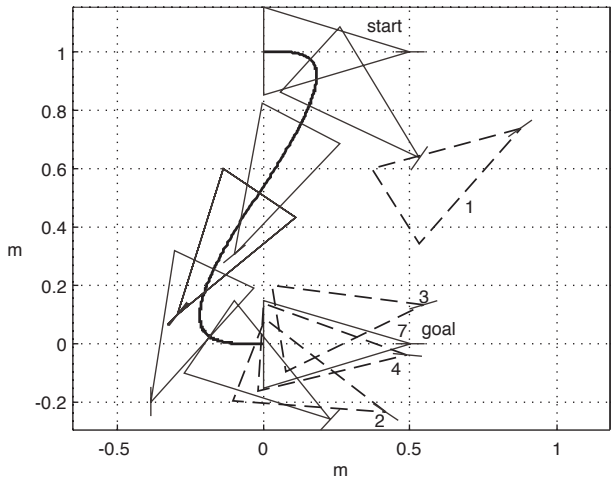

Fig. 2.: First simulation: Minimization of the trajectory length. The robot configurations at the end of iterations 1, 2, 3 and 4 of the robust learning phase are shown dashed.

Optimal Learning Phase			
iteration	H	H_1	H_2
1	1678.8	3.6	80.7
4	91.9	6.2	38.4
8	11.2	10.1	13
12	11.1	9	15.1
16	9.8	8.5	13.3

Robust Learning Phase	
iteration	$\|\varepsilon\|$
1	0.4559
2	0.2624
3	0.0764
4	0.0356
6	0.0070

Table 1.: Second simulation: Details of the learning process

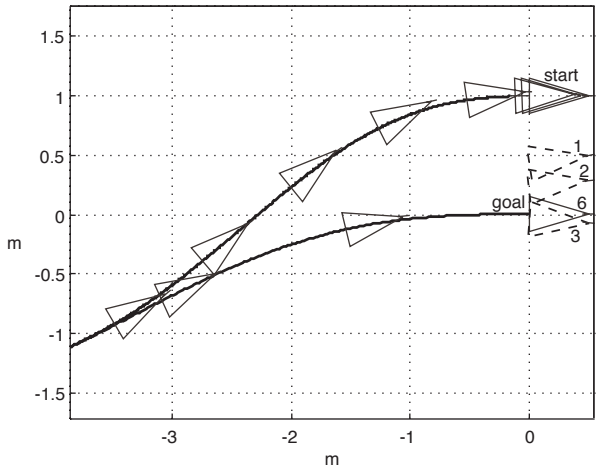

Fig. 3.: Second simulation: Minimization of the trajectory length in the presence of a steering saturation. The robot configurations at the end of iterations 1, 2 and 3 of the robust learning phase are shown dashed.

6 Conclusion

We have presented a solution to the problem of planning nice, robust trajectories for steering a car-like robot between given configurations. Here, *nice* means that the obtained trajectory optimizes—at least locally—a performance criterion inside the chosen control structure, while *robust* indicates that the robot is able to track it to reach the final destination even in the presence of perturbations w.r.t. the nominal model.

The proposed approach makes use of an 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.

Simulation results have been reported to show the method performance. 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.

Work in progress includes the implementation of this approach on the car-like robot MARIO available in our lab. This low-cost prototype displays the typical problems of practical vehicles, such as friction, backlash, wheel deformation, etc. Besides, the measure of the state variables necessary for the feedback transformation comes from the odometric system, that can be a source of large errors. Nevertheless, preliminary results of the application of the proposed learning method are satisfactory [5].

Finally, it is noteworthy that the necessity of exact system re-initialization at each experiment can be overcome by devising a *cyclic* learning controller, as shown in [13].

7 Acknowledgments

This work was partially supported by MURST under 40% funds and by ENEA Antarctica Project.

Optimal Learning Phase	
iteration	H_3
1	0.6746
2	0.2154
3	0.0589
5	0.0386
6	0.0000

Robust Learning Phase	
iteration	$\|\|\varepsilon\|\|$
1	1.4198
2	0.2071
3	0.0356
4	0.0566
5	0.0229

Table 2.: Third simulation: Details of the learning process

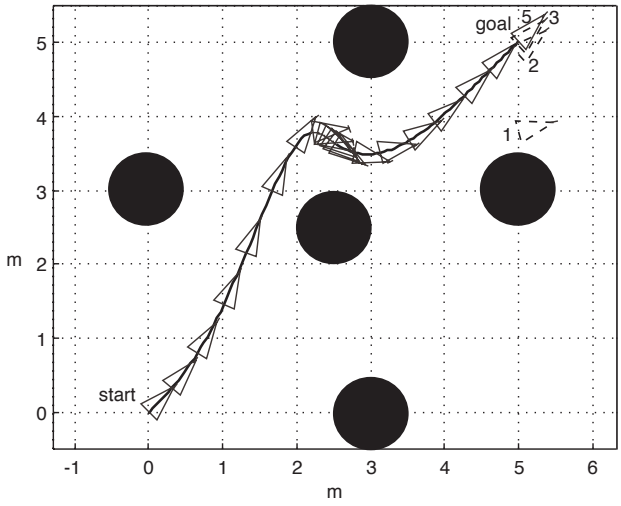

Fig. 4.: Third simulation: Generation of a collision-free trajectory. The robot configurations at the end of iterations 1, 2 and 3 of the robust learning phase are shown dashed.

References

1. J.-C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA, 1991.
2. C. Samson and K. Ait-Abderrahim, "Mobile robot control, part 1: Feedback control of a nonholonomic wheeled cart in cartesian space," Tech. Rep. 1288, INRIA Sophia-Antipolis, 1990.
3. P. Bondi, G. Casalino, and L. Gambardella, "On the iterative learning control theory of robotic manipulators," *IEEE J. of Robotics and Automation*, vol. 4, pp. 14–22, 1988.
4. P. Lucibello, "State Steering by Learning for a Class of Nonlinear Control Systems," *Automatica*, vol. 30, no. 9, pp. 1463–1468, 1994.
5. G. Oriolo, S. Panzieri, and G. Ulivi, "An iterative learning controller for nonholonomic robots," *1996 IEEE Int. Conf. on Robotics and Automation*, Minneapolis, MN, Apr. 1996.
6. P. Lucibello, S. Panzieri, G. Ulivi, "Experiments on repositioning learning control of a flexible link under gravity," *4th IFAC Symp. on Robot Control*, Capri, I, pp. 797–802, 1994.
7. G. Oriolo, S. Panzieri, and G. Ulivi, "Finite-dimensional optimal learning control: Application to a flexible link," *4th IEEE Mediterranean Symp. on New Directions in Control & Automation*, Chania, GR, Jun. 1996.
8. R. M. Murray and S. S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *IEEE Trans. on Automatic Control*, vol. 38, pp. 700–716, 1993.
9. D. Tilbury, R. M. Murray, and S. S. Sastry, Trajectory generation for the N-trailer problem using Goursat normal form, Memo. UCB/ERL M93/12, University of California at Berkeley, Feb. 1993.
10. C. R. Rao and S. K. Mitra, *Generalized Inverse of Matrices and its Applications*, John Wiley and Sons, New York, 1971.
11. W. Hahn, *Stability of Motion*, Springer-Verlag, 1967.
12. D. G. Luenberger, *Linear and Nonlinear Programming*, 2nd edition, Addison-Wesley, Reading, 1984.
13. G. Oriolo, S. Panzieri, and G. Ulivi, "Cyclic Learning Control of Chained-Form Systems with Application to Car-Like Robots," *13th IFAC World Congress*, San Francisco, CA, Jun. 1996.

This article was processed using the TEX macro package with SIRS96 style