Accepted for publication in the journal "Machine Intelligence & Robotic Control" (MIROC)
oct 2004

Autonomous Navigation of a Bi-steerable Car : Experimental Issues

J. Hermosillo C. Pradalier S. Sekhavat C. Laugier
INRIA Rhone-Alpes
655 av. de l'Europe, Montbonnot, 38334 St. Ismier Cedex, France

Abstract

The recent development of a new kind of public transportation system relies on a particular double-steering kinematic structure enhancing maneuverability in cluttered environments such as downtown areas. We call bi-steerable car *a vehicle showing this kind of kinematics. We present experimental results, from map-building to trajectory tracking, aiming at validating the theoretical considerations obtained recently for the general bi-steerable system. These results are a first step towards the motion autonomy of this kind of transportation system.*

1 Introduction

The development of new Intelligent Transportation Systems (*ITS*), more practical, safe and accounting for environmental concerns, is a technological issue of highly urbanized societies today [10]. One of the long run objectives is to reduce the use of the private automobile in downtown areas, by offering new modern and convenient public transportation systems. Examples of these, are the *Cycab* robot[1] and the $\pi - Car$ prototype of IEF[2].

The kinematic structure of these robots differs from that of a car-like vehicle in that it allows the steering of the rear wheels as a function of the front wheels. We call a vehicle showing this feature a bi-steerable car (or BiS-car for short).

Endowed with autonomy capacities, the bi-steerable car ought to combine suitably and safely a set of *abilities* that eventually could come to the relief of the end-user in complex tasks (e.g. parking the vehicle). Part of these abilities have been tackled separately in previous work [12, 6, 13]. The motion planning problem for the general BiS-car was solved thanks to its *flatness*[5] property.

In this paper we address the integration of three essential autonomy abilities: map-building & localization, motion planning amidst obstacles and trajectory execution. Regarding the latter, we discuss the linearization of the system that allowed us to use a linear control law for trajectory tracking. The experimental results presented in the paper allow to confirm the theoretical considerations about the BiS-car. We are convinced that these results represent a first step towards the motion autonomy of this kind of transportation system. Moreover, as the enhanced maneuverability of a BiS-car makes it an interesting alternative to the conventional car-like robot, we hope that these results will be encouraging as well.

Thus in section 2, we sketch the environment reconstruction and localization methods we used. Then in section 3 we recall how the central issue regarding the motion planning problem for the general BiS-car was solved. The linearization of such a system and the implementation of a linear trajectory tracking control law are discussed in section 4. In section 5 we present experimental settings showing the *fusion* of these essential autonomy capacities in our bi-steerable platform the Cycab robot. We close the paper with some concluding remarks and guidelines on future work in section 6.

2 Map-building and Localization

The objective was to generate a map which could be used for localization during motion execution and another map for motion planning amongst obstacles. The specification of the system included reliability, accuracy and flexibility. In this section, we will give an overview on how we reached these goals. The reader may refer to [12] for more details about the subject.

2.1 The Localization Map

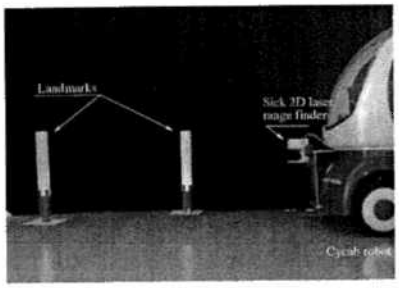
For localization purposes, we did not want to focus on the detection of natural features in the environment, since such detection is often subject to failure and not very accurate. Thus, in order to ensure reliability, we decided to install artificial landmarks in the environment leading to an accurate detection and to a robust matching. The obstacle map is discussed further below.

The artificial landmarks are cylinder covered with reflector sheets, specially designed for our Sick laser range finder. They are easy to detect, and can be localized with great accuracy. Yet, they can't be identified directly from the sensor output. We will see in the following how we solve this identification problem

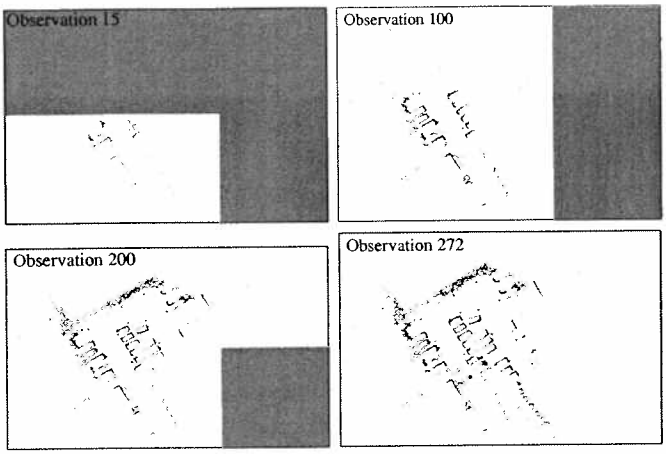
[1] designed at INRIA and currently traded by the Robosoft company (see www.robosoft.fr).

[2] Institut d'Electronique Fondamentale, Université Paris-Sud.

Figure 1: *Localization with landmarks and map building*



(a) Cycab robot and landmarks



(b) Obstacle map evolution: Experimental images during the obstacle map-building phase. The vehicle is driven within the car-park area as long as needed. Simultaneously, the laser range sensor is used to detect the landmarks to build-up the localization map

Figure 2(a) shows our robot, its sensor and the landmarks.

The Cycab robot is the size of a golf-cab capable of attaining up to 30Km/h. Its "natural" environment is the car-park area of the INRIA Rhône-Alpes (about $10000m^2$). In order to keep flexibility, we wanted to be able to equip the environment with non permanent beacons. Consequently, there is not a definitive landmark map. Hence, we had to build a system able to learn the current state of the car-park area, leading us to use SLAM[3] methods.

The method which was best suited to our needs was the Geometric Projection Filter [4]. It consists in building a map of features uncorrelated with the robot state. Such features are, for instance, the distance between landmarks and angles between three of them.

Owing to the accuracy of the laser range finder, to the good choice of our landmarks, and to the strength of the SLAM methods we use, we evaluate the accuracy of the localization system to the following value: about 10 centimeters in position and 2 degrees in orientation. We refer the reader to [12] for more details about the way we evaluate these values.

2.2 Identifying the observations

2.2.1 Difficulties. Since our landmarks are undistinguishable, their identification must be done by the localization software. For instance, assume that the robot knows a set $L = \{(x_i, y_i), i = 1..p\}$ of landmarks. After a sensor reading, we get a set $O = \{o_i = (r_i, \theta_i), i = 1..n\}$ of landmark measurements in the sensor frame. Among these observations, some are known landmarks, some are real landmarks seen for the first time and others are detection errors. The objective of the matching process is to find a function $\mathcal{I}$ from $[1..n]$ to $[1..p] \cup \{\emptyset\}$ such that, for $i \in [1..n]$, $\mathcal{I}(i) = j$ iff o_i is an observation of L_j and $\mathcal{I}(i) = \emptyset$ iff o_i cannot be identified.

To find this function, we would like to use a method robust with respect to the errors in the estimation of the robot position and robust to erroneous detections. Figure 2 illustrates the complexity of this task.

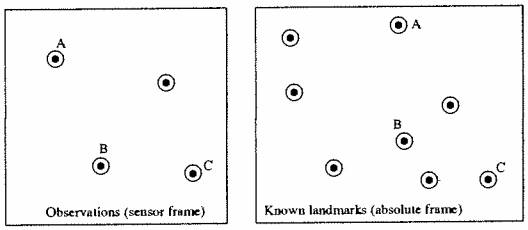


Figure 2: *The matching process*

2.2.2 Correspondence graphs : . A possible way of identifying the observations is to use the correspondence graph notion, as described in [1]. Basically, a correspondence graph is a graph whose nodes are potential matches $O_i \leftrightarrow L_j$ and whose edges express the relation "is consistent with".

The matching will be done in three steps:

[3]Simultaneous Localization And Mapping

1. From L, build a database of the invariant feature. For instance, with distance invariant, for each pair of landmarks (L_i, L_j), store the distance $d(L_i, L_j)$.
2. Look for the observed invariant features in the database and add corresponding edges to the matching graph. Again, with the distance invariant, for each pair (O_m, O_n), look for the pairs (L_i, L_j) whose distance is the closest from $d(O_m, O_n)$ in the database and add edges $(O_m, L_i) \leftrightarrow (O_n, L_j)$ and $(O_n, L_i) \leftrightarrow (O_m, L_j)$ to the correspondence graph.
3. Look for a maximum clique[4] in the correspondence graph. Its nodes correspond to the maximum subset of matches which are all consistent with each others. They will give the identification of each observation.

For the sake of efficiency, step 1 should not be done at each iteration. This database can be maintained over the iterations by using appropriate algorithms (see [12] for details about this point).

2.2.3 JCBB algorithm : . An other way of doing this identification is to use the *Joint Compatibility Branch and Bound* (JCBB) presented by Tardos in [11]. Basically, this algorithm is a heuristic for finding the biggest sets of matches all consistent with each others. Instead of building the correspondence graph, the JCBB algorithm explore the space of hypothesised $\mathcal{I}$ functions in order to find the ones which explains the biggest set of observation.

In this algorithm, a hypothesised $\mathcal{I}$ function is represented as an array $H = \mathcal{I}([1..p])$ which is built recursively: all possible identifications for O_1 are tried, then for each hypthesized value of $\mathcal{I}(1)$, possible identification of O_2 are tried. The algorithm discards the hypothesis which do not satisfy consistency constraints. For instance if $d(O_1, O_2)$ is close enough to $d(L_{\mathcal{I}(1)}, L_{\mathcal{I}(2)})$, then this value of $\mathcal{I}(2)$ is acceptable.

The algorithm continues until all observations have been treated. At this point, only the hypothesis resulting in the biggest set of identified observations are kept. In order to limit the complexity of the algorithm, only the hypothesis which can result in a set of identified observations bigger than the current best one are treated.

2.2.4 Remarks. Both correspondence graphs and JCBB are robust algorithms which can be used to solve the data association problem. So we can wonder which one should be chosen. We believe that this choice is not relevant. Indeed these two algorithms are means to identify as many observation as possible, trying to maintain as much consistency between these identifications. The main difference is that when using a correspondence graph, the maximum clique extraction should be implemented with some efficient heuristic which still has to be chosen, although the JCBB is immediatly a heuristic to reach this goal.

[4] In a graph, a clique is a subset of nodes which are all connected to each others.

2.3 The Obstacle Map

The previous method builds only a landmark map. Therefore we had also to build a map of observed obstacles in order to plan safe paths. To achieve this goal, we build an occupancy grid[3] on the environment. This structure gives us informations correlated with the probability that a given place is occupied by an obstacle.

Both maps are built online, in real-time, by the robot during the reconstruction phase. Figure 2(b) shows how the obstacle map evolves while we are exploring the environment. This map is made of small patches which are added according to the need of the application. In this way, the map can be extended in any direction, as long as memory is available. Once the map-building phase has finished, the obstacle map is converted into a pixmap and passed to the Motion Planning stage.

3 Motion Planning Amidst Obstacles

The Motion Planner adopted for the Cycab was presented in [13]. Essentially, it is a two step approach, dealing separately with the physical constraints (the obstacles) and with the kinematic constraints (the nonholonomy). The planner first builds a collision-free path without taking into account the nonholonomic constraints of the system. Then, this holonomic path is approximated by a sequence of collision-free feasible paths computed by a *suitable*[5] steering method. Finally, the resulting path is smoothed.

We shall now discuss the practical aspects of the overall planning scheme.

3.1 Feasible paths for a BiS-car

One way of designing steering methods for a nonholonomic system is to use its *flatness* property [5]. This is what we did for the general BiS-car for which a flat output—or linearizing output—was given in [13]. Flatness allows also for feedback linearization of the nonlinear system as discussed in section 4.

The flat output $\mathbf{y} = (y_1, y_2)^T$ of a BiS-car are the coordinates of a point H $(x_H, y_H)^T = (y_1, y_2)^T$ in a global cartesian frame (see Figure 3). Thus, for a robot reference frame placed at point F in Figure 3, it was shown in [13] the flat output is computed as a

[5] i.e. Verifying the topological property as explained in [13].

function of the state[6] variables as follows:

$$\vec{P_H} = \vec{P_F} + \mathcal{P}(\varphi)\vec{u}_\theta + \mathcal{Q}(\varphi)\vec{u}_{\theta^\perp}$$

where $\mathcal{P}(\varphi)$ and $\mathcal{Q}(\varphi)$ are coordinate functions relative to the robot's reference frame (see [13] for details) and where $\vec{u}_{(.)}$ (resp. $\vec{u}_{(.)^\perp}$) is the unitary vector in the direction (.) (resp. the direction (.) + $\frac{\pi}{2}$).

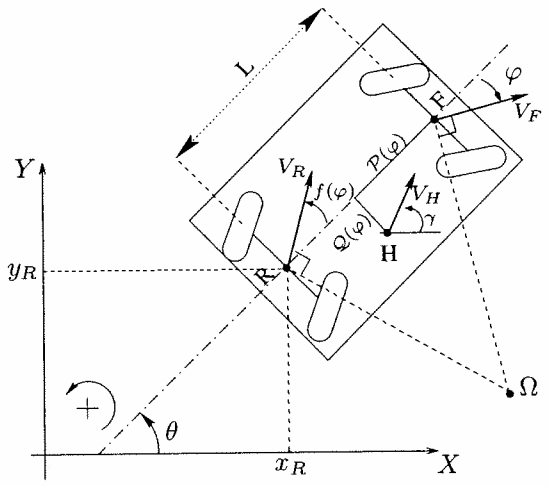


Figure 3: *Kinematics model of a bi-steerable car showing the coordinates of the flat output (point H) with respect to the reference frame of the robot placed at point F. In our case we have that $(x_F, y_F, \theta, \varphi)$ is the state of the robot.*

The striking advantage of planning a path in the flat space is that we only need to parameterize a 2-dimensional curve whose points and derivatives define everywhere the current n-dimensional state[7] of the robot (in the case of the BiS-car $n = 4$). The main characteristic of such a curve is its curvature κ, defined as

$$\kappa = det\left(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}\right) / \left(\left[y_1^{(1)}\right]^2 + \left[y_2^{(1)}\right]^2\right)^{3/2} \quad (1)$$

It was shown in [13] that the relation between the curvature and the front steering angle of the general bi-steerable vehicle is

$$\kappa = \mathbf{K}(\varphi) = -\frac{\beta'(\varphi)}{\mathcal{M}'(\varphi) - \beta'(\varphi)\mathcal{N}(\varphi)} \quad (2)$$

where (')$\equiv \partial/\partial\varphi$, the function $\beta(\varphi)$ is the characteristic angle of the velocity vector of the flat output and where $\mathcal{M}$,$\mathcal{N}$ are coordinate functions of H (see [13] for details).

[6]The configuration space in robotics is called the *state space* in control theory, so we will use indistinctly both terms.

[7]The configuration space in robotics is called the *state space* in control theory, so we will use indistinctly both terms.

Expression (2) is far from being invertible analytically[8]. Hence we tabulated 5000 values of the steering angle against the corresponding curvatures. Thence, to perform the inversion we compute the current curvature state from the expression (1) and perform a cubic interpolation between the steering angles defining the interval where the curvature value falls in.

Figure 4 shows the outcome of the motion planner using an obstacle map generated as described in the previous section.

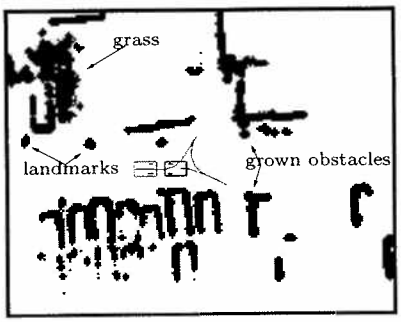


Figure 4: *Simulated path computed by the motion planner in a work-station using a real obstacle map generated by the previously described map-building stage. The obstacles are grown as well as the robot before computing the path.*

3.2 User-Planner Interface

The User-Planner interface in the Cycab is achieved through a *touch-screen* superposed to a 640×480 pixels LCD display. The display may be used in text mode or graphics mode by direct access to video memory (by using SVGA-lib). Additionally, we use the keyboard to allow for the entrance of data.

The interface is used to display the current position of the robot within its environment and to capture the goal position entered by the user. These positions together with the obstacle map is passed to the motion planner. The output path is then displayed allowing the user to validate the path or start a new search.

Finally, the reference trajectory is generated using a regular parameterization of the path [8] and the user is requested to accept to start the execution of the trajectory.

4 Trajectory tracking using flatness

It is well known that a nonholonomic system cannot be stabilized using only smooth state static feedbacks [2]. On the other hand, flat systems are feedback linearizable by means of a restricted class of dynamic feedback called *endogenous* [5]. The interest is that we are able to use state-of-the-art linear control techniques to stabilize the system around a nominal trajectory. We present here results coming from

[8]Yet, it is not surprising that it is close to a tan function.

recent work on feedback linearization of the general BiS-car.

4.1 Open-loop controls of the BiS-car

Looking for a tractable relation between the controls of the robot and the linearizing output, we found an expression giving the flat output dynamics with respect to a more convenient reference frame [7]. This new frame has orientation $\gamma = [\theta + \beta(\varphi)] \pm \pi$ and is placed at the middle of the front axle of the robot (point F).

The convenience of this new reference frame relies on the fact that the velocity of the flat output has a single component in it. More precisely—assuming that $\gamma = \theta + \beta(\varphi) + \pi$—one can show that, in this reference frame, the flat output dynamics is given by the following expression [7]:

$$\begin{aligned} \dot{\vec{P}}_H &= v_H \, \vec{u}_\gamma \\ v_H &= v_F[\cos(\varphi - \beta - \pi) - \mathcal{N}\mathcal{F}] + \omega_\varphi[\mathcal{M}' - \beta'\mathcal{N}] \end{aligned} \tag{3}$$

where: (v_F, ω_φ) are the controls[9] of the robot (i.e. the linear and the front steering angle speeds), $(\varphi - \beta - \pi)$ is the angle subtended between the velocity vector $\vec{V}_F$ of point F and the velocity vector of the flat output $\vec{V}_H$ (see Figure 3), and $\mathcal{F}(\varphi) = \frac{\sin(\varphi - f(\varphi))}{L\cos(f(\varphi))}$.

From expressions (3) and (2) the open-loop controls of the robot can be found as soon as the trajectory of point H is known. Indeed, derivating expression (2) one has that $\omega_\varphi = \left(\frac{\mathrm{d}\kappa}{\mathrm{d}t}\right)/\mathcal{K}'$. Hence the controls of the robot are known as soon as the course speed of point H ($v_H = \sqrt{\dot{x}_H^2 + \dot{y}_H^2}$) and the derivative of the curvature along the path described by H are known:

$$\begin{aligned} \omega_\varphi &= \left(\frac{\mathrm{d}\kappa}{\mathrm{d}t}\right) \cdot \frac{1}{\mathcal{K}'} \\ v_F &= \frac{v_H - \omega_\varphi(\mathcal{M}' - \beta'\mathcal{N})}{\cos(\varphi - \beta - \pi) - \mathcal{N}\mathcal{F}} \end{aligned}$$

4.2 Linearized system and control law

As we are interested in stabilizing the BiS-car around a reference trajectory, we explored the fact that, owing to the flatness property, the system is diffeomorphic to a linear controllable one [5]. The endogenous dynamic feedback that linearizes the general bi-steerable system is presented in [7].

Roughly speaking, the flatness property of the system allows for the computation of this feedback by means of successive derivations of expression (3) until controls appear[10] (notice that we need $\left(\frac{\mathrm{d}\kappa}{\mathrm{d}t}\right)$). The linearizing feedback allows to deal with the following controllable canonical linear system:

$$\dot{\tilde{y}} = \underbrace{\begin{bmatrix} \mathbf{A}_{c1} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{c2} \end{bmatrix}}_{\mathbf{A}} \tilde{y} + \underbrace{\begin{bmatrix} \mathbf{B}_{c1} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_{c2} \end{bmatrix}}_{\mathbf{B}} \begin{pmatrix} y_1^{(3)} \\ y_2^{(3)} \end{pmatrix} \tag{4}$$

where the canonical matrices $\mathbf{A}_{ci}$ and $\mathbf{B}_{ci}$, $i = 1, 2$ read as follows

$$\mathbf{A}_{ci} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \qquad \mathbf{B}_{ci} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix},$$

and where we have defined the 6-dimensional vector $\tilde{y} := (\tilde{y}_1, \tilde{y}_2)^T$ as the new state, with $\tilde{y}_i = (y^{(0)}, y^{(1)}, y^{(2)})^T$ for $i = 1, 2$.

Therefore, if we let

$$\begin{pmatrix} y_1^{(3)} \\ y_2^{(3)} \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix},$$

from linear control theory, the closed-loop control stabilizing the reference trajectory $\mathbf{y}^*$ is given by:

$$y_i^{(3)} = (y_i^*)^{(3)} - \sum_{j=0}^{2} k_{i,j} \left(y_i^{(j)} - (y_i^*)^{(j)} \right) \quad i = 1, 2 \tag{5}$$

where the coefficients $k_{i,j}$ are chosen in order that the linear time invariant error dynamics

$$e_i^{(3)} = \sum_{j=0}^{2} k_{i,j} e_i^{(j)} \quad i = 1, 2$$

with $e_i^{(j)} = \left(y_i^{(j)} - (y_i^*)^{(j)} \right)$, is stable.

5 Experimental Settings

We tested the integration of these essential autonomy capacities in our experimental platform the Cycab robot. The aim was to validate the theoretical considerations made for the BiS-car and to get insight into the limitations of the whole motion scheme.

The computation power onboard the Cycab is a *Pentium II*™ 233MHz running a RedHat™ Linux system. All programs were written in C/C++ language.

During the experiments the front-to-back steering function was set to $f(\varphi) = -\varphi$ and the speed of the robot was limited to $1.5ms^{-1}$. The control rate of the robot was fixed at $50ms$. The throughput rate of the laser range-finder is limited to $140ms$[11]; therefore the control system relies momentarily in odometry[6] readings.

Figure 5 is a set of pictures showing a complete application integrating the stages described throughout the paper. Figure 6 shows the evolution of the

[9] Also called the *inputs* in control theory.

[10] This procedure is formalized in the *dynamic extension algorithm* .

[11] This rate is fair enough for our needs, even though we could use a real-time driver.

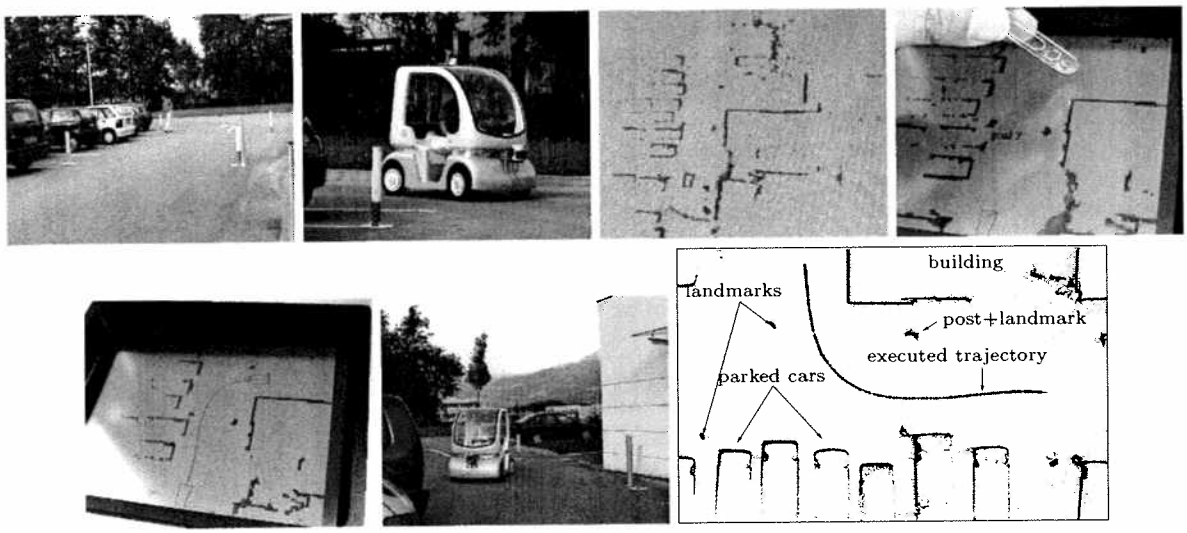


Figure 5: *An experimental setting showing from left top to bottom right: The arbitrary placing of the landmarks; the manual driving phase for landmark and obstacle map-building; the obstacle map generated together with the current position of the robot as seen on the LCD display; the capture of the goal position given by the user by means of the touch-screen; the reference path found by the motion planner; the execution of the maneuver using the linear control law (5); and the executed trajectory recovered after the experiment.*

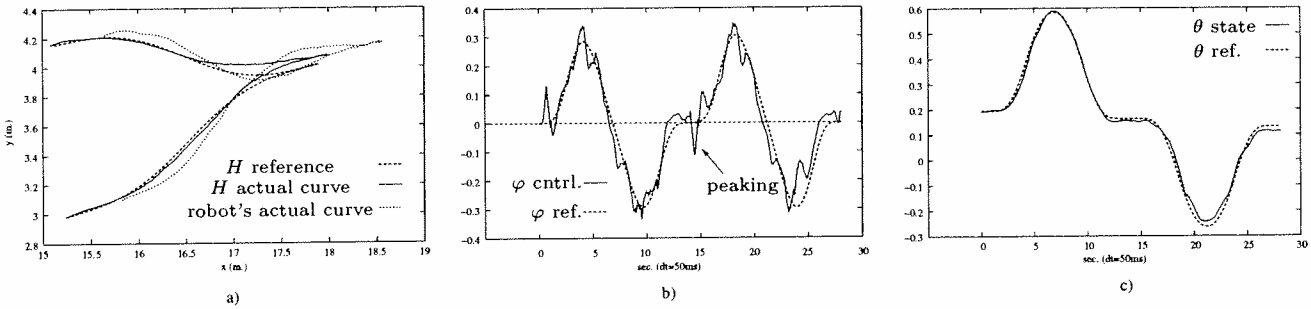


Figure 6: *Parking maneuver experimental setting: a) Reference curve for point H and actual curves for this point and for the robot (i.e. point F); b) Reference and actual curves of the steering angle control φ (the control system is sensible at cusp points inducing "peaks"); c) Reference and actual orientation of the robot θ.*

flat output and the robot during a short parking maneuver. A couple of remarks may be done regarding these curves: The controller is able to track the reference curves for the 4-dimensional state (x, y, φ, θ). The peaking phenomenon shown for the steering angle is intrinsic to the feedback at cusp points. The performance of the controller is of good quality if we consider that the speed of the robot is low.

6 Conclusions

The modest computation capacity onboard our experimental platform allowed us mainly to validate the theoretical approach regarding the flatness property of the bi-steerable car. The technical developments discussed in the paper allowed us to successfully integrate a basic application, representing the first step towards motion autonomy of this new kind of transportation system. Taking further steps requires work regarding the safety and robustness of the execution accounting further for uncertainty issues and for a dynamical environment. We would like to focus our work in this direction. In particular, obstacle avoidance techniques and dynamic planning ([9]) ought to be integrated in the future as well as formal verification issues in the software architecture (e.g. using Orccad[14] development tool).

Acknowledgements- This work was partially supported by the Inria LaRA *(La Route Automatisée - 1998-2002)* program on urban public transport and partially by CONACYT *Consejo Nacional de Ciencia y Tecnologia (Mexico).*

References

[1] T. Bailey, E.M Nebot, J.K. Rosenblatt, and H.F. Durrant-Whyte. Data association for mobile robot navigation : a

graph theoretic approach. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 2000.

[2] R.W. Brockett. Asymptotic stability and feedback stabilization. In R.W. Brockett, R.S. Millman, and H.J. Sussmann, editors, *Differential Geometric Control Theory*, pages 181–191, Boston, MA: Birkhäuser, 1983.

[3] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *AAAI/IAAI, Vol. 2*, pages 896–901, 1996.

[4] M. Deans and M. Hebert. Invariant filtering for simultaneous localization and mapping. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 2001.

[5] M. Fliess, J. Lvine, P. Martin, and P. Rouchon. Flatness and defects of nonlinear systems: introductory theory and examples. *Int. Journal of Control*, 61(6):1327–1361, 1995.

[6] J. Hermosillo, C. Pradalier, and S. Sekhavat. Modelling odometry and uncertainty propagation for a bi-steerable car. In *Proc. of the IEEE Intelligent Vehicle Symp.*, Versailles (FR), June 2002. Poster session.

[7] J. Hermosillo and S. Sekhavat. Feedback control of a bi-steerable car using flatness; application to trajectory tracking. In *Proc. of the American Control Conference*, Denver, CO (US), June 2003.

[8] F. Lamiraux, S. Sekhavat, and J.-P. Laumond. Motion planning and control for hilare pulling a trailer. *IEEE Trans. Robotics and Automation*, 15(4):640–652, August 1999.

[9] F. Large, S. Sekhavat, Z. Shiller, and C. Laugier. Using non-linear velocity obstacles to plan motions in a dynamic environment. In *Proc. of the Int. Conf. on Control, Automation, Robotics and Vision*, Singapore (SG), December 2002.

[10] Ch. Laugier, S. Sekhavat, L. Large, J. Hermosillo, and Z. Shiller. Some steps towards autonomous cars. In *Proc. of the IFAC Symp. on Intelligent Autonomous Vehicles*, pages 10–18, Sapporo (JP), September 2001.

[11] J. Neira and J.D. Tardos. Data association in stochastic mapping using the joint compatibility test. *IEEE Trans. Robotics and Automation*, Vol. 17:890–897, 2001.

[12] C. Pradalier and S. Sekhavat. Concurrent localization, matching and map building using invariant features. In *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, 2002.

[13] S. Sekhavat, J. Hermosillo, and P. Rouchon. Motion planning for a bi-steerable car. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 3294–3299, Seoul (KR), May 2001.

[14] D. Simon, B. Espiau, K. Kapellos, and Pissard-Gibollet R. The orccad architecture. *Int. Journal of Robotics Research, Special issues on Integrated Architectures for Robot Control and Programming*, vol.17(4):338–359, April 1991.