Motion Planning and Control for Hilare Pulling a Trailer

F. Lamiraux, S. Sekhavat and J.P. Laumond, *Member, IEEE*

Abstract— **This paper deals with motion planning and control for mobile robots. The various components of an integrated architecture for the mobile robot Hilare pulling a trailer are presented. The nonholonomic path planner is based on an original steering method accounting for the small-time controllability of the system. Then the path is transformed into a trajectory by including the dynamical constraints of the system (bounded velocity and bounded acceleration). Finally motion control is addressed: thanks to a geometric transformation introducing a virtual robot, we show how to reduce the problem to a classical approach of trajectory tracking for a mobile robot moving only forward. Experimental results include two types of robot-trailer connection systems.**

Keywords— **Mobile robot with trailer, motion control, Nonholonomic path planning.**

I. Introduction

This paper presents all the components required to devise a practical navigation system for a mobile robot pulling a trailer in a known environment. Experiments involving the mobile robot Hilare are reported. Two different robot-trailer connection systems are considered: on System A (Figure 1 left), the trailer is hooked up above the wheel axis of the robot, whereas on System B (Figure 1 right), the trailer is hooked up behind the wheel axis.

We assume that the mobile robot moves sufficiently slowly ($.5ms^{-1}$, $.5rad\, s^{-1}$) to make all the dynamical effects negligible (no slippage). We do not consider any restriction on the shape of the obstacles. The inputs of our system are a geometric map of the environment, the type of robot-trailer connection, and bounds on the linear and angular velocities and accelerations. The output is the execution of a motion in the real world.

For the past eight years, mobile robots with trailers have been fruitful examples to support advanced researches on control of nonholonomic systems (e.g., [28], [19], [27]). Such researches address either open loop control or feedback control. Most of them ignore the obstacle avoidance problem and few of them report experiments on a real system. In [38] or [6] the experiments are carried out only to evaluate various feedback control laws. There is no path planning. [30] presents experiments for a robot with a trailer in a specific environment composed of corridors. The robot and the trailer are grown in such a way that they can be considered as a car-like system. [12] presents an experimental system consisting of a car with one or two trailers. This work addresses the planning and the execution of maneuvers such as parallel parking and docking.

Manuscript received June 2, 1997; revised April 14, 1999. This work was recommended for publication by Editor A. De Luca upon evaluation of the reviewers' comments. This paper was supported in part by Alcatel-Telecom.
F. Lamiraux is with the Department of Computer Science, Rice University, Houston, TX 77005 USA (e-mail: lamiraux@cs.rice.edu). S. Sekhavat is with INRIA, Grenoble, France (e-mail: Sepanta.Sekhavat@inrialpes.fr). J.P. Laumond is with LAAS-CNRS, Toulouse 31077, France (e-mail: jpl@laas.fr).
Publisher Item Identifier S 1042-296X(99)05604-9.

In this paper, we present an integrated approach to path planning and motion execution for a mobile robot with a trailer. Our approach may face any constrained environment while taking into account bounds on both velocities and accelerations of the robot. A short version of this paper can be found in [46].

The paper is organized according to the three main components of our integrated system: computation of a *collision-free feasible* path[1], transformation of this path into a *feasible* trajectory[2] and trajectory tracking. We will report prior work related to each step in the corresponding sections.

Nonholonomic path planning (Section III). Both systems A and B are small-time controllable[3]. Small-time controllability means that the set of configurations reachable after any given time always contains a neighborhood of the starting configuration. As a consequence any collision-free path can be approximated by a sequence of collision-free feasible paths. Then nonholonomic path planning can be addressed by dealing separately with the physical constraints due to the obstacles and the kinematic constraints due to the wheels. This approach was first proposed in [26] for a car-like robot. At this level our contribution is to propose a new open-loop steering method accounting for the small-time controllability of the systems. We will show that this property is crucial for the convergence of the algorithm.

Transformation of a feasible path into a feasible trajectory (Section IV). Following a path for an articulated system subjected to bounds on velocities and accelerations requires an appropriate time-parameterization of the path. This issue has been addressed in the case of manipulators with acceleration limitations. We propose here a numerical method based on the algorithm proposed in [49], taking into account additional constraints implied by velocity bounds.

Motion control (Section V). The selected solution is based on a classical trajectory tracking control law for mobile

[1] A path is said to be *collision-free* if it does not collide the obstacles in the configuration-space. A path is said to be *feasible* if it respects the nonholonomic constraints.

[2] A trajectory is said to be *feasible* if it respects the dynamical constraints, i.e. acceleration and speed bounds.

[3] See [25] and references therein for a proof.

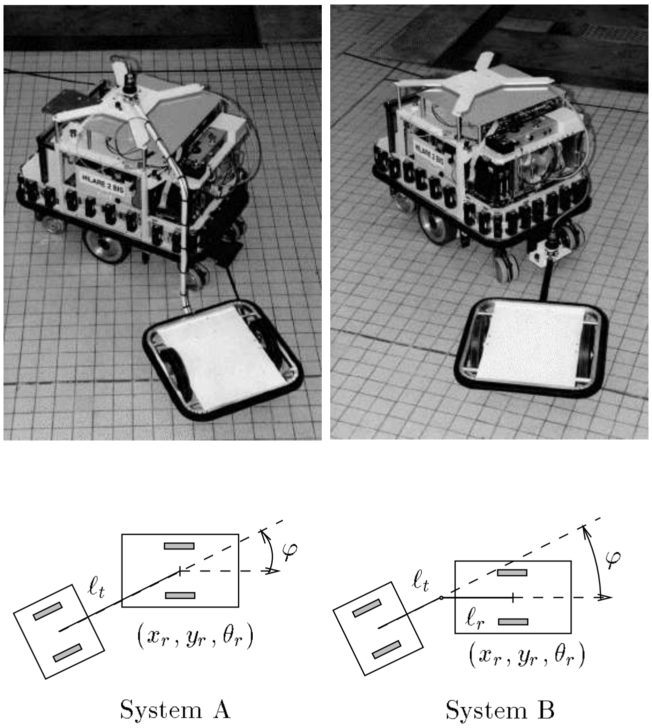


Fig. 1. Hilare with its trailer

robots [41]. Our contribution lies in the extension of this law to a mobile robot with trailer via geometric transformation allowing the tracking problem to be reduced to forward motion only. Robustness is analyzed through an iterative control scheme combining motion planning and trajectory tracking.

Experiments (Section VI). Last but not least, all these components have been integrated within a modular architecture (Section II) to perform experiments in an indoor environment with a real robot.

II. Hilare and its trailer

Our experimental platform is Hilare-2-bis. It is a two driving-wheel mobile robot belonging to the family of mobile robot Hilare growing at LAAS since 1976 [16]. In the following experimentation we use proprioceptive sensors: the odometer (based on optical encoders on dedicated wheels) gives the position (x_r, y_r) and the direction θ_r of the robot w.r.t. a starting configuration; an angular encoder gives the relative direction φ of the trailer w.r.t. the direction of the robot. Three cameras on the ceiling give the initial configuration of the system w.r.t. an absolute frame.

We consider two different systems A and B of robot-trailer connection (Figure 1). The corresponding control systems are given by:

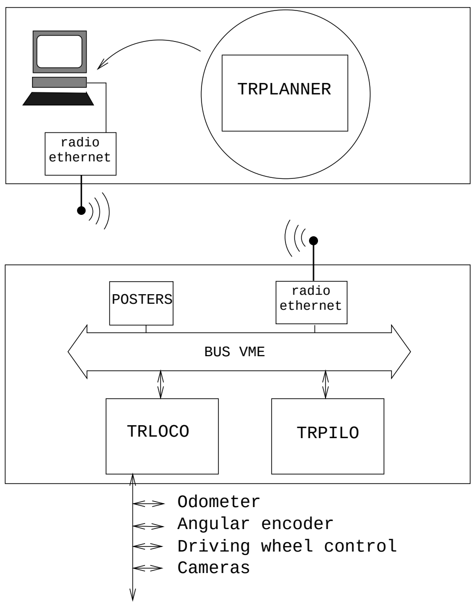


Fig. 2. Computer architecture of the experimental systems.

System A:

$$\left\{\begin{array}{rcl} \dot{x}_r &=& v_r\cos\theta_r \\ \dot{y}_r &=& v_r\sin\theta_r \\ \dot{\theta}_r &=& \omega_r \\ \dot{\varphi} &=& -\frac{v_r}{l_t}\sin\varphi - \omega_r \end{array}\right.$$

System B:

$$\left\{\begin{array}{rcl} \dot{x}_r &=& v_r\cos\theta_r \\ \dot{y}_r &=& v_r\sin\theta_r \\ \dot{\theta}_r &=& \omega_r \\ \dot{\varphi} &=& -\frac{v_r}{l_t}\sin\varphi \\ && -\frac{l_r\omega_r}{l_t}\cos\varphi - \omega_r \end{array}\right.$$

where the inputs v_r and ω_r are the linear and angular velocities of the robot. They are subjected to the following constraints: $|v_r| \leq v_{max}$, $|\omega_r| \leq \omega_{max}$, $|\dot{v}_r| \leq \dot{v}_{max}$ and $|\dot{\omega}_r| \leq \dot{\omega}_{max}$. These constraints and the weight of the robot ensures the absence of lateral slipping of the wheels. l_r and l_t are constants defining the geometry of the robot-trailer connection (Figure 1).

Let us notice that although System A is a particular case of System B with $l_r = 0$, both systems have different properties from a control point of view and they have to be studied separately.

The hardware architecture of our experiments (Figure 2) is composed of a Unix workstation and on-board processors, communicating via radio Ethernet. The software architecture is organized in three modules and an interface to control the execution during experiments [15]. The module `TRPLANNER` on the workstation computes a collision-free feasible path and sends this path to the module `TRPILO` on-board. This latter module first computes a time parameterization, and then samples the corresponding open-loop inputs $(v_r(t), \omega_r(t))$ on a segment of shared memory called *poster*. The module `TRLOCO` reads these data on the poster and computes in real-time the closed loop control of each motor. The position of the initial and final configurations are measured by an absolute localization system composed of three cameras mounted on the ceiling of the experimentation site.

III. Nonholonomic path planner

As described previously, this task is performed by the module TRPLANNER. The inputs to this module are a geometric map of the environment, initial and final configurations. The output is a collision-free feasible path between these configurations.

A. Motion Planner Based on a Steering Method

TRPLANNER is based on a two step approach, dealing separately with the physical constraints (obstacles) and with the kinematic constraints (rolling without slipping of the wheels). This approach formerly proposed in [26], first builds a collision-free path without taking into account the nonholonomic constraints of the system. Then, this path is approximated by a sequence of collision-free feasible sub-paths computed by a steering method. Finally, the resulting path is smoothed.

This approach is applicable to any small-time controllable system. The convergence is guaranteed as soon as the steering method satisfies the following topological property, first introduced in [44].

Let us denote by $\mathcal{C}$ the configuration space of the system. Let $d_{\mathcal{C}}$ be a distance in $\mathcal{C}$. A steering method is a function that maps any pair of configurations (q_0, q_1) to a continuous function $q(t)$ from $[0,1]$ to $\mathcal{C}$, such that $q(t)$ represents a feasible path[4] between q_0 and q_1.

Definition 1 Let Σ be a small-time controllable system. Let $\mathsf{Steer} : \mathcal{C} \times \mathcal{C} \to C^0([0,1], \mathcal{C})$ be a steering method for Σ. Steer verifies the *topological property* if

$$\forall \varepsilon > 0, \exists \eta > 0, d_{\mathcal{C}}(q_1, q_2) < \eta \Rightarrow$$
$$\forall t \in [0,1], d_{\mathcal{C}}(q_1, \mathsf{Steer}(q_1, q_2)(t)) < \varepsilon$$

A steering method verifying the topological property is said to be *TP-admissible.*

The property introduced in this definition is directly related to the small-time controllability of the system. It can be roughly summarized by: the closer to each other two configurations are, the closer to these configurations the path computed by the steering method has to remain. This property which may seem somehow technical and in relation with our approximation scheme, is in fact more general and required in other path planning schemes (see below). This property is critical for obstacle avoidance.

The four following sections respectively present the geometric planner, steering methods for both systems A and B, the approximation scheme and comments justifying our choices with respect to other approaches.

B. Geometric planner

The configuration space of our systems is $\mathbf{R}^2 \times (S^1)^2$. There is no method solving exactly the (holonomic) path planning problem in a reasonable amount of time for a system of dimension 4 (see [24]). For this reason we chose to use the random path planner (RPP) presented in [1]. This planner is *probabilistically complete*. That is, if a solution to a problem exists, the probability to find a solution tends toward 1 when the searching time increases toward infinity. RPP generates two types of paths: gradient paths to get closer to the goal and random walks to escape local minima. Figure 3 shows a path computed by this planner.

[4] Here the obstacles are ignored.

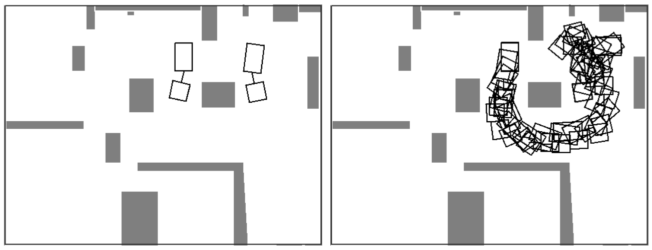


Fig. 3. A first collision-free path that does not take into account the nonholonomic constraints.

C. Steering method

1) Background: Even in the absence of obstacles, steering a nonholonomic system from a configuration to another one is not an easy task. According to the state of the art (e.g., [28], [19], [27]), steering exactly any nonholonomic system to a goal is an open problem. Fortunately both systems considered in this paper belong to special classes: they can be put into chained forms [52], [50], [53] and thus they are flat [37].

For chained form systems it is possible to devise steering methods by using either sinusoidal controls [52] or multirate controls [31]. The property of differential flatness of our systems makes the steering problem equivalent to the determination of a plane C^2 curve connecting two points with given tangent and curvature at the extremities. This latter problem can be solved easily using polynomials for instance [37].

Nevertheless none of these steering methods accounts for the small-time controllability of the considered systems. None of them is TP-admissible (see Definition III-A). Designing adequate steering methods from the existing one raises technical issues. In [44], we showed how to tune free parameters of the sinusoidal inputs to make the corresponding steering method TP-admissible. This has been the first solution. However, sinusoidal inputs revealed useless maneuvers for very simple problems (see Figure 5). The resulting collision free feasible path contains a lot of maneuvers, since this path is a concatenation of elementary paths generated by the steering method.

To palliate this bad behavior, we have devised a new steering method using the differential flatness of the system and based on natural curves like arcs of circles and straight lines. Each sub-path is composed at most of one cusp point and is very natural as seen in the experimental results. We present now this method in its one trailer version. An extended version for a n-trailer system can be found in [22]. We consider first the case of system A. The computations relative to System B are more complicated from a numerical point of view and will be exposed afterward.

2) Flatness-based TP-admissible steering method–System A: Let us consider the plane curve $\gamma(s)$ followed by the mid-point P of the trailer axle when the system is moving. The path of the whole system can be reconstructed from $\gamma(s)$. Indeed, the direction of the tangent to the curve is equal to the direction of the trailer, whereas the angle $\varphi(s)$ between the robot and the trailer is given by $\tan\varphi(s) = -\frac{\kappa(s)}{l_t}$, where $\kappa(s)$ is the curvature of $\gamma(s)$. The existence of a such construction constitutes the core of the flatness notion [37].

Let us consider two configurations $q_1 = (x_1, y_1, \theta_1, \varphi_1)$ and $q_2 = (x_2, y_2, \theta_2, \varphi_2)$. Let $\gamma_{q_1,q_2}(s)$ be a curve in $\mathbf{R}^2$

- starting at (x_1, y_1) with orientation θ_1 and curvature κ_1 for $s = 0$,
- arriving at (x_2, y_2) with orientation θ_2 and curvature κ_2 for $s = 1$, and such that
- $\tan\varphi_1 = -\frac{\kappa_1}{l_t}$ and $\tan\varphi_2 = -\frac{\kappa_2}{l_t}$

The family of curves $\gamma_{q_1,q_2}(s)$ constitute a TP-admissible steering method iff: $\forall\varepsilon > 0, \exists\eta > 0$,

$$\left\{\begin{array}{l} |x_2 - x_1| < \eta \\ |y_2 - y_1| < \eta \\ |\theta_2 - \theta_1| < \eta \\ |\kappa_2 - \kappa_1| < \eta \end{array}\right. \Rightarrow \forall s \in [0,1], \left\{\begin{array}{l} |x(s) - x_1| < \varepsilon \\ |y(s) - y_1| < \varepsilon \\ |\theta(s) - \theta_1| < \varepsilon \\ |\kappa(s) - \kappa_1| < \varepsilon \end{array}\right.$$

where $\gamma_{q_1,q_2}(s) = (x(s), y(s))$, $\theta(s)$ and $\kappa(s)$ are respectively the orientation of the tangent vector and the curvature of $\gamma_{q_1,q_2}(s)$ at s.

A lot of families of curves verifying various constraints have been studied in geometric modelling. Nevertheless, to our knowledge, none of them fits our requirement. Our solution is based on the perturbation of *canonical curves*. A canonical curve is associated to a configuration $q = (x, y, \theta, \varphi)$: this is the unique curve $\gamma_q(s)$ defined by a constant curvature κ verifying $\tan\varphi = -\frac{\kappa}{l_t}$, and passing through the point (x, y) with a tangential orientation of θ. The canonical curve associated to a configuration is thus an arc of circle (when $\varphi \neq 0$) or a straight line segment (when $\varphi = 0$).

Now, two configurations $q_1 = (x_1, y_1, \theta_1, \kappa_1)$ and $q_2 = (x_2, y_2, \theta_2, \kappa_2)$ being given, we define v the abscissa of the projection of (x_2, y_2) on $\gamma_{q_1}(s)$. Then using an increasing function α over $[0,1]$ verifying $\alpha(0) = 0$, $\alpha(1) = 1$ and $\alpha'(0) = \alpha'(1) = \alpha''(0) = \alpha''(1) = 0$, and combining the two canonical curves as follows:

$$\gamma_{q_1,q_2}(t) = \alpha(t)\gamma_{q_1}(vt) + (1 - \alpha(t))\gamma_{q_2}(v(t-1)) \qquad (1)$$

we obtain a C^2 curve going from q_1 to q_2. Indeed the two first derivatives of $\gamma_{q_1,q_2}(t)$ at 0 (resp. 1) are the same as those of $\gamma_{q_1}(vt)$ (resp. $\gamma_{q_2}(v(t-1))$). At this point, the family of curves γ_{q_1,q_2} defines a steering method denoted by $\mathsf{Steer}^*_{flat}(q_1, q_2)$.

The reparameterization of the canonical curves by $s = vt$ in equation (1) may seem confusing and useless, but this reparameterization is very important since it ensures that if q_2 is on the canonical curve of q_1, the curve $\gamma_{q_1,q_2}(t)$ remains on this canonical curve. By continuity, perturbing slightly q_2 around $\gamma_{q_1}(s)$ results in curves close to this canonical curve. This idea is the basis of our construction and enables

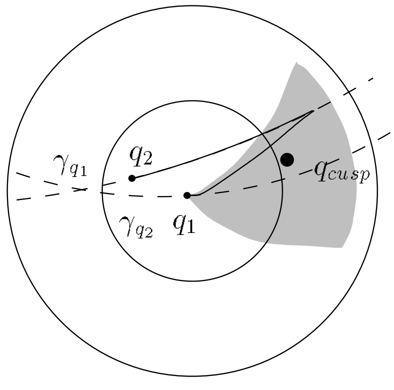


Fig. 4. A TP-admissible flat steering method.

us to define a cone around $\gamma_{q_1}(s)$, reachable without leaving a given ball centered on q_1 (shaded area in Figure 4), as proved in details in [22].

However this cone is not a neighborhood of the configuration q_1 and Steer^*_{flat} is not a TP-admissible steering method. To reach a neighborhood of q_1 without escaping a given ball centered on q_1, we need to introduce a cusp point where the robot changes its direction of motion. Indeed Steer^*_{flat} paths are always free of cusp points. Connecting the two following configurations $(0,0,0,0)$ and $(0,\varepsilon,0,0)$ by a C^2 curve without cusp point requires the direction of the tangent vector $\theta(s)$ along the curve to reach $\pm\pi/2$. Thus, when ε tends toward 0, the path between $(0,0,0,0)$ and $(0,\varepsilon,0,0)$ does not remain in a decreasing neighborhood of $(0,0,0,0)$. This instance is a counter-example of Definition III-A.

The second step of the construction is also based on continuity. If two configurations q_1 and q_2 are close to one another, so are their canonical curves. Then γ_{q_2} necessarily intersects the cone reachable from q_1 (see Figure 4. Then we define a configuration q_{cusp} in the intersection between γ_{q_2} and the previous cone and we decompose the motion into two parts:

- a forward motion along $\mathsf{Steer}^*_{flat}(q_1, q_{cusp})$ and
- a backward motion along γ_{q_2}.

Note that if q_2 is in the cone, then $q_{cusp} = q_2$. The second part of the motion is useless and the configurations are connected without maneuver.

The previous computation leads to a new steering method (denoted by Steer_{flat}). This steering method is TP-admissible and thus can be used in a collision-free planning scheme. The geometric construction above gives some intuition of this property. A detailed proof appears in [22].

Paths generated by this method are shown in Figure 5.

3) Flatness-based TP-admissible steering method–System B: System B is also differentially flat. The previous method can thus be applied to this system. However, the difficulty lies now in the change of variable $(x_r, y_r, \theta_r, \varphi) \leftrightarrow (x, y, \theta, \kappa)$. The flat output (x, y) is no more a fix point of the system as for System A. It is given by the following

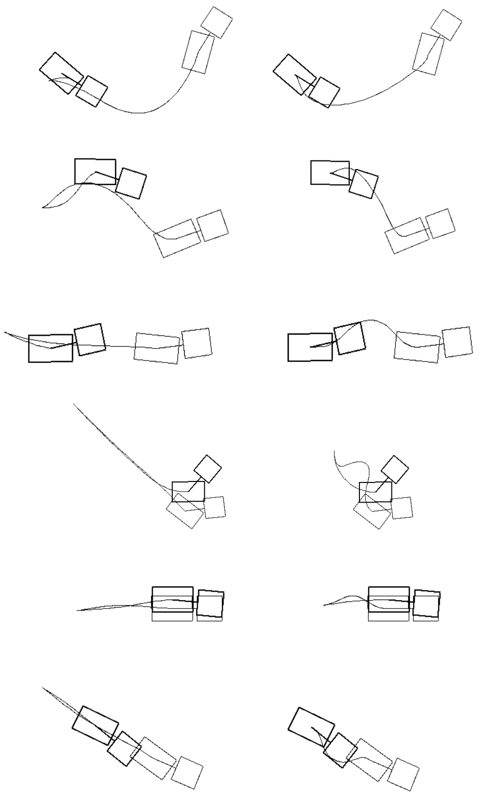


Fig. 5. Comparison between two steering methods on System A: sinusoidal inputs for the equivalent chained form system (left) and flatness-based steering method (right). Initial and final configurations are the same in each of the six cases. Note that Steer_{flat} generates more "natural" paths requiring fewer cusps and less space than the sinusoidal inputs.

expression from [36]:

$$\begin{cases} x = x_r - l_t\cos(\theta_r+\varphi) - L(\varphi)\frac{l_t\sin(\theta_r+\varphi)+l_r\sin(\theta_r)}{\sqrt{l_r^2+l_t^2+2l_rl_t\cos(\varphi)}} \\ y = y_r - l_t\sin(\theta_r+\varphi) + L(\varphi)\frac{l_t\cos(\theta_r+\varphi)+l_r\cos(\theta_r)}{\sqrt{l_r^2+l_t^2+2l_rl_t\cos(\varphi)}} \end{cases} \quad (2)$$

where $L(\varphi)=\int_0^\varphi \frac{\cos\sigma}{\sqrt{l_r^2+l_t^2+2l_rl_t\cos\sigma}}d\sigma$ is an elliptic function. The direction θ of the tangent to the curve $(x(t),y(t))$ and its curvature are given by the following formulas.

$$\tan\theta = \frac{l_r\sin\theta_r + l_t\sin(\theta_r+\varphi)}{l_r\cos\theta_r + l_t\cos(\theta_r+\varphi)} \quad (3)$$

$$\kappa = \frac{-\sin\varphi}{\cos\varphi\sqrt{l_r^2+l_t^2+2l_rl_t\cos(\varphi)}+L(\varphi)\sin\varphi} \quad (4)$$

These expressions and their inverses cannot be computed explicitly. We have to use numerical approximations. Thus, we sample functions $L(\varphi)$, $\kappa(\varphi)$ and $\frac{d\kappa}{d\varphi}(\varphi)$ in a precomputed array at the beginning of the initialization of the module **TRPLANNER**. Then, the values of $L(\varphi)$, $\kappa(\varphi)$ and its inverse $\varphi(\kappa)$ are computed by cubic interpolation.

Note. The steering method we have developed in this section can be used for any two input driftless flat system of dimension 4. In particular, chained form systems belong to this category.

D. Approximation scheme

Let us now describe our approximation scheme. This scheme is derived from the motion planner for car-like robots presented in [26]. This reference does not mention any notion of steering method admissibility. The seminal method uses optimal length feasible paths between two configurations. This strategy results automatically in a TP-admissible steering method because optimizing a cost function prevents the system to go far away from the configurations to connect.

Given a collision-free path not taking into account the nonholonomic constraints, this path is iteratively split into pieces the endpoints of which are linked by feasible paths computed with Steer_{flat}, as soon as they are collision-free. The resulting path is then randomly smoothed in a third step: two configurations are randomly chosen on the collision-free feasible path. If the local steering method succeeds in connecting these two configurations by a shorter collision-free path, the sequence of sub-paths previously connecting them is replaced by the new local path. This operation is repeated until the global number of sub-paths stops decreasing.

Let us notice that following a feasible path without stopping requires this path to be C^1 in $\mathcal{C}$. This condition is fulfilled by Steer_{flat} by constraining the third derivative of α: $\alpha^{(3)}(0)=\alpha^{(3)}(1)=0$.

Figure 6 (top) shows the application of the algorithm on the geometric path shown in Figure 3. The other examples show solutions in a highly constrained space.

E. Related work and discussion

A direct approach to motion planning for mobile robots with trailers is proposed in [2]. Piecewise constant inputs are used to explore a discretized model of the configuration space. The search is performed by Dijkstra's algorithm allowing to take into account optimality criteria such as the path length or the number of reversals. The algorithm is proved to be asymptotically complete. Nevertheless, the paths obtained by this method are not C^1 and the robot has to stop at each node of the graph. Augmenting the dimension of the system could be a way to obtain C^1 paths. However, the efficiency of this algorithm dramatically decreases with the dimension of the configuration space.

A method presented in [14] combines the two step approach above and a so-called variational approach. First, a collision-free path is generated. Then the nonholonomic constraints are introduced progressively. At each iteration, a path is generated from the previous one to satisfy more severe nonholonomic constraints. The search explores the neighborhood of the current path according to a dynamic programming procedure. The method is neither complete nor asymptotically complete. Completeness would require back-tracking, an expensive procedure. Nevertheless simulations have been performed with success for a mobile robot with three trailers and for two tractor-trailer robots sharing the same environment.

Another nonholonomic path planner for mobile robots with trailers has been proposed in [45]. It combines proba-

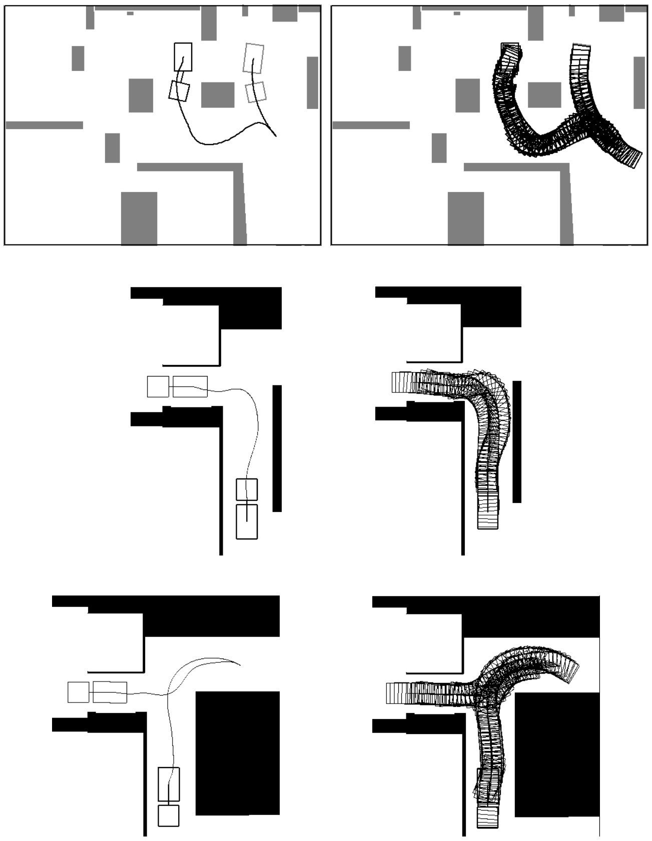


Fig. 6. Collision-free admissible paths. The paths on the left are the paths followed by the reference point of the robot. The pictures on the right show the corresponding volumes swept by both the robot and the trailer.

bilistic roadmaps and approximation scheme. The strategy consists in introducing the nonholonomic constraints one by one. The steering method uses sinusoidal inputs. The objective here is to face the complexity when the number of trailers increases. We have experienced the method for our practical cases: the quality of the paths was worse (due to the use of sinusoidal inputs) and the time of computation was higher.

A heuristic approach to obstacle avoidance for a car pulling several trailers with off-axle hitching system appears in [7].

IV. From Path to Trajectory

Once a collision-free feasible path between two configurations has been produced, it has to be parameterized by time in order to take into account the bounds on the velocities and accelerations of the robot. This task is performed on board by the module `TRPILO`. The input of this module is a collision-free feasible path $\{(x_r(s), y_r(s), \theta_r(s), \varphi(s)), s \in [s_{start}, s_{end}]\}$, v_{max}, $\dot{v}_{max}$, ω_{max} and $\dot{\omega}_{max}$. The output is a feasible trajectory:

$((x_r(s(t)), y_r(s(t)), \theta_r(s(t)), \varphi(s(t))), t \in [t_{start}, t_{end}])$,

that satisfies the input dynamical constraints. $s(t)$ is the time parameterization to be computed.

A. Related work and motivation

Integrating constraints on velocities and accelerations can be done at the planning level. This is the so-called kinodynamic motion planning problem [13], [34]. The methods are based on a discretization of the configuration space and require a perfect knowledge of the $\mathcal{C}$-obstacles. We did not explore this direction because of the computational cost of a search in the phase space of our system which is 6-dimensional.

Transforming a path into a trajectory is a classical problem in robotics. The minimal time parameterization of a given path has been mainly addressed for manipulators. Different methods have been proposed in this context [48], [4], [49] (see [35] for an overview). Application to mobile robots appears in [47]: the computation of a time-optimal motion along a path is used to evaluate the cost of this path. The objective is to compute optimal trajectories for a mobile robot moving on a terrain.

The problem is well understood: formal solutions exist. However the parameterization functions are most of the time described as solutions of differential equations. Their effective computation thus requires numerical integration and has to be dealt with carefully, mainly because of the two following points:

- The functions $\theta_r(s)$ and $\varphi(s)$ and their derivatives returned by the path planner of Section III present huge variations. For this reason, the exact integration of the time-optimal curves would require elaborate methods with adaptative time-step like Runge-Kutta for instance.
- The code and execution of this step is on-board where memory is a very critical component.

Moreover, the optimality of the time parameterization is not our first concern in this work. For this reason, we have chosen to adapt an existing method described in [49]. In the algorithm we propose, the trade-off between optimality and memory space is parameterized. Our contribution here is more practical than formal.

Our main idea here is to describe the time-parameterization $s(t)$ by piece-wise constant acceleration curves. The size of the constant acceleration intervals is automatically adapted. Our method results in less memory consuming data structures since we do not try to follow exactly the time optimal solution. Instead, we keep a constant acceleration as long as this acceleration remains in a suitable interval. One of our constant acceleration time interval usually includes several steps of a Runge-Kutta method.

In addition to the acceleration constraints, our method deals with bounds on the velocities of the robot. This point has not been taken into account in prior work about time optimal parameterization (it is mentioned in [47]).

B. Constraints in the phase plane $(s, \dot{s})$

In this section, we recall some key notions used in [4], [48], [49] and introduce some notation. Without loss of generality we consider now the case of a forward motion. By setting $d_v(s) = \sqrt{\frac{dx_r}{ds}(s)^2 + \frac{dy_r}{ds}(s)^2}$ and $d_\omega(s) = \frac{d\theta_r}{ds}(s)$, the velocities and acceleration have the following expressions:

$$\begin{aligned} v &= d_v(s)\dot{s} && (5)\\ \omega &= d_\omega(s)\dot{s} && (6)\\ \dot{v} &= d_v(s)\ddot{s} + \delta_v(s)\dot{s}^2 && (7)\\ \dot{\omega} &= d_\omega(s)\ddot{s} + \delta_\omega(s)\dot{s}^2 && (8) \end{aligned}$$

where

$$\delta_v(s) = \frac{d}{ds}d_v(s) \text{ and } \delta_\omega(s) = \frac{d}{ds}d_\omega(s).$$

Velocity Constraints. The velocity constraints $0 \leq v \leq v_{max}$ and $|\omega| \leq \omega_{max}$ are represented by a forbidden area in the *phase plane* $(s, \dot{s})$.

$$\dot{s} \leq \mathrm{Inf}\{\frac{v_{max}}{d_v(s)}, \frac{\omega_{max}}{|d_\omega(s)|}\}$$

We call *velocity saturation curve* the curve obtained when the previous inequality is an equality.

Acceleration constraints. From (7) and (8), $(s, \dot{s})$ being given in the phase plane, the acceleration constraints $|\dot{v}| \leq \dot{v}_{max}$ and $|\dot{\omega}| \leq \dot{\omega}_{max}$ impose $\ddot{s}$ to belong to the intersection of two intervals. We denote by $[\alpha(s,\dot{s}), \beta(s,\dot{s})]$ this intersection when it is not empty. An equivalent condition for this interval not to be empty is (after computations):

$$\dot{s}^2 \leq \frac{\dot{\omega}_{max}|d_v(s)| + \dot{v}_{max}|d_\omega(s)|}{|\Delta|} \quad (9)$$

with

$$\Delta = d_v(s)\delta_\omega(s) - d_\omega(s)\delta_v(s)$$

The curve corresponding to equality in (9) is called the *maximal velocity curve.* It is denoted by $g(s)$. The signification of this curve can be interpreted as follows. At any point on the path, if the velocity is too high, both acceleration constraints cannot be satisfied simultaneously. An example of this fact is the case of a car following a road composed of a straight line and a turn of increasing curvature. This situation corresponds to a coefficient δ_ω starting from 0 and increasing along the turn. If the speed of the car is too high, even by braking as much as possible, the angular acceleration cannot be made smaller than its maximal allowed value. This example illustrates the fact that finding a correct parameterization of a path is a global problem that requires knowledge of the path in the future.

C. Our algorithm

Before explaining our algorithm, we need to define the notion of *characteristic point* introduced in [48]. We define then what we call acceleration and deceleration curves.

Characteristic points. From the previous definition, the interval $[\alpha(s,\dot{s}), \beta(s,\dot{s})]$ is empty iff $\dot{s} > g(s)$. Moreover, $\alpha(s, g(s)) = \beta(s, g(s))$ if $d_\omega(s) \neq 0$. In our case, d_v never vanishes, it is a property of our local planner.

We define

$$\tau(s) = \frac{d\dot{s}}{ds} - \frac{dg}{ds}$$

the difference between the slope of the phase plane trajectory and the slope of the maximal velocity curve.

We say that $(s, \dot{s})$ is an *out-point* if $\tau(s) > 0$ and an *in-point* if $\tau(s) < 0$. With these notations, *characteristic points* are defined as points $(s, \dot{s})$ where $\tau(s^-) > 0$ and $\tau(s^+) < 0$. At these points and only at these points, a phase plane trajectory can meet the maximum velocity curve without violating the acceleration constraints.

Acceleration and deceleration curves. From now on we call $\dot{s}$ and $\ddot{s}$ pseudo-velocity and pseudo-acceleration. Let $\mu < 1/4$ be a positive real number. The key idea of an acceleration (resp. deceleration) curve is to define contiguous intervals of the s-axis where the pseudo-acceleration can be kept constant and in the upper (resp. lower) 2μ-portion of the interval $[\alpha(s,\dot{s}), \beta(s,\dot{s})]$. The size of the intervals is thus automatically adapted to the variation of the coefficients. This strategy enables us to gain memory space, losing optimality. Coefficient μ tunes the trade-off between memory space and optimality.

The $\ddot{s_0}$-*constant pseudo-acceleration curve* passing by $(s_0, \dot{s_0})$ is represented in the phase plane by a parabola:

$$\dot{s} = \Gamma_{(s_0,\dot{s_0},\ddot{s_0})}(s) = \sqrt{\dot{s_0}^2 + 2\ddot{s_0}(s - s_0)}$$

The acceleration curve starting from a point in the phase plane $(s_0, \dot{s_0})$ is defined by the following algorithm. Let $\ddot{s_i} = (1-\mu)\beta(s_i, \dot{s_i}) + \mu\alpha(s_i, \dot{s_i})$. We define

$$\begin{aligned} s_{i+1} &= \mathrm{Inf}\{s > s_i, \ddot{s_i} \notin [(1-2\mu)\beta(s, \Gamma_{(s_i,\dot{s_i},\ddot{s_i})}(s)) \\ &\quad + 2\mu\alpha(s, \Gamma_{(s_i,\dot{s_i},\ddot{s_i})}(s)), \beta(s, \Gamma_{(s_i,\dot{s_i},\ddot{s_i})}(s))]\} \\ \dot{s}_{i+1} &= \Gamma_{(s_i,\dot{s_i},\ddot{s_i})}(s_{i+1}) \end{aligned}$$

The acceleration curve starting from $(s_0, \dot{s}_0)$ is then defined by $\dot{s} = \Gamma_{(s_i,\dot{s}_i,\ddot{s}_i)}(s)$ over each interval $[s_i, s_{i+1}]$. Notice that when μ tends toward zero these curves tends to maximal velocity curves.

Deceleration curves are identically defined, replacing α by β.

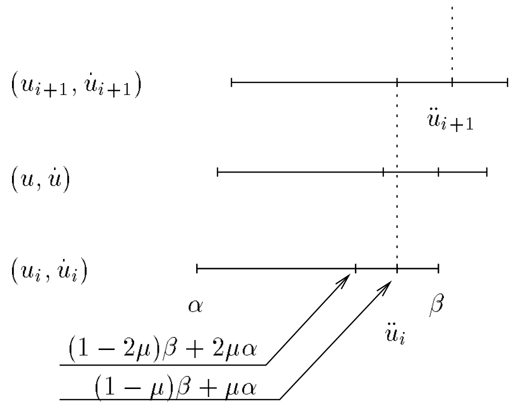


Fig. 7. Acceleration curve: $\ddot{s}_i$ must remain in the upper interval between s_i and s_{i+1}.

Algorithm. Starting from $(s_{start}, 0)$, we build an acceleration curve until an out-point is reached. Then we build a deceleration curve backward from the next characteristic point. If the deceleration curve intersects the acceleration curve, we start again from the characteristic point. If the deceleration curve reaches the area above the last acceleration curve, it is stopped and the acceleration curve is extended. Finally the last deceleration is built backward from $(s_{end}, 0)$ until it intersects the already built curve. For more details we refer to [20].

Figure 8 shows an example of phase curve taking into account only the acceleration constraints.

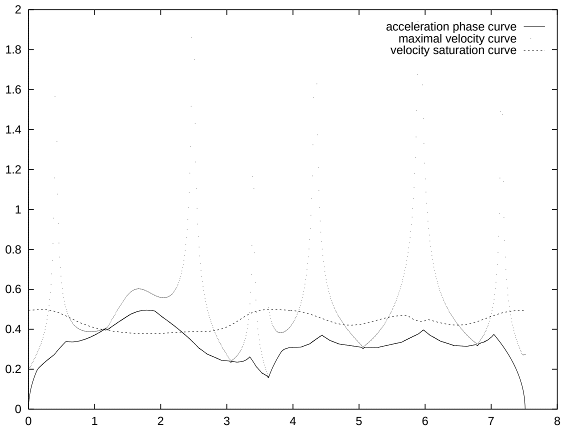


Fig. 8. Phase curve (in the plane $(s, \dot{s})$) taking into account the acceleration constraints.

The velocity constraints. From now on, we call the formerly built phase plane curve the *acceleration phase curve.* From this curve we are going to build another one which takes into account the velocity constraints of the robot.

The method consists in following the acceleration phase curve until a velocity constraint is violated. Then the velocity saturation curve is followed as long as its slope corresponds to a suitable acceleration, and the acceleration phase curve remains above the velocity saturation curve. Three events can then occur:

1. The slope of the velocity saturation curve becomes too big: an acceleration curve is built until it reaches the velocity saturation again or until it reaches the acceleration phase curve.
2. The slope of the velocity saturation curve becomes too small: from the next point on this curve such that the slope is again suitable, a deceleration curve is built backward.
3. The velocity saturation curve intersects the acceleration phase curve: it is followed until it intersects again the velocity saturation curve.

Figure 9 shows the final phase curve taking into account all the constraints.

V. Motion control

A. Motivation and related work

Motion control for nonholonomic systems have given rise to a lot of work. Brockett's condition [5] made stabilization

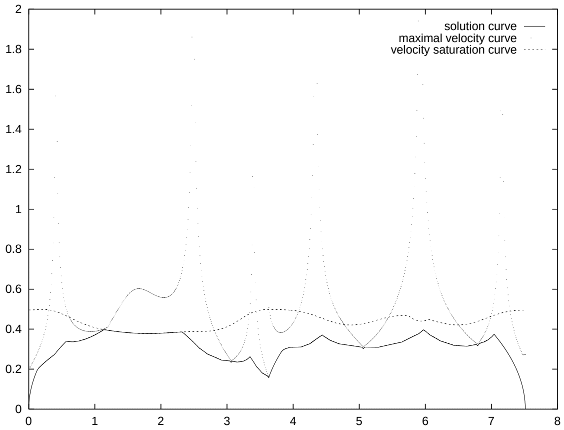


Fig. 9. Final phase curve in the plane $(s, \dot{s})$: in this example, only step 3 was encountered.

about a given configuration a challenging task for such systems, proving that it could not be performed by a simple continuous state feedback. Alternative solutions as time-varying feedback [32], [9], [33], [40], [43], [42], [51] or discontinuous feedback [8] have been then proposed. On the other hand, tracking a trajectory for a nonholonomic system does not meet Brockett's condition and thus is an easier task. A lot of work have also addressed this problem [11], [17], [18], [38], [41] for the particular case of mobile robots. See [10] for a recent survey in mobile robot motion control.

All these control laws work under the same assumption: the evolution of the system is exactly known and no perturbation makes the system deviate from its trajectory. Few papers dealing with mobile robots control take into account perturbations in the kinematic equations. [3] however proposed a method to stabilize a car about a configuration, robust to control vector fields perturbations, and based on iterative trajectory tracking. [29] proposed another iterative method robust to control error for chained form systems.

The presence of obstacles makes the task of reaching a configuration even more difficult. The approach we have implemented combines iteratively open loop controls together with closed loop controls. Such a strategy is analyzed by assuming that the execution of a given trajectory is subjected to perturbations. The model we chose for these perturbations is simple and general. The approach presents some common points with [3] and [29]. The main difference lies in the error model used.

B. Trajectory tracking

Mobile robot without trailer. The low velocity (50 cm/s) of Hilare's motions, the good quality of its locomotion system and the good quality of the planned trajectories make the trajectory tracking task non critical. We devised a simple control law enabling us to reuse the controller of our robot without trailer [23] (this is its only advantage with respect to the various existing approaches). This controller directly derived from [41]. Let (x, y, θ) be the coordinates of the reference robot in the frame of the real robot. Let (v_r^0, ω_r^0) be the inputs of the reference trajectory. The control law has the following expression:

$$\left\{ \begin{array}{lcl} v_r & = & v_r^0 \cos\theta + k_1 x \\ \omega_r & = & \omega_r^0 + k_3\theta + k_2 \frac{\sin\theta}{\theta} y \end{array} \right. \quad (10)$$

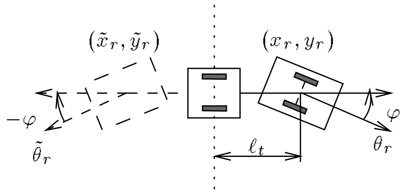


Fig. 10. Virtual robot

In the following paragraph, we show how to extend this control law to a robot with trailer.

System A. The idea of our controller is the following. When the robot goes forward, the trailer is not taken into account and we stabilize the robot according to the simple control law above. When the robot goes backward, we define a virtual robot, symmetrical to the real robot with respect to the wheel axle of the trailer (Figure 10). The configuration of the virtual robot with respect to the real one is given by:

$$\left\{ \begin{array}{l} \tilde{x_r} = x_r - 2\ell\cos\theta_t \\ \tilde{y_r} = y_r - 2\ell\sin\theta_t \\ \tilde{\theta_r} = \theta_t - \varphi + \pi \end{array} \right.$$

If $(\tilde{v}, \tilde{\omega})$ are the linear and angular velocities of the virtual robot, we get : $\tilde{v} = -v, \tilde{\omega} = \frac{2v}{\ell}\sin(\varphi) - \omega$. Thus the virtual robot goes forward and virtually pulls the trailer. We apply therefore the control law (10) to the virtual robot $(\tilde{x_r}, \tilde{y_r}, \tilde{\theta_r})$.

System B. When the trailer is hitched behind the robot, the former construction is even more simple: we can replace the virtual robot by the trailer (a similar idea appears in [39]). In this case indeed, the velocities of the robot (v_r, ω_r) and of the trailer (v_t, ω_t) are connected by a one-to-one mapping. The configuration of the virtual robot is then given by the following system:

$$\left\{ \begin{array}{lcl} \tilde{x}_r & = & x_r - l_r\cos\theta_r - l_t\cos(\theta_r + \varphi) \\ \tilde{y}_r & = & y_r - l_r\sin\theta_r - l_t\sin(\theta_r + \varphi) \\ \tilde{\theta}_r & = & \theta_r + \varphi + \pi \end{array} \right.$$

Stability of the trailer. Do the previous approaches make the motion of the trailer truly stable ? To answer the question we consider here a forward motion for System A. The analysis of backward motions is equivalent by considering the virtual robot transformation. Moreover the following analysis may be applied as well to System B by considering the motion of the hitching point.

Let us denote by $(x_r^0, y_r^0, \theta_r^0, \varphi^0, v_r^0, \omega_r^0)$ a reference trajectory and by $(x_r, y_r, \theta_r, \varphi, v_r, \omega_r)$ the real motion of the system. We assume that the robot follows exactly its reference trajectory: $(x_r, y_r, \theta_r, v_r, \omega_r) = (x_r^0, y_r^0, \theta_r^0, v_r^0, \omega_r^0)$ and

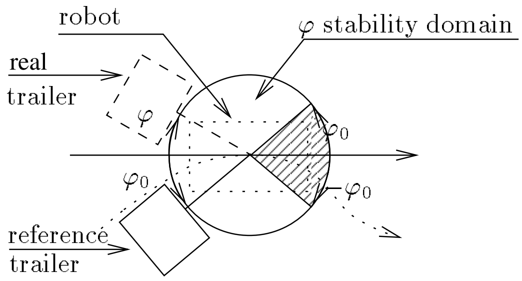


Fig. 11. Stability domain for $\hat{\varphi}$

we focus our attention on the trailer deviation $\hat{\varphi} = \varphi - \varphi^0$. The evolution of this deviation is easily deduced from the equation of System A ($l_r = 0$):

$$\begin{array}{lcl} \dot{\hat{\varphi}} & = & -\frac{v_r}{l_t}(\sin\varphi - \sin\varphi^0) \\ & = & -\frac{2v_r}{\ell_t}\cos(\frac{\varphi+\varphi^0}{2})\sin(\frac{\hat{\varphi}}{2}) \end{array}$$

$|\hat{\varphi}|$ thus decreases iff

$$-\frac{\pi}{2} < \varphi^0 + \frac{\hat{\varphi}}{2} < \frac{\pi}{2} \qquad [2\pi] \quad (11)$$

Our system is moreover constrained by the inequalities

$$-\pi/2 < \varphi, \varphi^0 < \pi/2 \quad (12)$$

so that $-\pi < \hat{\varphi} < \pi$ and (11) is equivalent to

$$\left\{ \begin{array}{ccc} 0 < \varphi^0 < \frac{\pi}{2} & \text{and} & -\pi < \hat{\varphi} < \pi - 2\varphi^0 \\ & \text{or} & \\ -\frac{\pi}{2} < \varphi^0 < 0 & \text{and} & -\pi - 2\varphi^0 < \hat{\varphi} < \pi \end{array} \right. \quad (13)$$

Figure 11 shows the domain on which $|\hat{\varphi}|$ is decreasing for a given value of φ^0. This domain is always a neighborhood of 0. Moreover, the previous computations permit easily to show that 0 is an asymptotically stable value for the variable $\hat{\varphi}$.

Thus if the real or virtual robot follows its reference forward trajectory, the trailer is stable and will converge toward its own reference trajectory.

C. Iterative scheme and robustness

Once the robot stops after tracking a planned trajectory, the gap to the real goal is computed. If this gap is greater than some threshold, then a new trajectory is planned and tracked. As we will see below this simple iterative scheme gives very good results. Usually, no more than one maneuver is needed to improve the final position of the system. Before presenting the experimental results, let us analyze the robustness of this control scheme from a formal point of view.

For this, we need to have a model of the perturbations arising when the robot moves. In our experiment we observed random perturbations due for instance to some play in the hitching system. These perturbations are very difficult to model. For this reason, we make only two simple hypotheses:

$$\begin{array}{rcl} d_{\mathcal{C}}(q(s), q^0(s)) & \leq & \delta s \\ d_{\mathcal{C}}(q(s), q^0(s)) & \leq & \Delta \end{array}$$

where s is the curvilinear abscissa along the planned path, q and q^0 are respectively the real and reference configurations, $d_{\mathcal{C}}$ is a distance over the configuration space of the system and δ, Δ are positive constants. The first inequality means that the distance between the real and the reference configurations is proportional to the length of the motion already executed along the planned path. The second inequality is ensured by the trajectory tracking control law that prevents the system to go too far away from its reference trajectory. Let us point out that these hypotheses are very realistic and fit a lot of perturbation models.

We need now to know the length of the paths generated at each iteration. We have seen that the steering method we use to compute these paths is TP-admissible. This means that if the goal is sufficiently close to the starting configuration, the computed trajectory remains in a neighborhood of the starting configuration[5]. In [21] we compute an estimate in terms of distance: if q_1 and q_2 are two sufficiently close configurations, the length $\ell(q_1, q_2)$ of $\mathsf{Steer}_{flat}(q_1, q_2)$ verifies

$$\ell(q_1, q_2) < \eta d_{\mathcal{C}}(q_1, q_2)^{\frac{1}{4}}$$

where η is a positive constant.

Thus, if $(q_i)_{i=1,2,\ldots}$ is the sequence of configurations reached after i motions, we have the following inequalities:

$$\begin{array}{rcl} d_{\mathcal{C}}(q_1, q_{goal}) & \leq & \Delta \\ d_{\mathcal{C}}(q_{i+1}, q_{goal}) & \leq & \delta \ell(x_i, x_{goal}) \\ & \leq & \delta \eta d_{\mathcal{C}}(q_i, q_{goal})^{\frac{1}{4}} \end{array}$$

These inequalities ensure that $d_{\mathcal{C}}(q_i, q_{goal})$ is upper bounded by a sequence $(d_i)_{i=1,2,\ldots}$ of positive numbers defined by

$$\begin{array}{rcl} d_1 & = & \Delta \\ d_{i+1} & = & \delta \eta d_i^{\frac{1}{4}} \end{array}$$

and converging toward $(\delta\eta)^{\frac{4}{3}}$.

Thus, our iterative method does not converge exactly toward the goal, but ensures the existence of a stable domain of convergence around the goal. This result essentially comes from the very general model of perturbations we have chosen.

The experimental results of the following section show however, that the converging domain of our control scheme is very small.

VI. Experiments

We present three experiments for each system. The geometric map of the environment covers 170 m^2. The bitmap representation of the environment is a grid of 150000 pixels. The geometric parameters of System A are $l_r = 0\ cm$ and $l_t = 120\ cm$. Those of System B are $l_r = 65\ cm$ and $l_t = 90\ cm$. For both systems, the bounds on the velocities and accelerations are $v_{max} = .5\ ms^{-1}$, $\omega_{max} = .5\ rads^{-1}$, $\dot{v}_{max} = .5\ ms^{-2}$ and $\dot{\omega}_{max} = 1.8\ rads^{-2}$.

For each experiment, we proceed as follows. We localize the initial position of the robot using the cameras on the ceiling. Then we specify a goal configuration via the interface. After computations, the motion is executed. The position of the robot is updated by the dead-reckoning system combining the odometer of Hilare-2-bis and the angular encoder of the trailer. If the reached configuration is too far from the goal, we reexecute the same process. Figures 12 and 13 display the paths computed and give the precision reached after the first and second motions, with respect to the dead-reckoning localization. The times of computation correspond to the first path planning task on a Sun Sparc Ultra. The time parameterization is very fast ($<$ 1 sec). Let us point out that the second planning task is almost instantaneous because both configurations are very close to one another and only one call to the local planner is generally enough. The exact position of the robot cannot be measured exactly after each motion because the robot is not always under one of the cameras. However, the experimental results reported in Figure 12 and 13 are representative of the efficiency of the motion control task since the feedback law uses dead-reckoning data. Moreover, for paths such as those we executed in these experiments, the drift of the dead-reckoning system is less than 5 cm which represents a very good precision. We give the accuracy of the reached configuration only at the end of the motion because we experienced that the error during the motion increases at the beginning of the motion and then remains stable. Thus, values at the end are a good estimate of the precision during the whole motion.

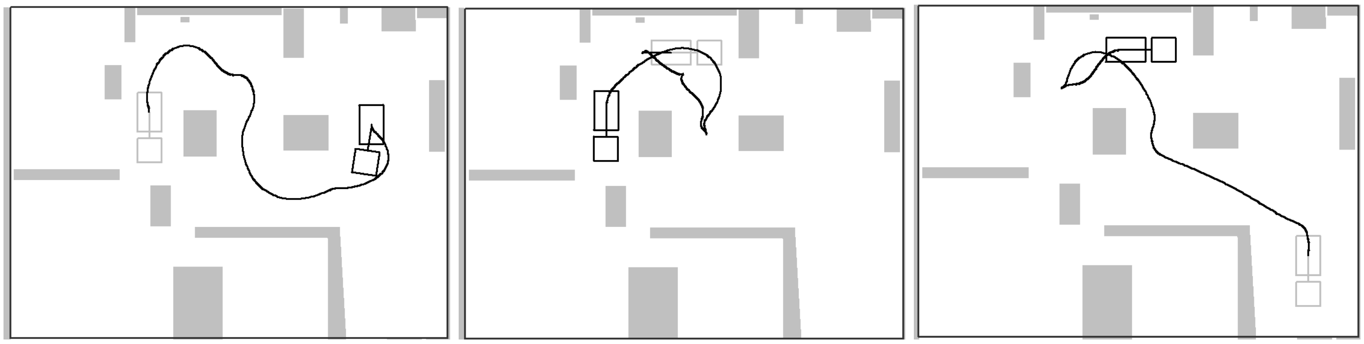
Figure 12 and 13 gather the results for System A and System B respectively[6]. Times correspond to the total time of the planning phase and the transformation of the paths into the trajectories to be executed. The good accuracy is mainly due to both the performance of the locomotion system of Hilare and the "smoothness" quality of the planned trajectory.

VII. Conclusion

We have presented in this paper an integrated approach to path planning and motion execution for a mobile robot with a trailer in a constrained environment. The critical point of this work, the path planning step, has been solved without any approximation on the robot geometry and kinematics. Our nonholonomic motion planner inherits the probabilistic completeness from the geometric planner RPP. This means that we may face any constrained environment. As an example, Figure 6 shows maneuver executed by our experimental system in a narrow corridor.

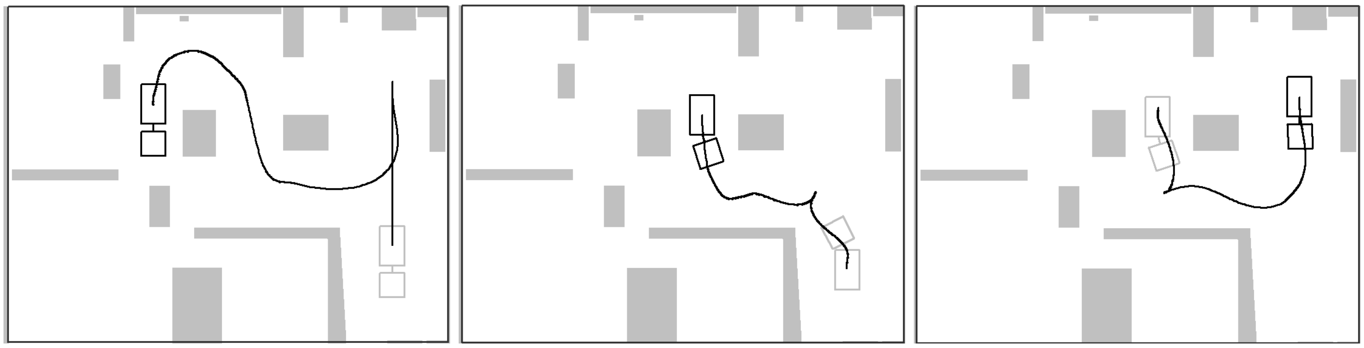
[5] This property is related to the small-time controllability of our system.

[6] See also a movie of an experiment at http://www.laas.fr/~jpl/movies/trailer.mov



	q_{init}	q_{goal}	Δq_1	Δq_2	
I	$x = 9.42$ $y = -2.90$ $\theta = 91.39$ $\varphi = -10.19$	$x = 2.04$ $y = -2.5$ $\theta = 90$ $\varphi = 0$	$\Delta x = 0$ $\Delta y = -0.02$ $\Delta\theta = 0.18$ $\Delta\varphi = 0$		18 sec 0 cusp
II	$x = 2.10$ $y = -2.43$ $\theta = 89.7$ $\varphi = 0$	$x = 4.26$ $y = -0.50$ $\theta = 180$ $\varphi = 0$	$\Delta x = 0$ $\Delta y = 0.14$ $\Delta\theta = 4.13$ $\Delta\varphi = -4.93$	$\Delta x = 0.01$ $\Delta y = -0.02$ $\Delta\theta = 1.96$ $\Delta\varphi = -0.71$	1 min 53 sec 3 cusps
III	$x = 4.27$ $y = -0.51$ $\theta = 179.63$ $\varphi = 0.35$	$y = 10.30$ $y = -7.32$ $\theta = 90$ $\varphi = 0$	$\Delta x = 0.13$ $\Delta y = 0.01$ $\Delta\theta = 4.45$ $\Delta\varphi = -4.93$	$\Delta x = -0.02$ $\Delta y = -0.01$ $\Delta\theta = 0.96$ $\Delta\varphi = -0.35$	23 sec 1 cusp

Fig. 12. Three experiments I, II and III from left to right, for system A. The initial configuration is in black, the final configuration is in grey. Notice that in Experiment I, the system backs up over a quite long distance (10 meters).



	q_{init}	q_{goal}	Δq_1	Δq_2	
I	$x = 2.20$ $y = -2.24$ $\theta = 90$ $\varphi = 0$	$x = 10.10$ $y = -6.90$ $\theta = 90$ $\varphi = 0$	$\Delta x = -0.05$ $\Delta y = 0.02$ $\Delta\theta = -1.10$ $\Delta\varphi = 1.41$	$\Delta x = 0$ $\Delta y = -0.01$ $\Delta\theta = -1.90$ $\Delta\varphi = 1.05$	1 min 10 sec 1 cusp
II	$x = 5.31$ $y = -2.61$ $\theta = 89.44$ $\varphi = 19.01$	$x = 10.08$ $y = -7.70$ $\theta = -90$ $\varphi = 27.65$	$\Delta x = 0.02$ $\Delta y = 0.03$ $\Delta\theta = 0.81$ $\Delta\varphi = -0.19$		40 sec 1 cusp
III	$x = 9.99$ $y = -2.00$ $\theta = 90.20$ $\varphi = 1.41$	$x = 5.34$ $y = -2.66$ $\theta = 90$ $\varphi = 18.43$	$\Delta x = 0$ $\Delta y = -0.06$ $\Delta\theta = 0.54$ $\Delta\varphi = 1.28$	$\Delta x = 0$ $\Delta y = -0.01$ $\Delta\theta = -0.75$ $\Delta\varphi = 0.58$	1 min 12 sec 1 cusp

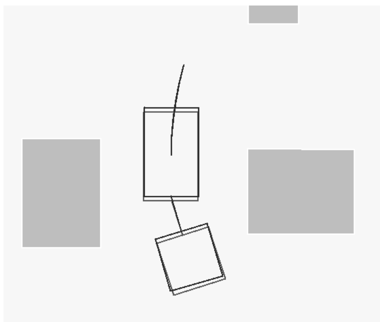


Fig. 13. Three experiments I, II and III from left to right, for system B. The initial configuration is in black, the final configuration is in grey. The zoom shows the final maneuver of the experiment III.

Moreover, the paths computed by our planner are very reasonable in terms of number of maneuvers and time of computation. This latter property is important from an experimental point of view and is mainly a consequence of the steering method we have proposed. Eventually, our approach has been proved very realistic in environments of the size of our experimentation room.

Acknowledgment

We thank G. Bauzil, S. Fleury, M. Herbb and M. Khatib for their help in integrating both the motion planner and the control law within the global architecture of Hilare-2-bis. The trailer of Hilare has been designed by A. Ferrand. Florent Lamiraux has been partially supported by Alcatel-Telecom.

Authors' e-mail addresses: lamiraux@cs.rice.edu, Sepanta.Sekhavat@inrialpes.fr, jpl@laas.fr.

References

[1] J. Barraquand and J.-C. Latombe, "Robot Motion Planning : a Distributed Representation Approach", *International Journal of Robotics Research*, Vol. 10 No. 6, 1991.

[2] J. Barraquand and J.-C. Latombe, "Nonholonomic Multibody Mobile Robots: Controllability and Motion Planning in the Presence of Obstacles", *Algorithmica*, Springer Verlag New York Inc., Vol. 10, pp. 121–155, 1993

[3] M. Bennani and P. Rouchon, "Robust Stabilization of Flat and Chained Systems", *European Control Conference*, 1995.

[4] J. Bobrow, S. Dubowski and J. Gibson, "Time-Optimal Control of Robotic Manipulator along Prespecified Paths", *International Journal of Robotics Research*, Vol. 4 No. 3, 1985.

[5] R. Brockett, "Asymptotic Stability and Feedback Stabilization", *Differential Geometric Control Theory*, R.W. Brockett, R.S. Millman and H.H. Sussmann Eds, 1983.

[6] F. Bullo and R. Murray, "Experimental Comparison of Trajectory Trackers for a Car with Trailers", in *IFAC World Congress*, San Francisco, 1996.

[7] L. Bushnell, "An Obstacle Avoidance Algorithm for a Car Pulling Trailers with Off-Axle Hitching," *IEEE International Conference on Decision and Control*, New Orleans, 1995.

[8] C. Canudas de Wit, O. Sørdalen, "Exponential Stabilization of Mobile Robots with Nonholonomic Constraints", *IEEE Transactions on Automatic Control*, Vol. 37, No. 11, 1992.

[9] J.-M. Coron, "Global Asymptotic Stabilization for Controllable Systems without Drift", *Mathematics of Control, Signals and Systems*, Vol. 5, 1992.

[10] A. De Luca, G. Oriolo and C. Samson, "Feedback Control of a Nonholonomic Car-Like Robot", *Robot motion planning and control*, J.-P. Laumond Ed., Lecture Notes in Control and Information Sciences, Springer Verlag, 229, 1998.

[11] R. DeSantis, "Path Tracking for a Tractor-Trailer-Like Robot", *International Journal of Robotics Research*, Vol. 13 No. 6, pp. 533–544, 1994.

[12] A. Divelbiss and J. Wen, "Trajectory Tracking Control of a Car-Trailer System", *IEEE Transactions on Control Systems Technology*, Vol. 5 No. 3, 1997.

[13] B. Donald, P. Xavier, J. Canny and J. Reif, "Kinodynamic Motion Planning," in *Journal of the ACM*, 40, pp. 1048–1066, 1993.

[14] P. Ferbach, "A Method of Progressive Constraints for Nonholonomic Motion Planning", *IEEE International Conference on Robotics and Automation*, pp. 2929–2955, Minneapolis, 1996.

[15] S. Fleury, M. Herrb and R. Chatila, "Design of a Modular Architecture for Autonomous Robot", *IEEE International Conference on Robotics and Automation*, San Diego, 1994.

[16] G. Giralt, R. Chatila, and M. Vaisset, "An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots", *Robotics Research : The First International Symposium* (M. Brady and R. P. Paul, eds.), MIT Press, pp. 191–214, 1984.

[17] A. Hemami, M. Mehrabi and R. Cheng, "Synthesis of an Optimal Control Law for Path Tracking in Mobile Robots", *Automatica*, 28(2), pp. 383–387, 1992.

[18] Y. Kanayama, Y. Kimura, F. Miyazaki and T. Nogushi, "A Stable Tracking Control Method for an Autonomous Mobile Robot", *IEEE International Conference on Robotics and Automation*, Cincinnati, pp. 384–389, 1990.

[19] I. Kolmanovski and N. McClamroch, "Developments in Nonholonomic Control Problems", *IEEE Control Systems*, Vol. 15 No. 6, 1995.

[20] F. Lamiraux and J.-P. Laumond, "From Paths to Trajectories for Multibody Mobile Robots", *International Symposium on Experimental Robotics*, Barcelona, 1997.

[21] F. Lamiraux, "Robots mobiles à remorque : de la planification de chemins à l'exécution de mouvements", PhD Thesis INPT97327, LAAS-CNRS, Toulouse, September 1997.

[22] F. Lamiraux and J.-P. Laumond, "Flatness and Small-Time Controllability of Multibody Mobile Robots: Application to Motion Planning", *European Control Conference*, Brussels, 1997.

[23] F. Lamiraux and J.-P. Laumond, "A Practical Approach to Feedback Control for a Mobile Robot with Trailer", *IEEE International Conference on Robotics and Automation*, Brussels, 1998.

[24] J.-C. Latombe, *Robot Motion Planning,* Kluwer Academic Publishers, 1991.

[25] J.-P. Laumond, "Controllability of a Multibody Mobile Robot", *IEEE Transactions on Robotics and Automation*, Vol. 9 No. 6, 1993.

[26] J.-P. Laumond, P. Jacobs, M. Taïx and R. Murray, "A Motion Planner for Nonholonomic Mobile Robot", *IEEE Transactions on Robotics and Automation*, Vol. 10 No. 5, 1994.

[27] J.-P. Laumond Ed, *Robot Motion Planning and Control*, Lecture Notes in Control and Informations Sciences, 229, Springer Verlag, 1998.

[28] Z. Li and J. Canny Eds, *Nonholonomic Motion Planning*, The Kluwer International Series in Engineering and Computer Science 192, 1992.

[29] P. Lucibello and G. Oriolo, "Stabilization via Iterative State Steering with Application to Chained-Form Systems", *35th IEEE International Conference on Decision and Control*, Kobe, Japan, pp. 2614–2619, 1996.

[30] M. Viale, T. Tsubochi and S. Yuta, "A Practical Path and Motion Planner for a Tractor-Trailer Robot," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Grenoble, 1997.

[31] S. Monaco and D. Normand-Cyrot, "An Introduction to Motion Planning under Multirate Digital Control," *IEEE International Conference on Decision and Control*, pp. 1780–1785, Tucson, 1992.

[32] P. Morin and C. Samson, "Application of Backstepping Techniques to the Time-Varying Exponential Stabilization of Chained Form Systems", *European Journal of Control*, Vol 3, No 1, 1997.

[33] J.-B. Pomet, "Explicit Design of Time-Varying Stabilizing Control Laws for a Class of Controllable Systems without Drift", *Systems and Control Letters*, North Holland, Vol 18, pp. 147–158, 1992.

[34] J. Reif and H. Wang, "Non-Uniform Discretization Approximations for Kinodynamic Motion Planning and its Applications", in *Algorithms for Robotic Motion and Manipulation*, J.-P. Laumond and M. Overmars Eds, A.K. Peters, pp. 97–112, 1997.

[35] M. Renaud and J.-Y. Fourquet, "Time-optimal Motions of Robot Manipulators Including Dynamics," *The Robotics Review 2*, O. Khatib, J.J. Craig and T. Lozano-Pérez Eds, MIT Press, 1992.

[36] P. Rouchon, M. Fliess, J. Lévine and P. Martin, "Flatness, motion planning and trailer systems", *IEEE International Conference on Decision and Control*, San Antonio, TX, pp. 2700–2705, 1993.

[37] P. Rouchon, M. Fliess, J. Lévine and P. Martin, "Flatness and Motion Planning : the Car with n Trailer", *European Control Conference*, Groningen, pp. 1518–1522, 1993.

[38] M. Sampei, T. Tamura, T. Kobayashi and M. Shibui, "Arbitrary Path Tracking Control of Articulated Vehicles Using Nonlinear Control Theory", *IEEE Transactions on Control Systems Technology*, Vol. 3 N0. 1, pp. 125–131, 1995.

[39] M. Sampei and T. Kobayashi, "Path Tracking Control of Car-Caravan Type Articulated Vehicles Using Nonlinear Control Theory, *Transactions of SICE*, Vol. 30 No. 4, pp. 427–434, 1994 (in Japanese).

[40] C. Samson, "Velocity and Torque Feedback Control of a Nonholonomic Cart," *International Workshop in Adaptative and Nonlinear Control: Issues in Robotics*, Grenoble, France, 1990.

[41] C. Samson and K. Ait-Abderrahim, “Feedback Control of a Nonholonomic Wheeled Cart in Cartesian Space”, *IEEE International Conference on robotics and Automation*, pp. 1136–1141, Sacramento, 1991.
[42] C. Samson, “Control of Chained Systems: Application to Path Following and Time-Varying Point-Stabilization of Mobile Robots”, *IEEE Transaction on Automatic Control*, Vol. 40 No. 1, 1995.
[43] C. Samson, “Time-Varying Feedback Stabilization of Car-Like Wheeled Mobile Robots”, *International Journal of Robotics Research*, 12(1), 1993.
[44] S. Sekhavat, J.-P. Laumond, “Topological Property of Trajectories Computed from Sinusoidal Inputs for Nonholonomic Chained Form Systems”, *IEEE International Conference on robotics and Automation*, Minneapolis, 1996.
[45] S. Sekhavat, P. Švestka, J.-P. Laumond and M. Overmars, ”Multi-Level Path Planning for Nonholonomic Robots Using Semi-Holonomic Subsystems”, in *Algorithms for Robotic Motion and Manipulation*, J.-P. Laumond and M. Overmars Eds, A.K. Peters, pp. 79–96, 1997 (also in *International Journal of Robotics Research*, Vol. 17 No. 8, pp. 840–857, 1998).
[46] S. Sekhavat, F. Lamiraux, J.-P. Laumond, G. Bauzil and A. Ferrand, ”Motion Planning and Control for Hilare Pulling a Trailer: Experimental issues”, *IEEE International Conference on robotics and Automation*, Albuquerque, 1997.
[47] Z. Shiller and Y. Gwo, ”Dynamic Motion Planning of Autonomous Vehicles”, *IEEE Transactions on Robotics and Automation*, Vol. 7 No. 2, pp. 241–249, 1991.
[48] K. Shin and N. Mc Kay, “Minimum-Time Control of a Robotic Manipulator with Geometric Path Constraints”, *IEEE Transactions on Automatic Control*, Vol. 30 No. 6, 1985.
[49] J.-J. Slotine and H. Yang, “Improving the Efficiency of Time-Optimal Path-Following Algorithms”, *IEEE Transactions on Robotics and Automation*, Vol. 5 No. 1, pp. 118–124, 1989.
[50] O. Sørdalen, “Conversion of a Car with n Trailers into a Chained Form,” *IEEE International Conference on Robotics and Automation*, pp. 382–387, Atlanta, 1993.
[51] O. Sørdalen and O. Egeland, “Exponential Stabilization of Nonholonomic Chained Systems,” *IEEE Transactions on Automatic Control*, Vol 40, No 1, 1995.
[52] D. Tilbury, R. Murray and S. Sastry, “Trajectory Generation for the n-Trailer Problem Using Goursat Normal Form,” in *IEEE Transactions on Automatic Control*, Vol. 40 No. 5, pp. 802–819, 1995.
[53] D. Tilbury, “Exterior Differential Systems and Nonholonomic Motion Planning”, Memorandum # UCB/ERL M94/90, Electronics Research Laboratory, *University of California Berkeley*, 1994.

Florent Lamiraux graduated from the Ecole Polytechnique Paris in 1993. He received the Ph.D. degree in Computer Science from the Institut National Polytechnique de Toulouse in 1997 for his research in Mobile Robots at LAAS-CNRS in Toulouse. He is currently a postdoctoral research associate in the department of computer science at Rice University in Houston. His research interests include motion planning and control in Robotics, and assembly planning.

Sepanta Sekhavat graduated from ENSEEIHT, Toulouse, in 1993. He received the Ph.D. degree in Electrical Engineering from the Institut National Polytechnique de Toulouse in 1997. He was a Postdoctoral Researcher at the Department of Electrical Engineering and Computer Sciences of the University of California at Berkeley, in 1997. He is currently Chargé de Recherche at INRIA, Grenoble, France. His research interests include nonholonomic motion planning and multi-agent cooperation.

Dr Sekhavat received the Léopold Escand Award of INPT for his Ph.D. thesis.

Jean-Paul Laumond is Directeur de Recherche at LAAS-CNRS in Toulouse, France. He received the M.S. degree in Mathematics, the Ph.D. in Robotics and the Habilitation from the University Paul Sabatier at Toulouse in 1976, 1984 and 1989 respectively. From 1976 to 1983 he was teacher in Mathematics. He joined CNRS in 1985 as Chargé de Recherche. In Fall 1990 he has been invited senior scientist from Stanford University. He has been a member of the Comité National de la Recherche Scientifique from 1991 to 1995. He is Associated Editor of the IEEE Transactions on Robotics and Automation and member of the steering committee of WAFR.

His research interests include mainly robotics and algorithmic motion planning. From 1992 to 1995 he has been coordinator of the European Esprit 3 Basic Research project 6546 PROMotion (Planning RObot Motion). He is currently coordinator of the European Esprit 4 LTR project 28226 MOLOG (Motion for Logistics, 1999 – 2002).