Safe and Autonomous Navigation for a Car-Like Robot Among Pedestrian

Cédric Pradalier, Jorje Hermosillo, Carla Koike, Christophe Braillon, Pierre Bessiere, Christian Laugier

Safe and Autonomous Navigation for a Car-Like Robot Among Pedestrian

**C. Pradalier, J. Hermosillo, C.Koike, C.Braillon, P.Bessière,
C.Laugier**
INRIA Rhône-Alpes
655 av. de l'Europe, Montbonnot, 38334 St. Ismier Cedex, France
http://www.inrialpes.fr

October 15, 2003

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. Endowed with autonomy capacities, the bi-steerable car ought to combine suitably and safely a set of abilities: simultaneous localization and environment modeling, motion planning and motion execution amidst dynamic obstacles. In this paper we address the integration of these four essential autonomy abilities into a single application. Specifically, we aim at reactive execution of planned motion. We address the fusion of controls issued from the control law and the obstacle avoidance module using probabilistic techniques.*

Keywords — Car-like robot, navigation, path planning, obstacle avoidance, autonomy.

Safe and Autonomous Navigation for a Car-Like Robot Among Pedestrian

C. Pradalier, J. Hermosillo, C.Koike, C.Braillon, P.Bessière, C.Laugier

GRAVIR – INRIA – INPG Grenoble

INRIA Rhône-Alpes, 38334 Saint 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. Endowed with autonomy capacities, the bi-steerable car ought to combine suitably and safely a set of abilities: simultaneous localization and environment modeling, motion planning and motion execution amidst dynamic obstacles. In this paper we address the integration of these four essential autonomy abilities into a single application. Specifically, we aim at reactive execution of planned motion. We address the fusion of controls issued from the control law and the obstacle avoidance module using probabilistic techniques.

Index Terms— Car-like robot, navigation, path planning, obstacle avoidance, autonomy.

I. 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 – designed at INRIA and currently traded by the Robosoft company (see www.robosoft.fr) – and the pi-Car prototype of IEF (Institut d'Electronique Fondamentale, Universite Paris-Sud).

The kinematic structure of these robots differs from that of a car-like vehicle in that it allows the steering of both the front axle and the rear one. 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: simultaneous localization and environment modeling, motion planning execution amidst static obstacles and obstacle avoidance in a moderately dynamic environment without accounting for a planned motion.

In this paper we address the integration of these four essential autonomy abilities into a single application. Specifically, we aim at reactive execution of planned motion. We address the fusion of controls issued from the control law and the obstacle avoidance module using probabilistic techniques. We are convinced that these results represent a step further towards the motion autonomy of this kind of transportation system. The structure of the paper follows.

In section 2, we sketch the environment reconstruction and localization methods we used and we recall how the central issue regarding the motion planning and execution problem for the general BiS-car was solved. Section 3 explains how our obstacle avoidance system was designed and section 4 how it was adapted to the trajectory tracking system. 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.

II. LOCALIZATION, ENVIRONMENT MODELING, MOTION PLANNING AND EXECUTION

In the design of an autonomous car-like robot, we are convinced that localization, modeling of the environment, path planning and trajectory tracking are of fundamental importance.

A. Map-building and Localization
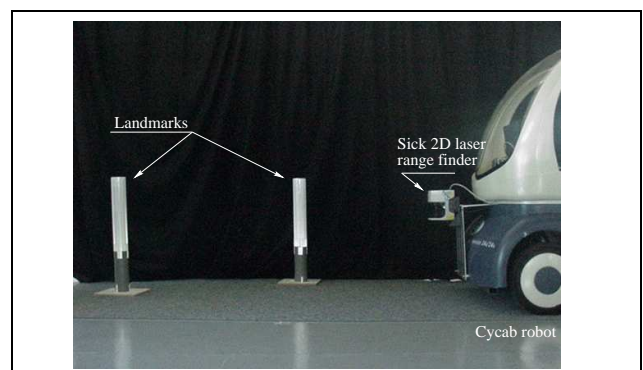
Fig. 1: Cycab robot and landmarks

The CyCab robot is the size of a golf-cab capable of attaining up to 30Km/h. Its "natural" environment

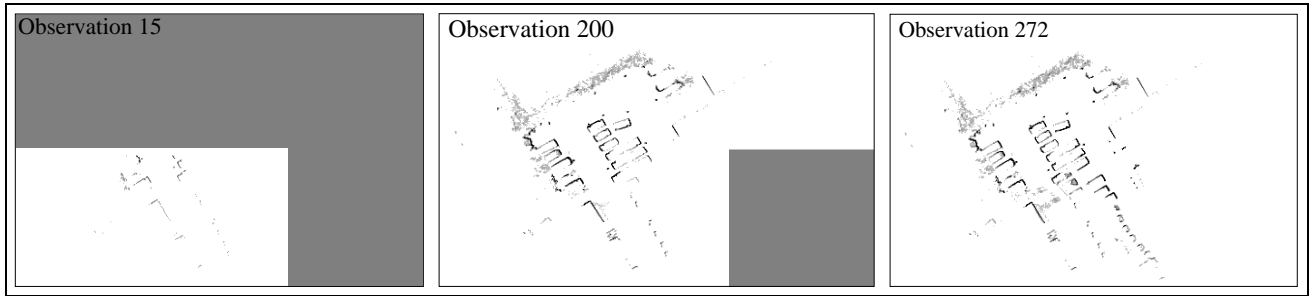**Fig. 2:** Obstacle map evolution: Experimental images during the obstacle map-building phase. The vehicle is driven within the car-park area as long as needed. Simultaneously, the laser range sensor is used to detect the landmarks to build-up the localization map.

is the car-park area of the INRIA Rhône-Alpes (about $10000m^2$). 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. So, in order to ensure reliability, we decided to install artificial landmarks in the environment. These landmarks had to be detected easily and accurately, and they should be identified with a reasonable computation effort. Fig. 1 shows our robot, its sensor and the landmarks : cylinder covered with reflector sheets, specially designed for our Sick laser range finder.

Moreover, in order to keep flexibility, we wanted to be able to equip the environment with non permanent beacons. So we could not rely on a definitive landmark map, and we had to build a system able to learn the current state of the car-park area. This led us to use SLAM[1] methods. The method which was best suited to our needs was the Geometric Projection Filter (see [12] for details). 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 our 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.

B. The Obstacle Map

The previous method localizes the robot and builds a landmark map. But, we still miss 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 construction phase. Fig. 2 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.

C. Motion Planning Amidst Obstacles

The Motion Planner adopted for the Cycab was presented in [14]. Essentially, it is a two step approach, dealing separately with the physical constraints (the obstacles) and with the kinematic constraints (the non-holonomy). The planner 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 *suitable*[2] steering method. Finally, the resulting path is smoothed.

A key issue in nonholonomic motion planning is to find a steering method accounting for the kinematics of the robot. One way of designing steering methods for a nonholonomic system is to use its *flatness* property [5] allowing also for feedback linearization of the nonlinear system (this is discussed in section II-F). This is what we did for the general BiS-car for which a flat output—or linearizing output—was given in [14].

D. Steering a BiS-car

The kinematics model of a general bi-steerable vehicle and its flat output are shown in Fig. 3.

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[3] of the robot

---

[1]Simultaneous Localization And Mapping

[2]i.e. Verifying the topological property as explained in [14].

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

Fig. 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.

(in the case of the BiS-car $n = 4$). The main characteristic of such a curve is its curvature κ, defined as

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

It was shown in [14] 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.

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

Fig. 4: Simulated path computed by the motion planner on 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.

E. User-Planner Interface

The User-Planner interface in the Cycab is achieved through a *touch-screen* superposed to a 640×480 pixels LCD display. 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.

F. Trajectory tracking using flatness

It is well known that a nonholonomic system cannot be stabilized using only smooth state static feedbacks [1]. Ever since then, time-varying feedbacks [13] and dynamic feedbacks have been successfully used in particular for the canonical tractor-trailer and car-like robots [4].

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. We present here results coming from recent work on feedback linearization of the general BiS-car.

For a reference frame of the robot placed at point F in Fig. 3, the flat output $\mathbf{y} = (y_1, y_2)^T$ of a BiS-car are the coordinates of a point H $(x_H, y_H)^T = (y_1, y_2)^T$, computed as a function of the state 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 [14] for details) and where $\vec{u}_\theta$ (resp. $\vec{u}_{\theta\perp}$) is the unitary vector in the direction θ (resp. the direction $\theta + \frac{\pi}{2}$).

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 having orientation $\gamma = [\theta + \beta(\varphi)] \pm \pi$ and 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]:

$$\dot{\vec{P_H}} = v_H \vec{u}_\gamma \quad (3)$$
$$v_H = v_F[\cos(\varphi - \beta - \pi) - \mathcal{NF}] + \omega_\varphi[\mathcal{M}' - \beta'\mathcal{N}]$$

where (v_F, ω_φ) are the controls of the robot (i.e. the heading and the front-steering speeds), $(\varphi - \beta - \pi)$ is

the angle subtended between the velocity vector of the robot $\vec{V_F}$ and the velocity vector of the flat output $\vec{V_H}$ (see Fig. 3) and $\mathcal{F}(\varphi) = \frac{\sin(\varphi - f(\varphi))}{L\cos(f(\varphi))}$.

From expression (3) the open-loop controls of the robot can be found as soon as the trajectory of point H is known. 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]. Then, from linear control theory, it can be shown that the closed-loop control stabilizing the reference trajectory $\mathbf{y}^*$ has the followin form :

$$y_i^{(3)} = (y_i^*)^{(3)} - \sum_{j=0}^{2} k_{i,j}\left(y_i^{(j)} - (y_i^*)^{(j)}\right) \quad i = 1, 2 \quad (4)$$

Where $(.)^{(p)}$ stands for the total derivative of order p, see [2] for details.

III. OBSTACLE AVOIDANCE USING PROBABILISTIC REASONING

The previous approach considers trajectories in a static environment. In order to make the execution of these trajectories more robust, an obstacle avoidance system should be prepared to react to unpredicted changes in the environment. This section presents the principles of our obstacle avoidance module.

We are convinced that probabilities are the ideal tool to cop with the uncertainty of both obstacle perception and action decision. With this in mind, we designed an obstacle avoidance system expressed as a probabilistic inference problem.

A. Specification

The CyCab can be commanded through a speed V and a steering angle Φ. It is equipped with π radians sweeping laser range finder. As we consider this sensor to be too accurate for the obstacle avoidance task, we summarized its output as 8 values : the distances to the nearest obstacle in a $\pi/8$ angular sector(see Fig. 5). We will call $D_k, k = 1\ldots8$ the probabilistic variables corresponding to these measures.

Besides, we will assume that this robot is commanded by some high-level system (trajectory following for instance) which provides him with a pair of desired commands (V_d, Φ_d).

Our goal is to find commands to apply to the robot, guarantying the vehicle security while following the desired command as much as possible.
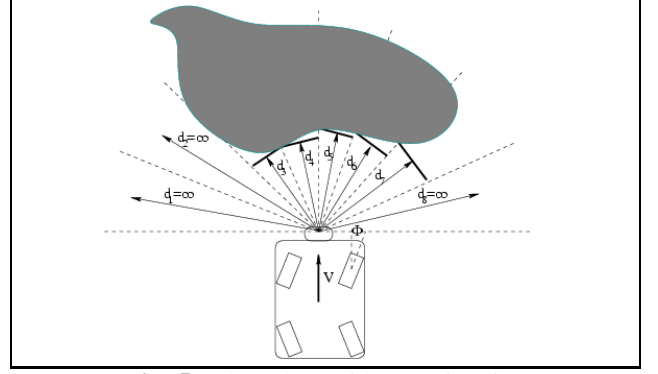

Fig. 5: Obstacle avoidance: situation

B. Sub-models definition

In angular sector i, we define a probability distribution over (V, Φ) knowing the desired commands and the distance D_i measured in this sector:

$$P_i(V\Phi \mid V_d\Phi_dD_i) = P_i(V \mid V_dD_i)P_i(\Phi \mid \Phi_dD_i) \quad (5)$$

where $P_i(V \mid V_dD_i)$ and $P_i(\Phi \mid \Phi_dD_i)$ are gaussian distributions respectively centered on $\mu_V(V_d, D_i)$ and $\mu_\Phi(\Phi_d, D_i)$ with standard deviation $\sigma_V(V_d, D_i)$ and $\sigma_\Phi(\Phi_d, D_i)$. Functions $\mu_V, \mu_\Phi, \sigma_V, \sigma_\Phi$ are defined with sigmoid shape as shown in Fig. 6.

There is two specific aspects to notice in this figure. First, concerning the means μ_V and μ_Φ, we can see that, the farther the obstacle, the closer to the desired command μ will be, and conversely, the nearer the obstacle, the more secure μ: minimal speed, strong steering angle.

Second, the standard deviation can be seen as a constraint level. For instance, when an obstacle is very close to the robot (small D_i), its speed *must* be strongly constrained to zero, this is expressed by a small standard deviation. Conversely, when obstacle is far, robot speed *can* follow the desired command, but there is no damage risk in not applying exactly this command. This low level constraint is the result of a big standard deviation.
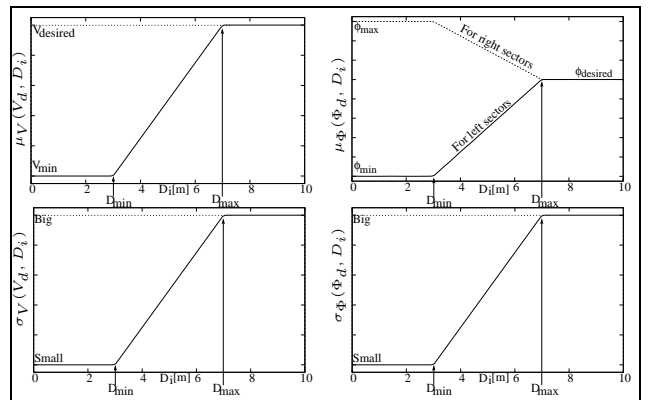

Fig. 6: Evolution of mean and standard deviation of $P_i(V \mid V_dD_i)$ and $P_i(\Phi \mid \Phi_dD_i)$ according to distance measured

C. Command fusion

Knowing desired controls and distance to the nearest obstacle in its sector, each sub-model, defined by $P_i(V\Phi \mid V_d\Phi_d D_i)$, provides us with a probability distribution over the robot controls. As we have eight sectors, we will have to fuse the controls from eight sub-models. Then we will find the best control in term of security and desired control following.

To this end, we define the following joint distribution:

$$P(V\ \Phi\ V_d\ \Phi_d\ D_1 \ldots D_8\ S) =$$
$$P(D_1 \ldots D_8)P(V_d\ \Phi_d)$$
$$P(S)P(V\ \Phi\ \mid\ V_d\ \Phi_d\ D_1 \ldots D_8\ S) \qquad (6)$$

where variable $S \in [1 \ldots 8]$ is such that when $S = i$, sub-model i controls the robot. $P(D_1 \ldots D_8)$ and $P(V_d\Phi_d)$ are unknown distribution[4]. As there is no need to favor a specific sub-model, we define $P(S)$ as a uniform distribution. The semantic of S will be emphasized by the definition of $P(V\Phi \mid V_d\Phi_d D_1 \ldots D_8 S)$:

$$P(V\Phi \mid V_d\Phi_d D_1 \ldots D_8[S = i]) = P_i(V\Phi \mid V_d\Phi_d D_i)$$

Using equation 6, we can now express the distribution we are really interested in:

$$P(V\ \Phi\ \mid\ V_d\ \Phi_d\ D_1 \ldots D_8) = \qquad (7)$$
$$\sum_S (P(S)P(V\ \Phi\ \mid\ V_d\ \Phi_d\ D_1\ \ldots\ D_8\ S))$$

This equation is actually the place where the different constraint level expressed by functions σ_V and σ_Φ will be useful. The more security constraints there will be, the more peaked will be the sub-model control distribution. So sub-models who see no obstacles in their sector will contribute to the sum with flat distribution, and those who see perilous obstacles will add a peaky distribution, hence having more influence (see Fig. 7). Finally the command really executed by the robot is the one which maximize $P(V\ \Phi\ \mid\ V_d\ \Phi_d\ D_1 \ldots D_8)$ (eq. 7).

D. Results

Fig. 8 illustrates the result of the obstacle avoidance system applied on a simulated example. The simulated Cycab is driven manually with a joystick in a square environment. In this specific situation, the driver is continuously asking for maximum speed, straight forward (null steering angle). We can observe on the dotted trajectory that the obstacle avoidance curves the trajectory in order to avoid the walls. From the density of dots, we can figure out the robot speed: it breaks when it comes close to the walls and while its turning and try to follow desired speed when obstacles are not so threatening.

---

[4]Actually, as we know we will not need them in future computation, we don't have to specify them.
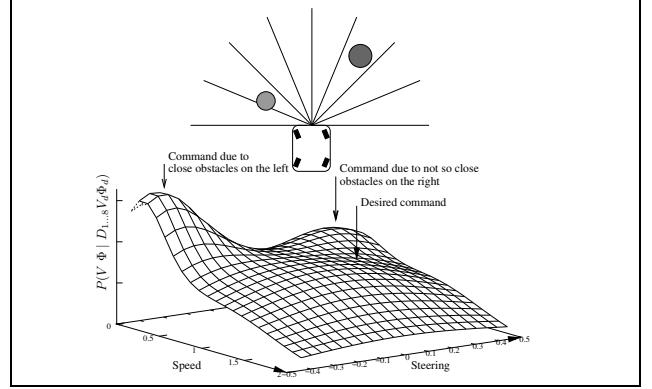
Fig. 7: Probability distribution over speed and steering, resulting from the obstacle avoidance system.
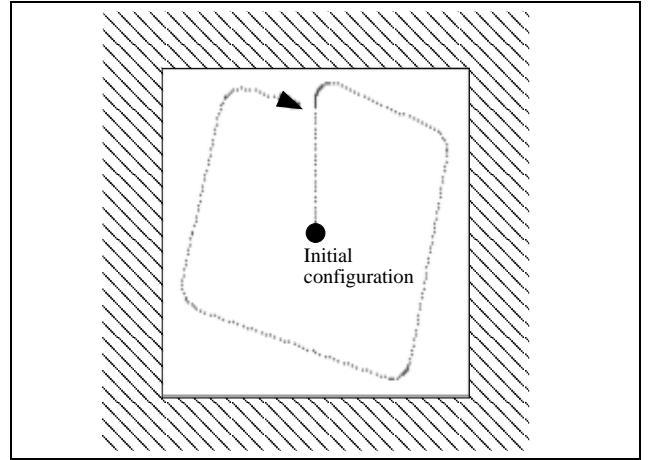
Fig. 8: Robot trajectory while driven manually with constant desired steering angle

E. Discussion

We found that the method presented in this section provided us with an efficient way to fuse a security system and orders from a high level system. Nevertheless the perturbations introduced in the trajectory following system by obstacle avoidance are such that they can make it become unstable. Next section will show how we integrate trajectory tracking and obstacle avoidance.

IV. TRAJECTORY TRACKING WITH OBSTACLE AVOIDANCE

While executing a trajectory, obstacle avoidance will modify certain commands in order to follow as much as possible desired orders while granting security. These modifications may introduce delay in the control loop. If no appropriate action is taken to manage these delays the control law may generate extremely strong accelerations or even become unstable when obstacles are gone. This is typically the case when our system evolves among moving pedestrians. Thus we designed a specific behavior to adapt smoothly our control system to the perturbations induced by obstacle avoidance.

A. Multiplexed trajectory tracking

a) Validity domain of flat control law: Experimentally, we found that the control law based on flatness can manage errors in a range of about 1 meter and 15 degrees around nominal trajectory. Furthermore, as this control law controls the third derivative of the flat output (eq. 4), it is a massively integrating system. For this reason, a constant perturbation such as immobilization due to a pedestrian standing in front of the vehicle will result in a quadratic increase of the control law output. This phenomena is mainly due to the fact that when obstacle avoidance slows the robot down, it breaks the dynamic rules around which the flat control law was built. So, there is no surprise in its failure.

b) Proportional control law: In order to deal with the situations that flat control law cannot manage, we designed a simplistic trajectory tracking behavior based again on probabilistic reasoning (section IV-B). As this behavior has many similarities with a weighted sum of proportional control laws, we do not expect it to be sufficient to stabilize the robot on its trajectory. Nevertheless, it is sufficient to bring it back in the convergence domain of the flat control law when obstacle avoidance perturbations have occurred. Basically, the resulting behavior is as follows: while the robot is close to its nominal position, it is commanded by flat control law. When, due to obstacle avoidance, it is too far from its nominal position, proportional control takes control, and try to bring it back to flat control law's convergence domain. When it enters this domain, flat control law is reinitialized and starts accurate trajectory tracking.

c) Time control: Path resulting from path planning (section II-C) is a list of robot configuration indexed by time. So when the robot is slowed down by a traversing pedestrian, it compensates its delay by accelerating. Nevertheless, when the robot is stopped during a longer time, let's say fifteen seconds, it should not consider to be delayed of fifteen seconds, otherwise it will try to reach a position fifteen second ahead, without tracking the intermediary trajectory. To tackle this difficulty, we introduced a third mode to the trajectory tracking: when the robot is really too far from its nominal position, we freeze the nominal position, and we use the proportional control to reenter the domain where nominal position can be unfrozen.

The global system is illustrated by Fig. 9: we implemented some kind of multiplexer/demultiplexer which manage transitions between control laws. In order to avoid oscillating between control laws when at the interface between two domains of validity, we had to introduce some hysteresis mechanism in the switching. This is illustrated in Fig. 10.
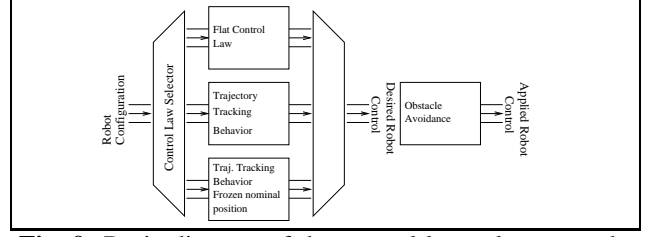
Fig. 9: Basic diagram of the control law selector mechanism
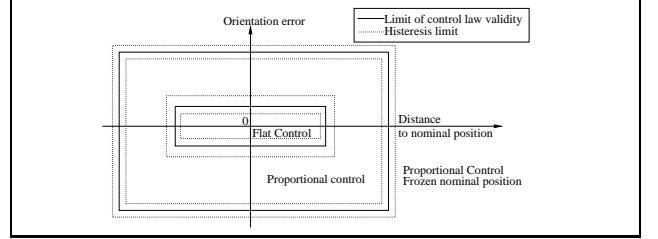
Fig. 10: Validity domains of the control laws

B. Trajectory tracking behavior

Our trajectory tracking behavior was built as a probabilistic reasoning, in a way similar to the obstacle avoidance presented above (section III). In this case, we use a mechanism of fusion with diagnosis[11] in the specification of our system. If A and B are two variables, we will define a diagnosis boolean variable I_A^B which express a consistency between A and B. Then, A and B will be called the *diagnosed variables* of I_A^B.

Our goal is to express the distribution over the desired controls (V_d, Φ_d) knowing reference controls (V_r, Φ_r) planned by the path planning stage, and error in position $(\delta X, \delta Y)$ and orientation $\delta\vartheta$ with respect to the nominal position.

In addition to the preceding variables, we will add four diagnosis variables $I_{V_d}^{\delta X}$, $I_{V_d}^{V_r}$, $I_{\Phi_d}^{\delta Y}$, $I_{\Phi_d}^{\delta\vartheta}$ and $I_{\Phi_d}^{\Phi_r}$. They will describe the relation between their diagnosed variables in the following joint distribution:

$$P(V_d \, \Phi_d \, V_r \, \Phi_r \, \delta X \, \delta Y \, \delta\vartheta \, I_{V_d}^{\delta X} \, I_{V_d}^{V_r} \, I_{\Phi_d}^{\delta Y} \, I_{\Phi_d}^{\delta\vartheta} \, I_{\Phi_d}^{\Phi_r}) = \quad (8)$$
$$P(V_d \, \Phi_d) P(V_r \, \Phi_r) P(\delta X \, \delta Y \, \delta\vartheta)$$
$$P(I_{V_d}^{\delta X} \mid V_d \, \delta X) P(I_{V_d}^{V_r} \mid V_d \, V_r)$$
$$P(I_{\Phi_d}^{\delta Y} \mid \Phi_d \, \delta Y) P(I_{\Phi_d}^{\delta\vartheta} \mid \Phi_d \, \delta\vartheta \, V_d) P(I_{\Phi_d}^{\Phi_r} \mid \Phi_d \, \Phi_r)$$

Using this joint distribution and Bayes rule, we will be able to infer

$$P(V_d \, \Phi_d \mid (V_r \, \Phi_r) \, (\delta X \, \delta Y \, \delta\vartheta) \, [I_{V_d}^{\delta X} = 1] \quad (9)$$
$$[I_{V_d}^{V_r} = 1] \, [I_{\Phi_d}^{\delta Y} = 1][I_{\Phi_d}^{\delta\vartheta} = 1] \, [I_{\Phi_d}^{\Phi_r} = 1])$$

In the preceding joint distribution, all the diagnosed variables are assumed to be independent, and to have uniform distributions. All the information concerning the relation between them will be encoded in the distribution over diagnosis variables. In order to define this

distributions, we first define the function $d_\sigma(x,y)$ as a Mahalanobis distance between x and y:

$$d_\sigma(x,y) = e^{-\frac{2}{2}\left(\frac{x-y}{\sigma}\right)^2}$$

Then, for two variables A and B, we define

$$P([I_A^B = 1] \mid AB) = d_{S(A,B)}(A, f(B)).$$

Let's see how preceding functions S and f are defined in specific cases.

a) Proportional compensation of errors: In the case of $I_{V_d}^{\delta X}$, we set $f(\delta X) = \alpha.\delta X$ and

$$S(V_d, \delta X) = \max((1 - \beta.\delta X)\sigma_{\max}, \sigma_{\min}).$$

Expression of f implies that the maximum of $P(I_{V_d}^{\delta X} \mid V_d\ \delta X)$ will be for a value of V_d proportional to the error δX. Expression of S defines the constraint level associated to this speed: the bigger the error, the more confident we are that a saturated proportional correction will work, so the smaller σ.

The basic behavior resulting from this definition is that when the robot is behind it nominal position, it will move forward to reduce its error, the bigger its error, the faster.

For $I_{\Phi_d}^{\delta Y}$, we use a similar proportional scheme. Its basic meaning is that when the robot has a lateral error, it has to steer, left or right, depending on the sign of this error. Again, the bigger the error, the more confident we are that we have to steer.

Finally, the same apply for $I_{\Phi_d}^{\delta \theta}$, except that the steering direction depends not only of the orientation error, but also of the movement direction V_d.

b) Using planned controls: In the path planning stage, the trajectory was defined as a set of nominal position, associated with planned speed and steering angle. They have to be accounted for when error is small.

Let's consider first $I_{V_d}^{V_r}$. We set f and S as follows: $f(V_r) = V_r$ and $S(V_d, V_r) = \sigma_{V_r} \in [\sigma_{\min}, \sigma_{\max}]$, rather close to $\sigma_{\max}$. By this way, planned speed is used as a indication to the trajectory following system. The distribution over $I_{\Phi_d}^{\Phi_r}$ is defined using the same reasoning.

C. Results

Fig. 11 illustrates the basic behavior of our trajectory tracking behavior. In both graphs, desired command will maximize either $P(V \mid \delta X\ V_c)$ or $P(\Phi \mid \delta Y\ \delta\theta\ \Phi_c)$. Since curve $P(V \mid \delta X\ V_c)$ is closer to $P(V \mid \delta X)$ than to $P(V \mid V_c)$, longitudinal error (δX) has much more influence that reference command on the vehicle speed. In the same manner, steering angle is a trade-off between what should be done to correct lateral error (δY) and orientation error ($\delta\theta$), lightly influenced by reference steering angle.

Fig. 12 shows the collaboration of obstacle avoidance and trajectory following on a simulated example. Planned trajectory passes through an obstacle which was not
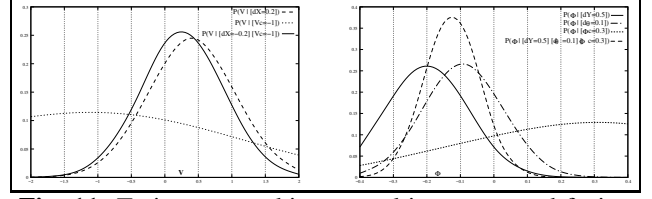

Fig. 11: Trajectory tracking : resulting command fusion

present at map building time. Obstacle avoidance modifies controls in order to grant security. When errors with respect to nominal trajectory is too big, our control law selector switch to the trajectory tracking behavior. Here it is a big longitudinal error, due to obstacle avoidance slowing down the vehicle, which trigger the switching.
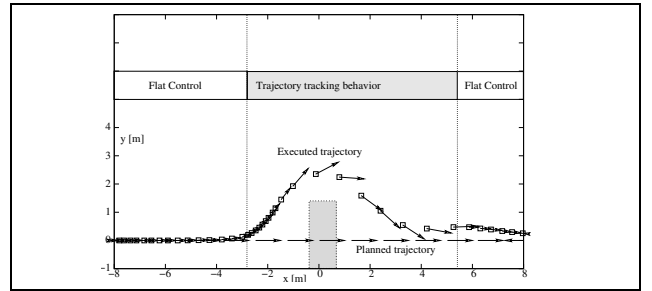

Fig. 12: Collaboration of trajectory tracking and obstacle avoidance on a simulated example

D. Discussion

Using the multiplexed control laws we managed to integrate, in the same control loop, our flat control, accurate but sensible to perturbation, with our proportional control law, inaccurate but robust to perturbation. By this way we obtained a system capable of tracking trajectory generated by our path planner while accounting for dynamic object in the environment.

Note that obstacle avoidance as implemented here has many similarities with a potential field obstacle avoidance system. So, it should not be expected to be safe in any dynamic environment[5]. Nevertheless, we found it safe in a moderately dynamic environment, such as our car park with maneuvering cars and moving pedestrians.

Finally, when the robot has gone too far from reference trajectory, or when reactive obstacle avoidance can not find suitable controls anymore, it may be necessary to re-plan a new trajectory to the goal. This has not been implemented on the robot yet, but it is quite sure that it will not make neither technical nor scientific problem arise.

V. EXPERIMENTAL SETUP

We tested the integration of these essential autonomy capacities in our experimental platform the Cycab robot.

[5]Interested reader may refer to [9] for solutions to these problems.
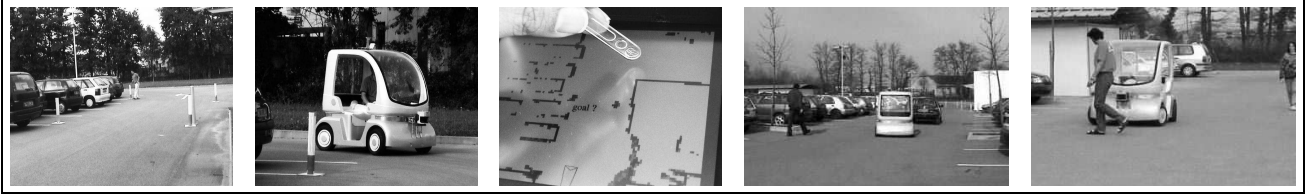
Fig. 13: An experimental setting showing from left to right: The arbitrary placing of the landmarks; the manual driving phase for landmark and obstacle map-building; the obstacle map generated together with the current position of the robot as seen on the LCD display; the capture of the goal position given by the user by means of the touch-screen; the execution of the found trajectory among pedestrians and unaccounted for vehicles.

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^{TM}* 233MHz running a RedHat^{TM} Linux system. All programs were written in C/C++ language.

During the experiments 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 $140ms$[6]; therefore the control system relies momentarily in odometry[6] readings.

Fig. 13 is a set of pictures showing a complete application integrating the stages described throughout the paper.

VI. DISCUSSION & CONCLUSIONS

In this paper, we presented our new steps toward the autonomy of a bi-steerable car. The integration of localization, map building, trajectory planning and execution in a moderately dynamic environment was discussed. Control law using the CyCab flatness property was found to be insufficient for trajectory tracking among moving pedestrians.

Even if this integration was successful and provides satisfactory results, we are convinced that a reactive behavior cannot be sufficient for the autonomy of vehicle in a real urban environment. For this reason, we are working on the perception and identification of road users (pedestrians, cars, bikes or trucks). By this way, we will be able to predict future movement of "obstacles" and to react accordingly, in a *smarter* way than the simple scheme proposed in this paper.

VII. REFERENCES

[1] 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.

[2] Carlos Canudas de Wit, Bruno Siciliano, and George Bastin Eds. *Theory of Robot Control*. Springer-Verlag, 1996.

[3] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer, Special Issue on Autonomous Intelligent Machines*, June 1989.

[4] M. Fliess, J. Lévine, P. Martin, and P. Rouchon. Design of trajectory stabilizing feedback for driftless flat systems. In *Proc. of the European Control Conference*, pages 1882–1887, Rome, Italy, september 1995.

[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 following. In *2003 American Control Conference*. To appear.

[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. Towards real-time global motion planning in a dynamic environment using the NLVO concept. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Lausanne (CH), September-October 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] C. Pradalier, F. Colas, and P. Bessiere. Expressing bayesian fusion as a product of distributions: Applications in robotics. In *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, 2003.

[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] C. Samson and K. Ait-Abderrahim. Feedback stabilization of a nonholonomic wheeled mobile robot. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1242–1246, Osaka (JP), November 1991.

[14] 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.

---

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