The CyCab: a car-like robot navigating autonomously and safely among pedestrians

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

The *CyCab*: a Car-Like Robot Navigating Autonomously and Safely Among Pedestrians

Cédric Pradalier, Jorge Hermosillo, Carla Koike,
Christophe Braillon, Pierre Bessière, Christian Laugier

firstname.lastname@inrialpes.fr
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 manoeuvrability 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 localisation and environment modelling, motion planning and motion execution amidst moderately 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.

Key words: Car-like robot, navigation, path planning, obstacle avoidance, autonomous navigation.

1 Introduction

The development of new Intelligent Transportation Systems (ITS), more practical, safe and accounting for environmental concerns, is a technological issue of highly urbanised societies today [18]. 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, Université 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 localisation and environment modelling, 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 localisation 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.

2 Localisation, Environment modelling, Motion planning and execution

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

2.1 Map-building and Localisation

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 $10000m^2$). For localisation 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

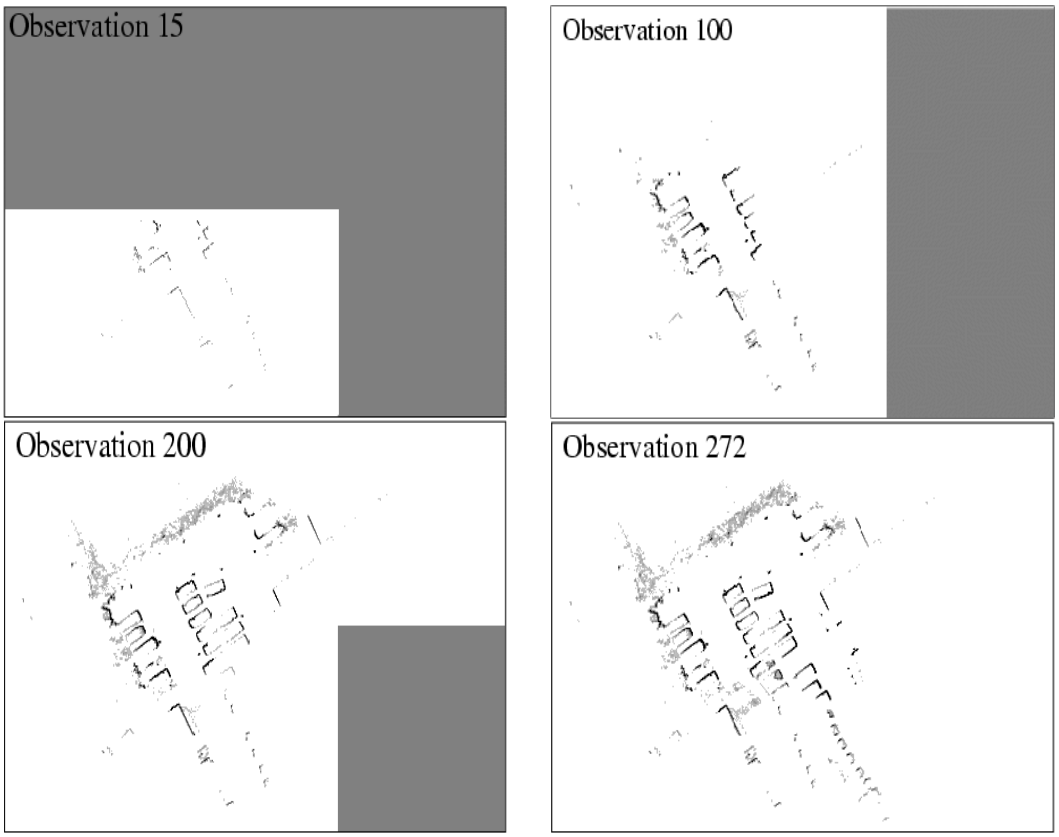

Fig. 1. Obstacle map evolution: Experimental images during the obstacle map-building phase. The vehicle is driven within the car-park area as long as needed. Simultaneously, the laser range sensor is used to detect the landmarks to build-up the localisation map.

accurately, and they should be identified with a reasonable computation effort. Fig. 2 shows our robot, its sensor and the landmarks : cylinder covered with reflector sheets, specially designed for our Sick laser range finder.

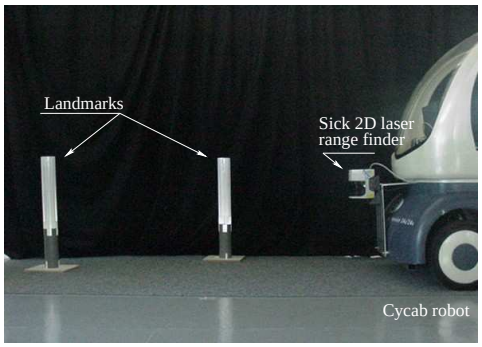

Fig. 2. Cycab robot and its landmarks for localization

Moreover, in order to keep flexibility, we wanted to be able to equip the environment with non permanent beacons. For this reason, 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 [21] for reference, and [24] for implementation details). It consists in building a map of features uncorrelated with the robot state. Such features are, for instance, the distance between landmarks or 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 worst case accuracy of our localisation system to the following value: about 10 centimetres in position and 2 degrees in orientation. We refer the reader to [24] for more details about the way we evaluate these values.

2.2 The Obstacle Map

The previous method localises 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 a kind of simplified occupancy grid[8] on the environment. This structure gives us informations correlated with the probability that a given place is the boundary of an obstacle.

Both maps are built online, in real-time, by the robot during the construction phase. Fig. 1 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.

2.3 Motion Planning Amidst Static Obstacles

The Motion Planner adopted for the CyCab was presented in [26]. 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 non-holonomic 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 non-holonomic motion planning is to find a steering method accounting for the kinematics of the robot. One way of designing steering methods for a

[1] Simultaneous Localisation And Mapping

[2] i.e. Verifying the topological property as explained in [26].

non-holonomic system is to use its ***flatness*** property [10] allowing also for feedback linearisation of the nonlinear system (this is discussed in section 2.6). This is what we did for the general BiS-car for which a flat output—or linearising output—was given in [26].

2.4 Steering a BiS-car

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

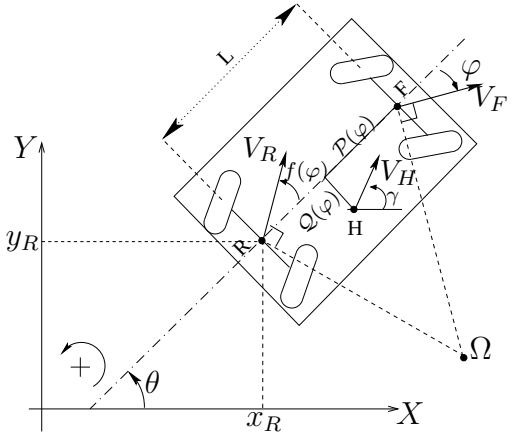

Fig. 3. Cycab robot, its landmarks and its kinematics model showing the coordinates of the flat output (point H) with respect to the reference frame of the robot placed at point F. In our case we have that $(x_{\scriptscriptstyle F}, y_{\scriptscriptstyle F}, \theta, \varphi)$ is the state of the robot.

The striking advantage of planning a path in the flat space is that we only need to parameterise a 2-dimensional curve whose points and derivatives define everywhere the current n-dimensional state[3] of the robot (in the case of the BiS-car $n = 4$). The main characteristic of such a curve is its curvature κ from which the steering angle can be computed.

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

2.5 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.

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

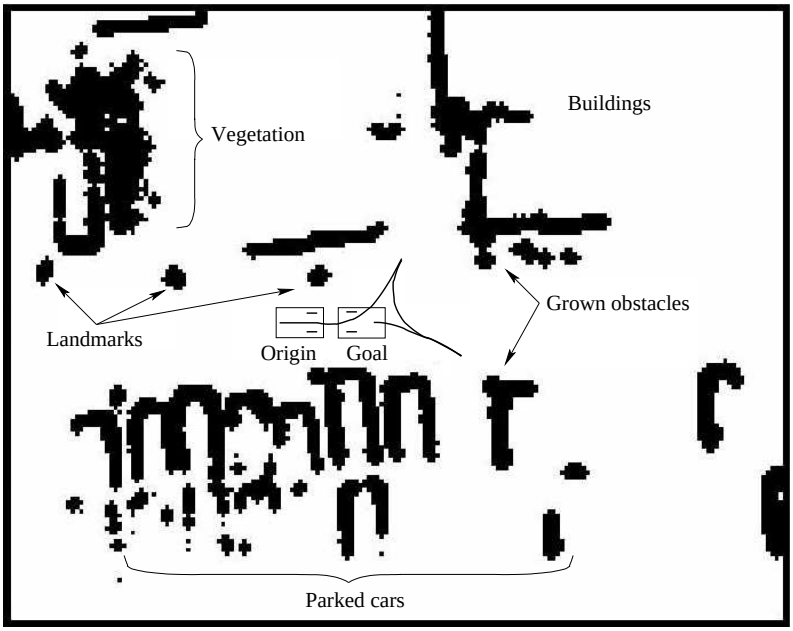

Fig. 4. Path computed by the motion planner using a real obstacle map. The obstacles are grown as well as the robot before computing the path.

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 parameterisation of the path [16] and the user is requested to accept to start the execution of the trajectory.

2.6 Trajectory tracking using flatness

It is well known that a non-holonomic system cannot be stabilised using only smooth state static feedbacks [6]. Ever since then, time-varying feedbacks [25] and dynamic feedbacks have been successfully used in particular for the canonical tractor-trailer and car-like robots [9].

Flat systems are feedback linearisable by means of a restricted class of dynamic feedback called *endogenous* [10]. The interest is that we are able to use state-of-the-art linear control techniques to stabilise the system. We present here results coming from recent work on feedback linearisation 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:

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

where $\mathcal{P}(\varphi)$ and $\mathcal{Q}(\varphi)$ are coordinate functions relative to the robot's reference frame (see [26] 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 linearising output, we found an expression giving the flat output dynamics with respect to a more convenient reference frame placed at the middle of the front axle of the robot (point F) and having orientation $\gamma = [\theta + \beta(\varphi)] \pm \pi$ where the function $\beta(\varphi)$ is the characteristic angle of the velocity vector of the flat output.

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 [14]:

$$\begin{aligned} \frac{\partial H}{\partial t} &= v_H \, \vec{u}_\gamma \\ v_H &= v_F[\cos(\varphi - \beta - \pi) - \mathcal{Q}\mathcal{F}] + \omega_\varphi[\frac{\partial \mathcal{P}}{\partial \varphi} - \frac{\partial \beta}{\partial \varphi}\mathcal{Q}] \\ \mathcal{F}(\varphi) &= \frac{\sin(\varphi - f(\varphi))}{L\cos(f(\varphi))} \end{aligned} \tag{1}$$

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).

From expression (1) 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 stabilising 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 [10]. The endogenous dynamic feedback that linearises the general bi-steerable system is presented in [14]. Then, from linear control theory, it can be shown that the closed-loop control stabilising the reference trajectory $\mathbf{y}^*$ has the following form :

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

Where $(.)^{(p)}$ stands for the total derivative of order p. See [7] for details.

3 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.

3.1 State of the art on reactive trajectory tracking

Most of the approaches for obstacle avoidance are local([11,5]), that is they do not try to model the whole environment. They goal is rather to use sensor measures to deduce secure commands. Being simpler and less computationnaly intensive, they seem more appropriate to fast reactions in a non-static environment. On the other hand, we can not expect optimal solutions from a local method. It is possible that some peculiar obstacle configuration create a dead-end from which the robot cannot escape with obstacle avoidance only.

3.1.1 Potential fields

The general idea of potential fields methods, proposed initially by O. Khatib in 1986, is to build a function representing both the navigation goals and the need for obstacle avoidance. This function is built so has to decrease when going closer to the goal and to increase near obstacles. Then, the navigation problems is reduced to an optimisation problem, that is, to find the commands that brings the robot to the global minimum of the function. This later can be defined with respect to the goal and the obstacles but other constraints can also be added therein.

Numerous extensions to the potential fields have been proposed since 1986. Among others, we can cite the Virtual Force Fields [3], the Vector Field Histograms [4] and their extensions VFH+[28] and VFH*[29]. Basically, these methods try to find the best path to the goal among the secure ones.

3.1.2 Steering Angle Field (SAF)

The SAF method, proposed by *Feiten et al.* in 1994, use obstacles to constrain steering angle in a continuous domain. Simultaneously, speed control is an iterative negociation process between the high-level driving module and the local obstacle-avoidance module.

One of the first extension to this method was published in [27]. It express the collision avoidance problem as an optimisation problem in the robot controls space

(linear and rotational speeds).

3.1.3 Dynamic Window

The Dynamic Window approach[11] propose to avoid obstacles by exploring command space in order to maximise an objective function. This later accounts for the progression toward the goal, the position of closer obstacles and current robot controls. Being directly derived from the robot dynamic, this methode is particularly well adapted to high speed movements.

The computational cost of the optimization process is reduced using the dynamic caracteristics of the robot (bounded linear and angular acceleration) so as to reduce the searched space. This kind of constraints are called *Hard Constraints* since the must be respected. Conversely, when the objective function includes preferences on the robot movement, we call the resulting constraints *Soft Constraints*.

3.1.4 Dynamic environments and Velocity Obstacles

In the specific case of moving obstacles, special methods have been proposed[17,2] using the *Velocity Obstacle* notion. Basically, this notion consist in projecting perceived obstacles and their expected movement in the space of secure commands. So, each mobile object generates a set of obstacles in the command space. These obstacles represent the commands that will bring to a collision in the future.

In the general case, obstacle movement parameters are not known *a priori*, so they have to be deduced from sensor data. Obstacle avoidance controls are then computed in reaction to theses previsions. Currently, it is still quite difficult to get reliable previsions of the obstacles future trajectory. Consequently, these obstacle avoidance methods are not appliable in real situations yet.

3.1.5 Obstacle avoidance and trajectory following

When we want to perform obstacle avoidance manoeuvres while following a trajectory, a specific problem appear. On our non-holonomous robot, the path planning stage took into account the kinematic of the robot and planned a feasible path. When the reactive obstacle avoidance generates commands, the vehicle leaves its planned trajectory. Then, we cannot be sure anymore that the initial objective of the trajectory is still reachable.

A solution to this problem was proposed in [20]. This method tries to deform the global trajectory in order to avoid the obstacle, respect the kinematic constraints and ensure that the final goal is still reachable. Even if theoretically very interesting, this obstacle avoidance scheme is still difficult to apply in real situations

due to it computational complexities, especially on an autonomous car. In our experiments[20], the vehicle had to stop for several minutes in order to perform the trajectory deformation

3.2 Objectives

After all these results on obstacle avoidance, it seems obvious that our goal is not to propose a new solution to this problem. It has been shown[19,1] that probalities and bayesian inference are appropriate tools to deal with real world uncertainty and to model reactive behaviors. We this in mind, we wanted to think about the expression of the obstacle avoidance problem as a bayesian inference problem. Consequently, the originality of our approach is mainly its expression and the semantic we can express with it.

3.3 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. In order to limit the volume of the data we manipulate, we summarised the sensor output as 8 values : the distances to the nearest obstacle in a $\pi/8$ angular sector(see Fig. 5). We will call $D_k, k = 1 \dots 8$ 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 it 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.

3.4 Sub-models definition

Given the distance D_i measured in an angular sector, we want to express a command to apply that is safe while tracking desired command. Nevertheless, since this sector only has limited information about robot surrounding, we choose to express the following conservative semantic: tracking the desired command should be a soft constraint whereas an obstacle avoidance command should be a hard constraint, the closer the obstacle, the harder the constraint.

We express this semantic using a probability distribution over the commands to apply (V, Φ) knowing the desired commands and the distance D_i measured in this

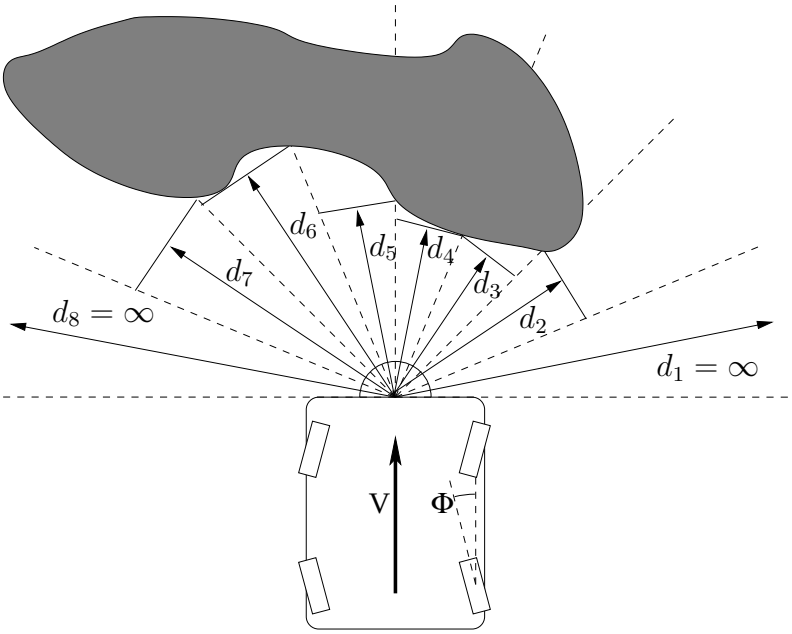

Fig. 5. Obstacle avoidance: situation

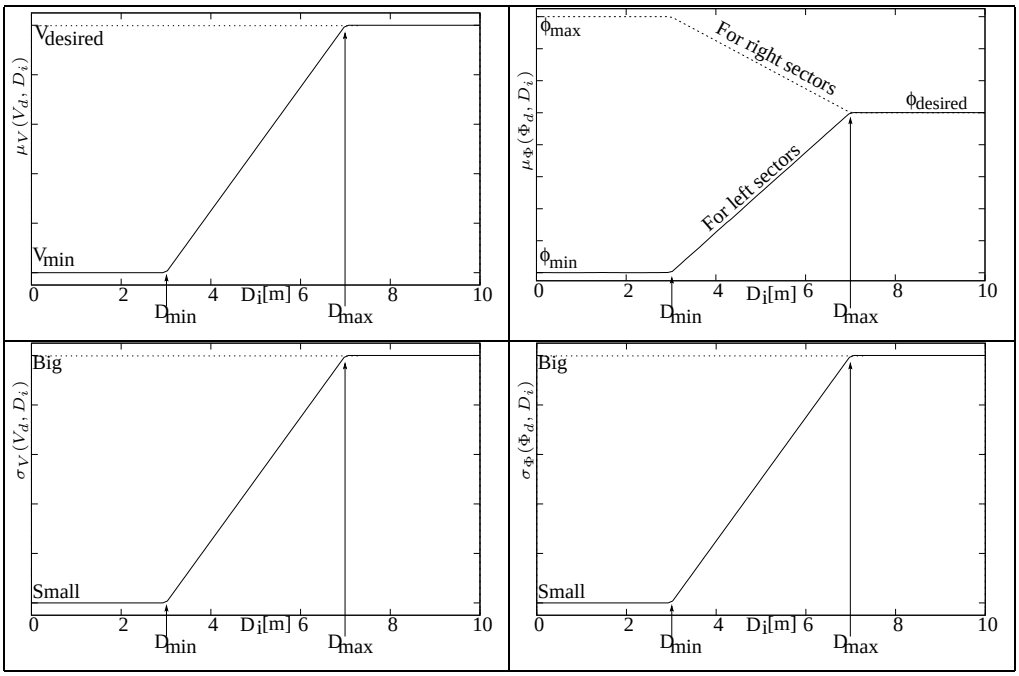

Fig. 6. Evolution of mean and standard deviation of $P_i(V \mid V_d D_i)$ and $P_i(\Phi \mid \Phi_d D_i)$ according to distance measured

sector:

$$P_i(V\Phi \mid V_d\Phi_d D_i) = P_i(V \mid V_d D_i) P_i(\Phi \mid \Phi_d D_i) \tag{3}$$

where $P_i(V \mid V_d D_i)$ and $P_i(\Phi \mid \Phi_d D_i)$ are Gaussian distributions respectively centred 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 illustrated in Fig. 6. Example of resulting distributions are shown in Fig. 7.

There is two specific aspects to notice in Fig. 6 and 7. 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.

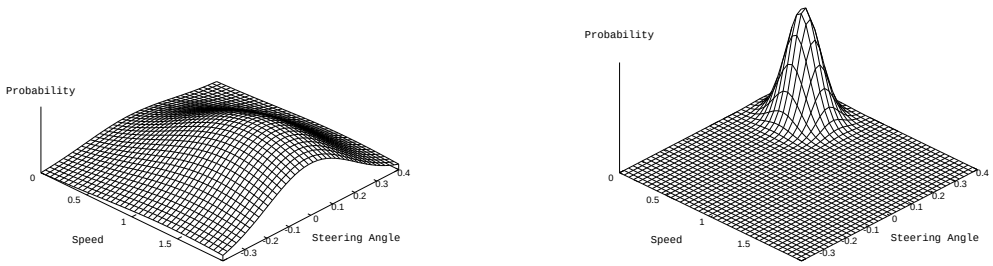

Fig. 7. Shape of $P_i(V\Phi \mid V_d\Phi_d D_i)$ for far and close obstacles

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.

3.5 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:

$$\begin{aligned} P(V\ \Phi\ V_d\ \Phi_d\ D_1 \ldots D_8\ S) = {} & P(D_1 \ldots D_8)\ P(V_d\ \Phi_d) \\ & P(S)\ P(V\ \Phi\ \mid\ V_d\ \Phi_d\ D_1 \ldots D_8\ S) \end{aligned} \tag{4}$$

where variable $S \in [1 \ldots 8]$ express which sector is considered. $P(D_1 \ldots D_8)$ and $P(V_d\Phi_d)$ are unknown distribution [4]. As there is no need to favour a specific sub-model, we define $P(S)$ as a uniform distribution. The semantic of S will be em-

[4] Actually, as we know we will not need them in future computation, we don't have to specify them.

phasised by the definition of $P(V\Phi \mid V_d\Phi_d D_1 \ldots D_8 S)$:

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

In this equation, we can see that the variable S acts as model selector: given its value i, the distribution over the commands will be computed by the sub-model i, taking into account only distance D_i.

Using equation 4, we can now express the distribution we are really interested in, that is the distribution over the commands accounting for all the distances but not variable S:

$$P(V\ \Phi\ \mid\ V_d\ \Phi_d\ D_1 \ldots D_8) = \sum_S (P(S)P(V\ \Phi\ \mid\ V_d\ \Phi_d\ D_1\ \ldots\ D_8\ S)) \quad (5)$$

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 quasi-flat distribution, and those who see perilous obstacles will add a peaky distribution, hence having more influence (see Fig. 8). Finally the command really executed by the robot is the one which maximise $P(V\ \Phi\ \mid\ V_d\ \Phi_d\ D_1 \ldots D_8)$ (eq. 5).

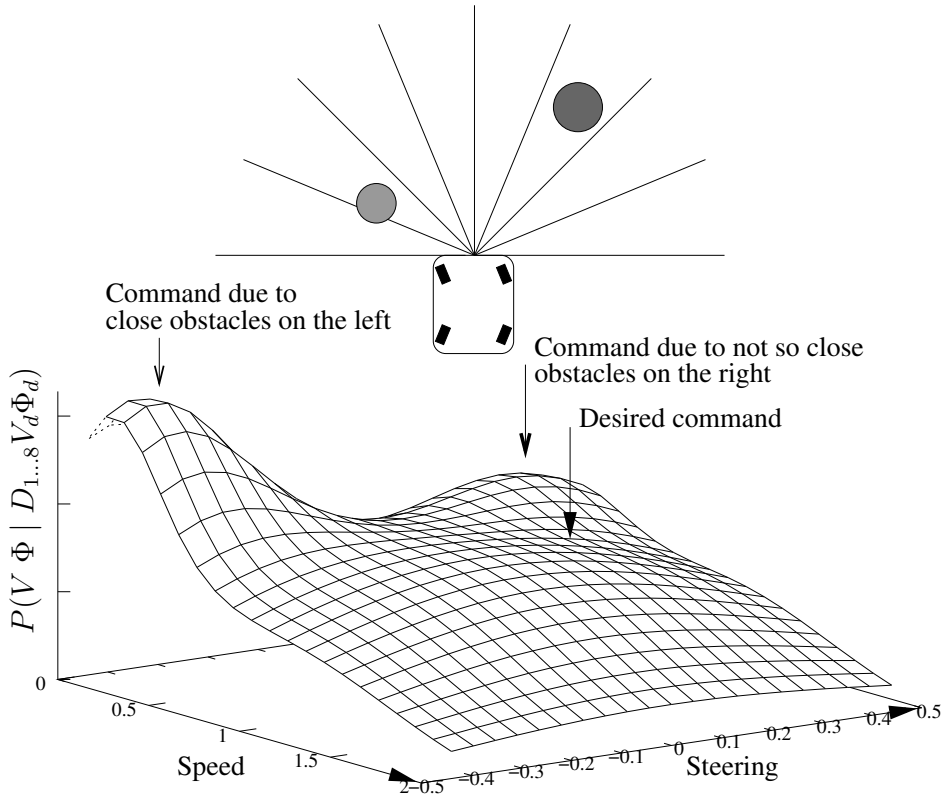

Fig. 8. Probability distribution over speed and steering, resulting from the obstacle avoidance system.

Fig. 9 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, first obstacle avoidance module bends the trajectory in order to avoid the walls, and second, when there is no danger of collisions, desired commands are applied exactly as requested.

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.

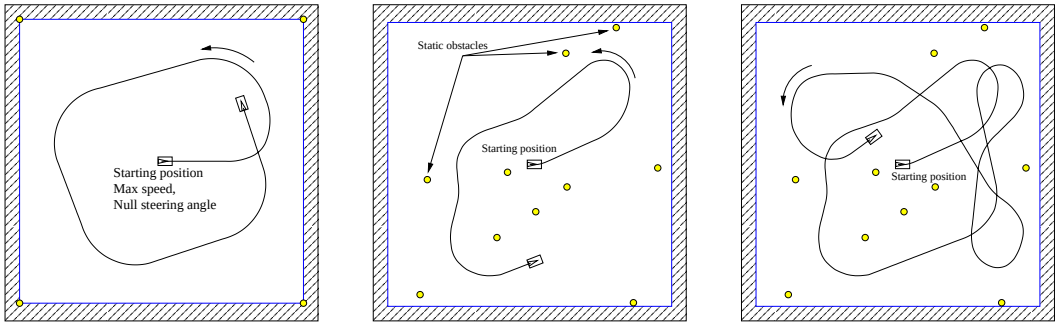

Fig. 9. Robot trajectory while driven manually with constant desired steering angle

3.7 Relation to fuzzy logic approaches

The design of our obstacle avoidance modules may remind some readers of a fuzzy logic controller[15,22,12]. It is rather difficult to say that one approach is better than the other. Both fuzzy logic and bayesian inference view themselves as extension of classical logic. Furthermore, both methods will deal with the same kind of problems, providing the same kind of solutions. Some will prefer the great freedom of fuzzy logic modelling and others will prefer to rely on the strong mathematical background behind bayesian inference.

As far as we can see, the choice between fuzzy logic and bayesian inference is rather an personal choice, similar to the choice of a programming language: it has more consequences on the way we express our solution than on the solution itself. To extend the analogy, one might relate fuzzy logic to the C language whereas Bayesian inference would be closer to Ada.

4 Trajectory tracking with obstacle avoidance

The method presented in the previous section provides 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. In this section will show how we integrate trajectory tracking and obstacle avoidance.

While following the 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 or diversions 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 behaviour to adapt smoothly our control system to the perturbations induced by obstacle avoidance.

4.1 Multiplexed trajectory tracking

4.1.1 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. 2), it is a massively integrating system. For this reason, a constant perturbation such as immobilisation 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 strongly breaks the dynamic rules around which the flat control law was built. So, there is no surprise in its failure.

4.1.2 Probabilistic control law

In order to deal with the situations that flat control law cannot manage, we designed a trajectory tracking behaviour (*TTB*) based again on probabilistic reasoning (section 4.2). As this behaviour has many similarities with a weighted sum of proportional control laws, we do not expect it to be sufficient to stabilise 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 behaviour 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, TTB takes control, and try to bring it back to

flat control law's convergence domain. When it enters this domain, flat control law is reinitialised and starts accurate trajectory tracking(this is illustrated in fig. 10).

4.1.3 Time control

Path resulting from path planning (section 2.3) 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 comes too far from its nominal position, we freeze the nominal position, and we use the TTB to reenter the domain where nominal position can be unfrozen.

The global system is illustrated by Fig. 10: 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.

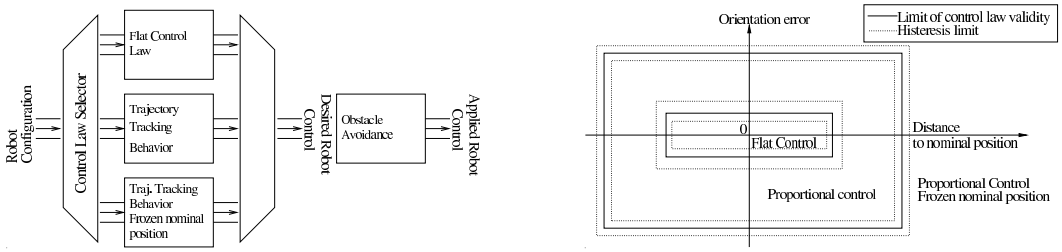

Fig. 10. Basic diagram of the control law selector mechanism and validity domains of the control laws

4.2 Trajectory tracking behaviour

Our trajectory tracking behaviour was built as a probabilistic reasoning, in a way similar to the obstacle avoidance presented above (section 3). Functionnaly, it is very similar to a fuzzy control scheme as presented in [15] and illustrated in [12].

To specify our module, we use a mechanism of fusion with diagnosis[23]. 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 $\delta\theta$ with respect to the nominal position. Fig. 11 illustrates theses variables.

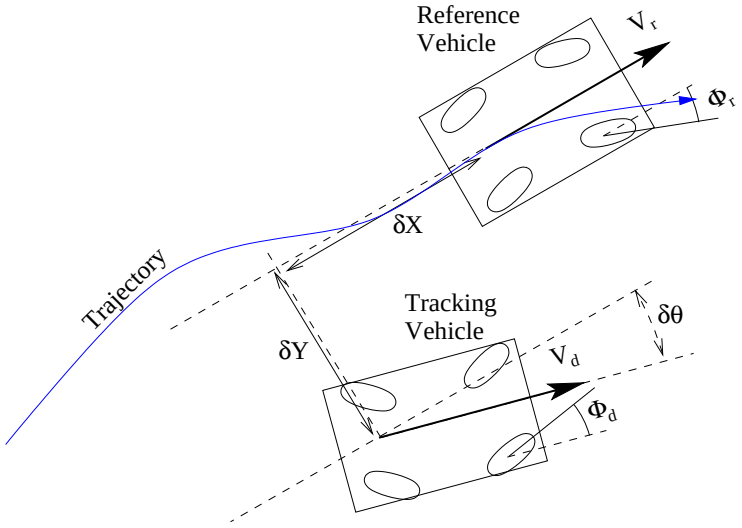

Fig. 11. Variables involved in trajectory tracking behaviour

In addition to the preceding variables, we will add five diagnosis variables $I_{V_d}^{\delta X}$, $I_{V_d}^{V_r}$, $I_{\Phi_d}^{\delta Y}$, $I_{\Phi_d}^{\delta\theta}$ and $I_{\Phi_d}^{\Phi_r}$. Variables linked to an error variable $(\delta X, \delta Y, \delta\theta)$ will diagnose if a given command helps correcting this error. Variables linked to reference commands evaluate if a command is similar to the reference one.

All these variables describe the relation between their diagnosed variables in the following joint distribution:

$$\begin{aligned} P(V_d\,\Phi_d\,V_r\,\Phi_r\,\delta X\,\delta Y\,\delta\theta\;I_{V_d}^{\delta X}\,I_{V_d}^{V_r}\,I_{\Phi_d}^{\delta Y}\,I_{\Phi_d}^{\delta\theta}\,I_{\Phi_d}^{\Phi_r}) = \\ P(V_d\;\Phi_d)\;P(V_r\;\Phi_r)\;P(\delta X\;\delta Y\;\delta\theta) \\ P(I_{V_d}^{\delta X}\mid V_d\;\delta X)\;P(I_{V_d}^{V_r}\mid V_d\;V_r) \\ P(I_{\Phi_d}^{\delta Y}\mid \Phi_d\;\delta Y)\;P(I_{\Phi_d}^{\delta\theta}\mid \Phi_d\;\delta\theta\;V_d)\;P(I_{\Phi_d}^{\Phi_r}\mid \Phi_d\;\Phi_r) \end{aligned} \tag{6}$$

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

$$\begin{aligned} P(V_d\,\Phi_d\,|\,(V_r\;\Phi_r)\;(\delta X\;\delta Y\;\delta\theta) \\ [I_{V_d}^{\delta X}=1]\;[I_{V_d}^{V_r}=1]\;[I_{\Phi_d}^{\delta Y}=1]\;[I_{\Phi_d}^{\delta\theta}=1]\;[I_{\Phi_d}^{\Phi_r}=1]) \end{aligned} \tag{7}$$

Basically, this equation expresses the fact that we are looking for the most likely commands in order to correct tracking error while accounting for reference commands. Having all the diagnosis variables set to one enforces this semantic.

In the preceding joint distribution (eq. 6), 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.

4.2.1 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 proportional correction will work, so the smaller σ.

The basic behaviour 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 and with more confidence that this is the good control to apply.

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.

4.2.2 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, especially 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.

4.3 Results

Fig. 12 illustrates the basic behaviour of our trajectory tracking behaviour. In both graphs, desired command will maximise 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 $P(V \mid V_c)$, we can observe that longitudinal error (δX) has much more influence than 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.

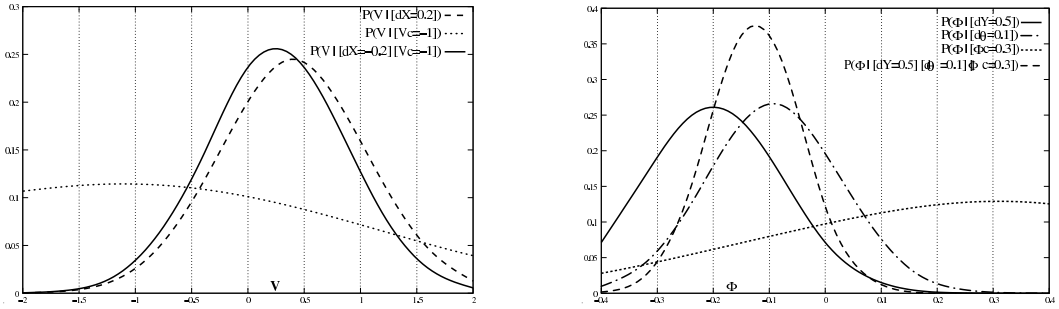

Fig. 12. Trajectory tracking : resulting command fusion

Fig. 13 shows the collaboration of obstacle avoidance and trajectory following on a simulated example. Planned trajectory passes through an obstacle which was not 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 behaviour. Here it is a big longitudinal error, due to obstacle avoidance slowing down the vehicle, which trigger the switching.

4.4 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 TTB, less accurate but robust to perturbations. By this way we obtained a system capable of tracking trajectory generated by our path planner while accounting for unexpected object in the environment.

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 this should not be considered neither a technical nor a scientific issue.

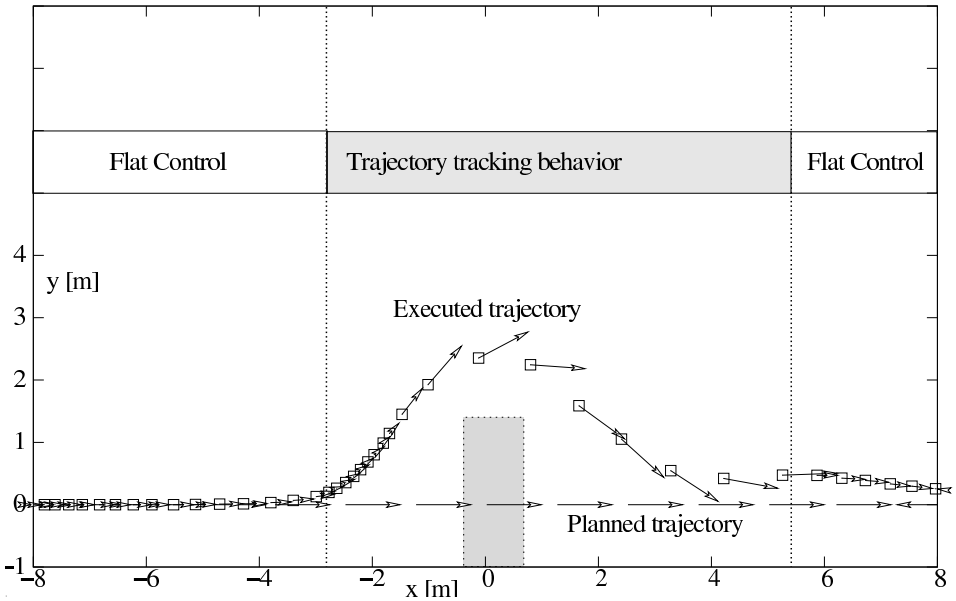

Fig. 13. Collaboration of trajectory tracking and obstacle avoidance on a simulated example

Fig. 14. 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 aggressive pedestrians.

5 Experimental setup

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 on-board the Cycab is a *Pentium II*^{TM} 233MHz running a 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 was limited to $140ms$ [5] ; therefore the control system has to rely momentarily in odometry[13] readings.

Fig. 14 is a set of pictures showing a complete application integrating the stages described throughout the paper.

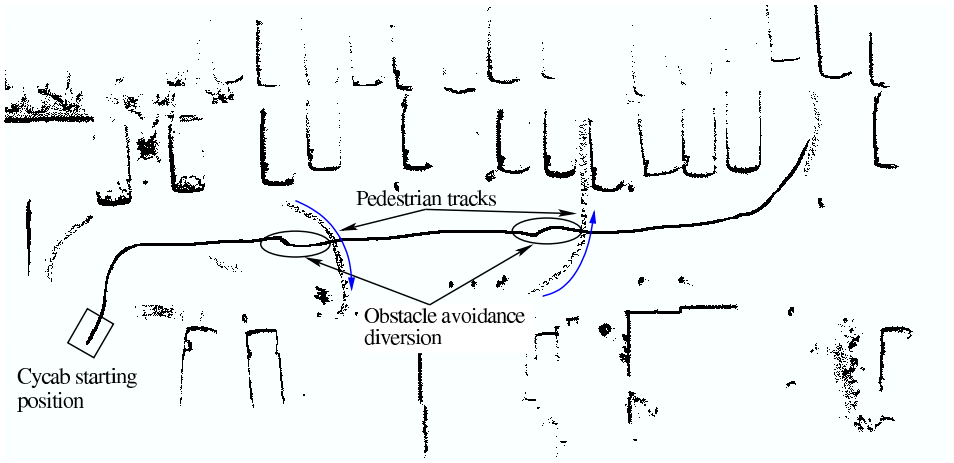

Fig. 15. Executed trajectory among static obstacles and moving pedestrians. Rear middle point (R in fig. 3) trajectory is drawn.

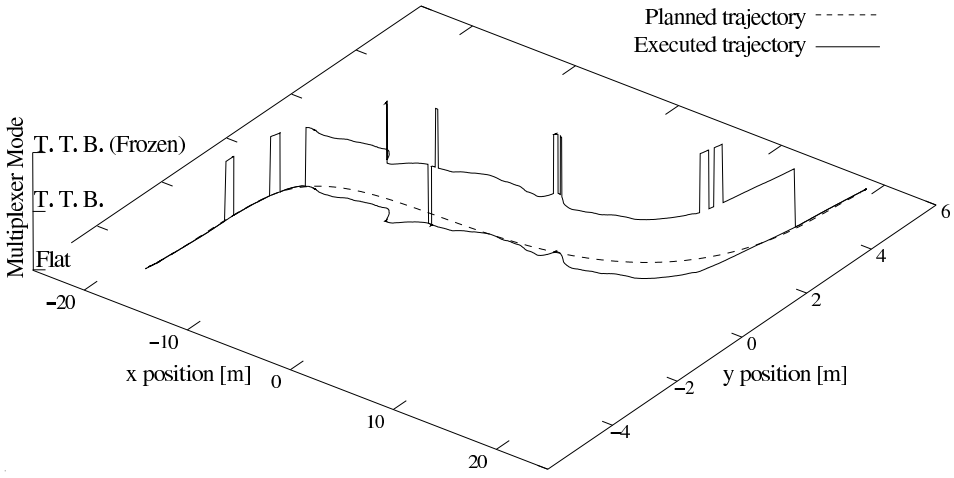

Fig. 16. Executed trajectory with respect to planned trajectory, and multiplexer mode.

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

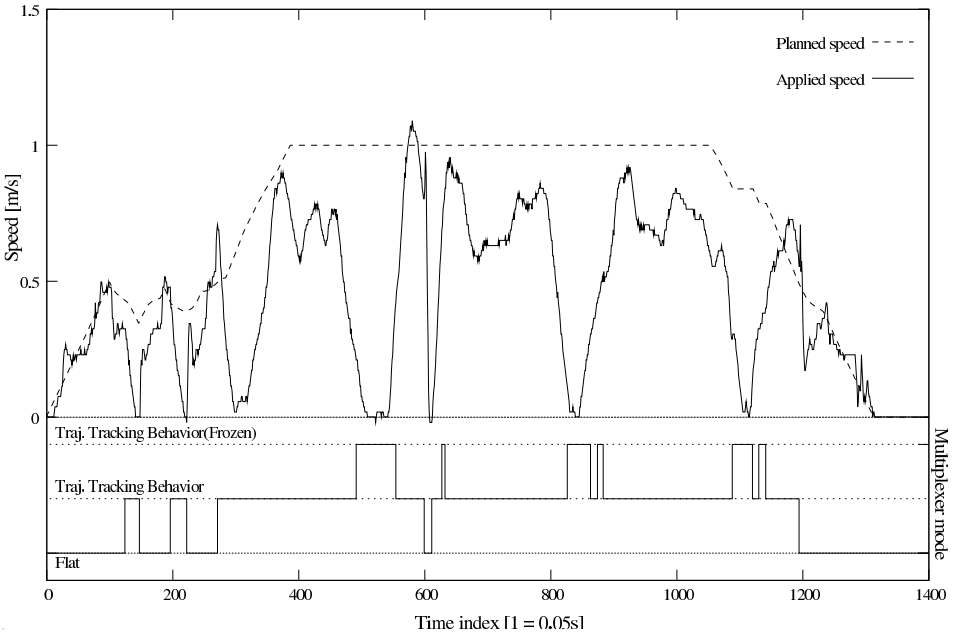

Fig. 17. Applied speeds with respect to planned speed, and multiplexer mode.

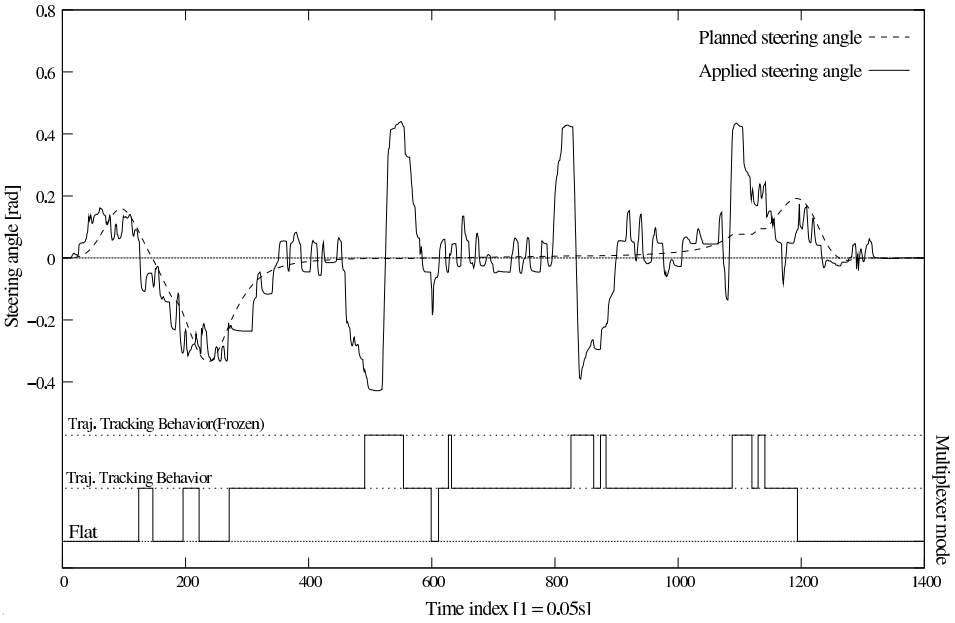

Fig. 18. Applied steering with respect to planned steering, and multiplexer mode.

Figs 15 to 18 illustrates how a planned trajectory is executed while avoiding moving pedestrians. In this environment, the control law using flatness could only be used at the beginning and at the end of the trajectory. On the remaining of the trajectory, speed and steering angle are adjusted in order to maintain security while keeping pace with the plan as much as possible.

6 Discussion & Conclusions

In this paper, we presented our new steps toward the autonomy of a bi-steerable car. The integration of localisation, 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 behaviour 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.

References

[1] P. Bessière and BIBA-INRIA Research Group. Survey: Probabilistic methodology and techniques for artefact conception and development. Technical Report RR-4730, INRIA, Grenoble, France, February 2003. http://www.inria.fr/rrrt/rr-4730.html.

[2] S. Blondin. Planification de mouvements pour véhicule automatisé en environnement partiellement connu. Mémoire de Diplôme d'Etudes Approfondies, Inst. Nat. Polytechnique de Grenoble, Grenoble (FR), June 2002.

[3] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, Sept/Oct 1989.

[4] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, June 1991.

[5] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Detroit, MI (US), May 1999.

[6] 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.

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

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

[9] 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.

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

[11] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33, March 1997.

[12] Th. Fraichard and Ph. Garnier. Fuzzy control to drive car-like vehicles. *Robotics and Autonomous Systems*, 34(1):1–22, December 2000.

[13] 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.

[14] 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.

[15] Lawrence A. Klein. *Sensor Data Fusion Concepts and Applications*. SPIE, 1993.

[16] 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.

[17] 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.

[18] 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.

[19] O. Lebeltel, P. Bessière, J. Diard, and E. Mazer. Bayesian robots progamming. *Autonomous Robots*, 2003. In Press.

[20] O. Lefebvre, F. Lamiraux, C. Pradalier, and Th. Fraichard. Obstacles avoidance for car-like robots. integration and experimentation on two robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, New Orleans, LA (US), April 2004.

[21] P. Newman. *On the structures and solution of simultaneous localization and mapping problem*. PhD thesis, Australian Center for Field Robotics,Sidney, 1999.

[22] G. Oriolo, G. Ulivi, and M. Vendittelli. *Applications of Fuzzy Logic: Towards High Machine Intelligent Quotient Systems*. Prentice-Hall, 1997.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] R. Simmons. The curvature-velocity method for local obstacle avoidance. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 3375–3382, Minneapolis, MN (US), April 1996.

[28] I. Ulrich and J. Borenstein. VFH+: Reliable obstacle avoidance for fast mobile robots. In *IEEE Int. Conf. Robotics Automation*, pages 1572–1577, Leuven, Belgium, May 1998.

[29] I. Ulrich and J. Borenstein. VFH*: Local obstacle avoidance with look-ahead verification. In *IEEE Int. Conf. Robotics Automation*, pages 2505–2511, San Francisco, CA, April 2000.