Autonoumous Robot, 6(2), 1–18 (1999)
© 1999 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

# Sensor-Based Control Architecture for a Car-Like Vehicle

C. LAUGIER, TH. FRAICHARD, PH. GARNIER,
I.E. PAROMTCHIK AND A. SCHEUER
*Inria* Rhône-Alpes, Zirst, 655 av. de l'Europe, 38330 Montbonnot Saint Martin, France*
[christian.laugier, thierry.fraichard]@inria.fr

*Received February 1998; Revised November 1998*

**Abstract.** This paper presents a control architecture endowing a car-like vehicle moving in a dynamic and partially known environment with autonomous motion capabilities. Like most recent control architectures for autonomous robot systems, it combines three functional components: a set of basic real-time skills, a reactive execution mechanism and a decision module. The main novelty of the architecture proposed lies in the introduction of a fourth component akin to a meta-level of skills: the *sensor-based manoeuvres*, *i.e.* general templates that encode high-level expert human knowledge and heuristics about how a specific motion task is to be performed. The concept of sensor-based manoeuvres permit to reduce the planning effort required to address a given motion task, thus improving the overall response-time of the system, while retaining the good properties of a skill-based architecture, *i.e.* robustness, flexibility and reactivity. The paper focuses on the *trajectory planning* function (which is an important part of the decision module) and two types of sensor-based manoeuvres, *trajectory following* and *parallel parking*, that have been implemented and successfully tested on a real automatic car-like vehicle placed in different situations.

**Keywords:** motion autonomy, control architecture, car-like vehicle.

## 1. Introduction

Autonomy in general and motion autonomy in particular has been a long standing issue in Robotics. In the late sixties-early seventies, Shakey (Nilsson, 1984) was one of the first robots able to move and perform simple tasks autonomously. Ever since, many authors have proposed control architectures to endow robot systems with various autonomous capabilities. Some of these architectures are reviewed in §7 and compared to the one presented in this paper. These approaches differ in several ways, however it is clear that the control structure of an autonomous robot placed in a dynamic and partially known environment must have both *deliberative* and *reactive* capabilities. In other words, the robot should be able to decide which actions to carry out according to its goal and current situation; it should also be able to take into account events (expected or not) in a timely manner.

The control architecture presented in this paper aims at meeting these two requirements. It is designed to endow a car-like vehicle moving on the road network with motion autonomy and was developed in the framework of the French Praxitèle programme aimed at the development of a

*Institut National de Recherche en Informatique et en Automatique.

new urban transportation system based on a fleet of electric vehicles with autonomous motion capabilities (Parent and Daviet, 1996). The road network is a complex environment, it is partially known and highly dynamic with moving obstacles (other vehicles, pedestrians, etc.) whose future behaviour is not known in advance. However the road network is a structured environment with motion rules (the highway code) and it is possible to take advantage of these features in order to design a control architecture that is efficient, robust and flexible.

The control architecture is presented in this paper as follows: in the next section, the rationale of the architecture and its main features are overviewed. It introduces the key concept of *sensor-based manoeuvres*, *i.e.* general templates that encode the knowledge of how a specific motion task is to be performed. The model of the car-like vehicle that is used throughout the paper is then described (§3). One important component of the architecture is the *trajectory planner* whose purpose is to determine the trajectory leading the vehicle to its goal. Trajectory planning for car-

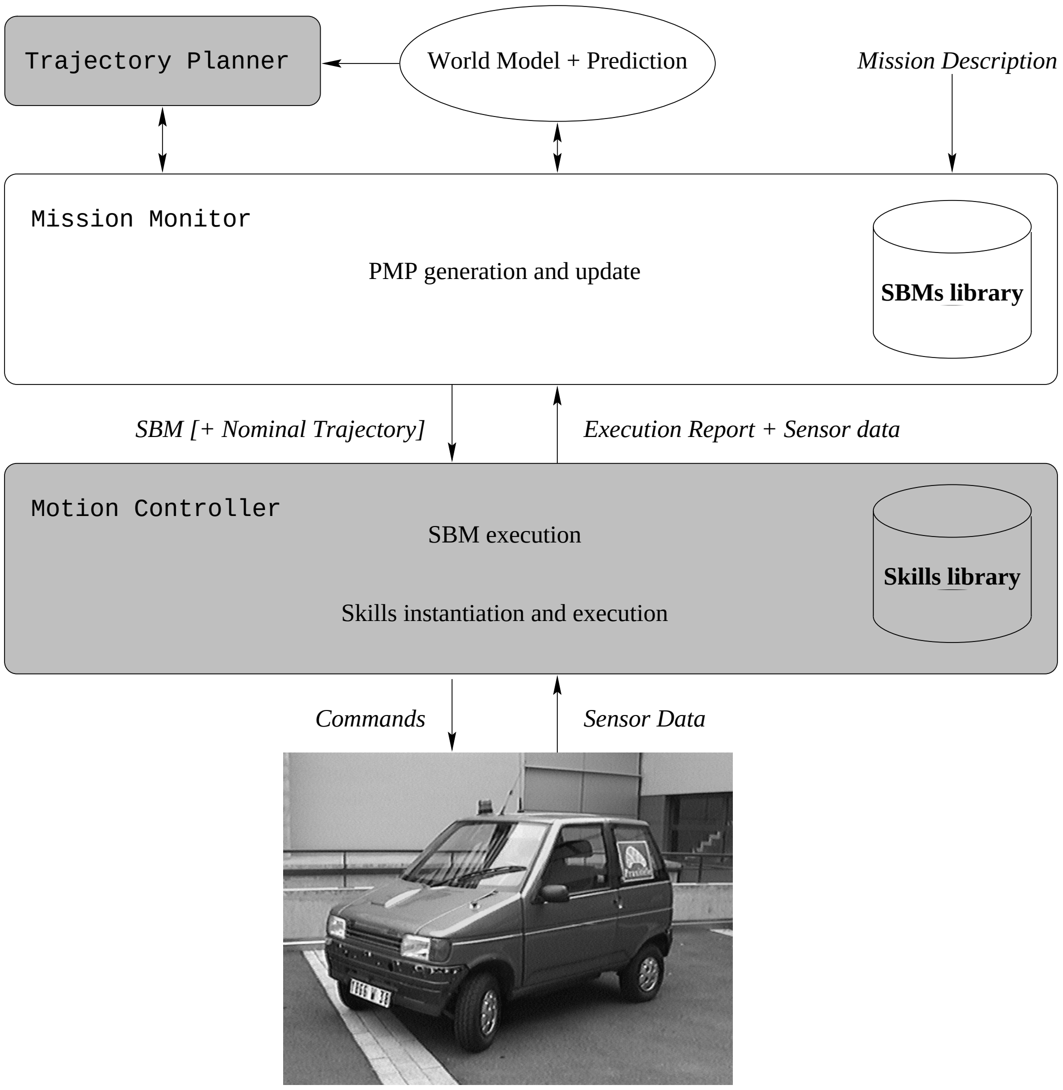

*Fig. 1.* The overall control architecture.

like vehicles in dynamic environments remains an open problem and a practical solution to this intricate problem is presented in §4. Afterwards the concept of sensor-based manoeuvres is explored in §5 and two types of manoeuvres are presented in detail. These two manoeuvres have been implemented and successfully tested on an experimental vehicle, the results of these experiments are finally presented in §6.

## 2. Overview of the Control Architecture

The control architecture is depicted in Fig. 1. It relies upon the concept of *sensor-based manoeuvres* (SBM) which is derived from the Artificial Intelligence concept of *script* (Rich and Knight, 1983). A script is a general template that encodes procedural knowledge of how a specific type of task is to be performed. A script is fitted to a specific task through the instantiation of variable parametres in the template; these parameters can come from a variety of sources (*a priori* knowledge, sensor data, output of other modules, etc.). Script parametres fill in the details of the script steps and permit to deal easily with the current task conditions.

The introduction of SBM was motivated by the observation that the kind of motion task that a vehicle has to perform can usually be described as a series of simple steps (a script). A SBM is a script, it combines *control* and *sensing skills.* Skills are elementary functions with real-time abilities: sensing skills are functions processing sensor data whereas control skills are control programs (open or closed loop) that generate the appropriate commands for the vehicle. Control skills may use data provided directly by the sensors or by the sensing skills.

The idea of combining basic real-time skills to build a plan in order to perform a given task can be found in other control architectures (*cf.* §7); they permit to obtain robust, flexible and reactive behaviours. SBMs can be seen as "meta-skills", their novelty is that they permit to encapsulate high-level expert human knowledge and heuristics about how to perform a specific motion task (*cf.* §5). Accordingly they permit to reduce the planning effort required to address a given motion task, thus improving the overall response-time of the system, while retaining the good properties of a skill-based architecture, *i.e.* robustness, flexibility and reactivity.

The control architecture features two main components, the *mission monitor* and the *motion controller*, that are described afterwards.

### *2.1. Mission Monitor*

When given a mission description, *e.g.* "go park at location $l$", the mission monitor (MN) generates a *parameterized motion plan* (PMP) which is a set of generic sensor-based manoeuvres (SBM) possibly completed with nominal trajectories. The SBMs are selected from a SBM library. A SBM may require a nominal trajectory (it is the case of the "Follow Trajectory" SBM). A nominal trajectory is a continuous time-ordered sequence of (position, velocity) of the vehicle that represents a theoretically safe and executable trajectory, *i.e.* a collision-free trajectory which satisfies the kinematic and dynamic constraints of the vehicle. Such trajectories are computed by the trajectory planner by using:

- an *a priori known* or *acquired model* of the vehicle environment,
- the current *sensor data*, *e.g.* position and velocity of the moving obstacles, and
- a *world prediction* that gives the most likely behaviours of the moving obstacles.

Trajectory planning is detailed in §4. The current SBM with its nominal trajectory is passed to the motion controller for its reactive execution.

### *2.2. Motion Controller*

The goal of the Motion Controller (MC) is to execute in a reactive way the current SBM of the PMP. For that purpose, the current SBM is instantiated according to the current execution context, *i.e.* the variable parametres of the SBM are set by using the *a priori* known or sensed information available at the time, *e.g.* road curvature, available lateral and longitudinal space, velocity and acceleration bounds, distance to an obstacle, etc. As mentioned above, a SBM combines control and sensing skills that are either control programs or sensor data processing functions. It is up to MC to control and coordinate the execution of the different skills required. The sequence of control skills that is executed for a given SBM is determined by the events detected by the sensor skills. When an event that cannot be handled by the current SBM happens, MC reports a failure to MN which updates PMP either by applying a replanning procedure (time permitting), or by selecting in real-time a SBM adapted to the new situation.

## 3. Model of the Vehicle

A car-like vehicle is modelled as a rigid body moving on the plane. It is supported by four wheels making point contact with the ground, it has two rear wheels and two directional front wheels. The model of a car-like vehicle that is used is depicted in Fig. 2. The configuration, *i.e.* the position and orientation of the vehicle, are characterized by the triple $q = (x, y, \theta)$ where $x = x(t)$ and $y = y(t)$ are the coordinates of the rear axle midpoint and $\theta = \theta(t)$ the orientation of the vehicle, *i.e.* the angle between the $x$ axis and the main axis of the vehicle. The motion of the vehicle is described by the following equations:

$$\begin{cases} \dot{x} = v \cos\phi \cos\theta \\ \dot{y} = v \cos\phi \sin\theta \\ \dot{\theta} = \frac{v}{L} \sin\phi \end{cases} \tag{1}$$

where $\phi = \phi(t)$ is the steering angle, *i.e.* the average orientation of the two front wheels of the vehicle. $v = v(t)$ is the locomotion velocity of the front axle midpoint and $L$ is the wheelbase. $(\phi, v)$, the steering angle and locomotion velocity, are the two control commands of the vehicle. Since the steering angle of a car is mechanically limited, the following constraint also holds (*maximum curvature constraint*):

$$|\phi| \leq \phi_{max} \tag{2}$$

Eqs. (1) correspond to a system with non-holonomic kinematic constraints because they involve the derivatives of the coordinates of the vehicle and are non-integrable (Latombe, 1991). They are valid for a vehicle moving on flat ground with perfect rolling assumption (no slippage between the wheels and the ground) at relatively low speed.

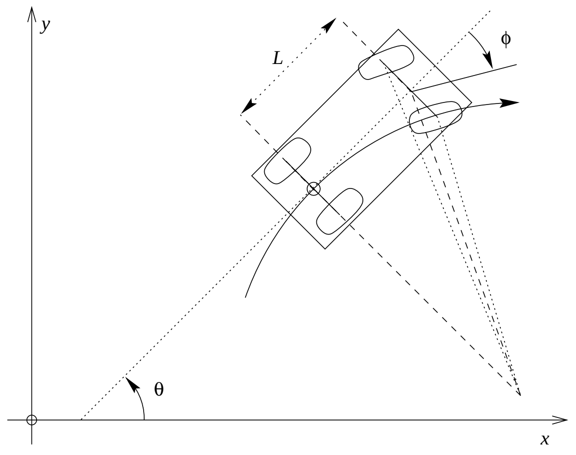

*Fig. 2.* Model of a car-like vehicle.

For high-speed motions, the dynamics of the vehicle must also be considered. In the current implementation of the architecture, only velocity and acceleration bounds are taken into account.

## 4. Trajectory Planning

As mentioned earlier, trajectory planning is an important function in the control architecture proposed. Its purpose is to compute a nominal trajectory leading the vehicle to its goal. A trajectory is a continuous time-ordered sequence of states, *i.e.* (configuration, velocity) pairs, between the current state of the vehicle and its goal. A trajectory must be collision-free and satisfy the kinematic and dynamic constraints of the vehicle.

In order to plan a trajectory that avoids the moving obstacles of the environment, the knowledge of their future behaviour is required. In most cases, this information is not *a priori* known. An estimation of the most likely behaviour of the moving obstacles is provided by a prediction function. The prediction function can be very simple (assuming that the moving obstacles keep a constant velocity) or more sophisticated (using models of human driver behaviour for instance). The quality of the prediction determines the quality of the nominal trajectory. However keep in mind that the trajectory planned is nominal: if the world does not 'behave' according to the prediction, the motion controller will deal with the prediction error and react accordingly. On the other hand, if the prediction is correct then the vehicle will follow a trajectory that has been planned so as to be optimal in time.

Trajectory planning for car-like vehicles in dynamic environments remains an open problem and a practical solution to this intricate problem is presented in this section.

## *4.1. Outline of the Approach*

The motion of a vehicle is subject to several types of constraints and the nominal trajectory has to respect them. These constraints are:

- *Kinematic constraints:* a wheeled car-like vehicle is subject to kinematic constraints, called *non-holonomic*, that restricts the geometric shape of its motion. Such a vehicle can move only in a direction which is perpendicular to its rear wheel axle (non-steering wheels) and its turning radius is lower-bounded.
- *Dynamic constraints:* these constraints arise because of the dynamics of the vehicle and the capabilities of its actuators (engine power, braking force, ground-wheel interaction, etc.). They restrict the accelerations and velocities of the vehicle.
- *No collision constraints:* collision with stationary and moving obstacles of the environment are forbidden.

A trajectory is a time-ordered sequence of states $(q, \dot{q})$. It can be represented also by a geometric path and a velocity profile along this path. Because of the intrinsic complexity of trajectory planning (*cf.* (Latombe, 1991) for complexity issues), the trajectory planner addresses the problem at hand in two complementary steps of lesser complexity:

1. *Path planning:* a geometric path leading the vehicle to its goal is computed. It is collision-free with the stationary obstacles of the environment and it respects the non-holonomic kinematic constraints of the vehicle.
2. *Velocity planning:* the velocity profile of the vehicle along its path is computed; this profile respects the dynamic constraints of the vehicle and yields no collisions between the vehicle and the moving obstacles of the environment.

Path planning is illustrated in the left-hand side of Fig. 3. It depicts an example path between two configurations. This collision-free path is a curve whose curvature is continuous and upper-bounded so as to respect the kinematic constraints of a car-like vehicle.

Velocity planning is illustrated in the right - hand side of Fig. 3. Recall that it requires the knowledge of the future behaviour of the moving obstacles (this information is provided by the prediction function). In the current implementation, a simple prediction function that assumes constant velocity for the moving obstacles is used. The right-hand side of Fig. 3 depicts a space-time diagram (the horizontal axis being the position along the path and the vertical one the time dimension). The curve represents the motion of the vehicle through time whereas the thick black lines are the traces left by moving obstacles when they cross the path of the vehicle.

The next two sections respectively present the path planning and the velocity planning steps.

## *4.2. Path Planning*

As mentioned earlier, a car-like vehicle is subject to non-holonomic kinematic constraints: it can move only along a direction perpendicular to its rear wheels axle (continuous tangent direction), and its turning radius is lower-bounded (maximum curvature). In the past ten years, numerous works, *e.g.* (Barraquand and Latombe, 1989; Laumond *et al.*, 1994; Švestka and Overmars, 1995), have tackled the problem of computing feasible paths for this type of vehicle. Almost all of them compute paths made up of circular arcs connected with tangential line segments. The key reason for that is that these paths are the shortest ones that respect the non-holonomic kinematic constraints of such a vehicle (Dubins, 1957; Reeds and Shepp, 1990). However their curvature profile is not continuous. Accordingly a vehicle following such a path has to stop at each curvature discontinuity, *i.e.* at each transition between a segment and an arc, in order to reorient its front wheels. This is hardly acceptable for a vehicle driving on the road. A solution to this problem is therefore to plan paths with a continuous curvature profile. In addition, a constraint on the curvature deriva-

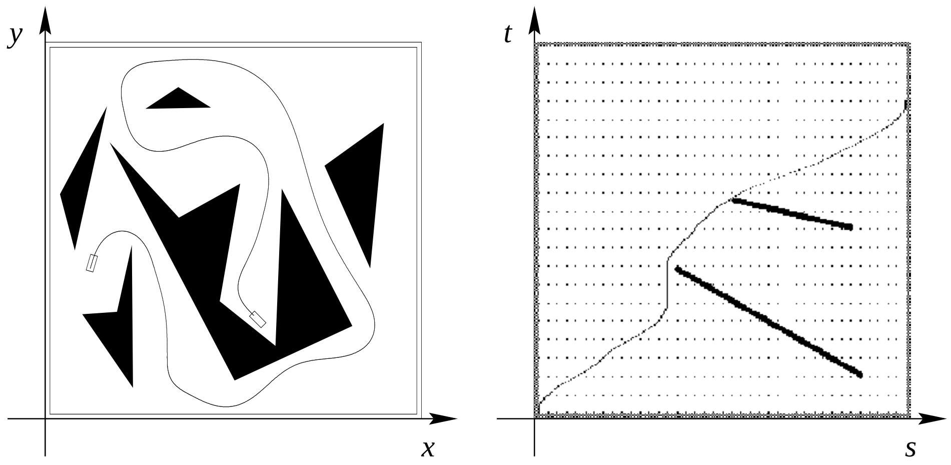

*Fig. 3.* (a) Path planning and (b) velocity planning.

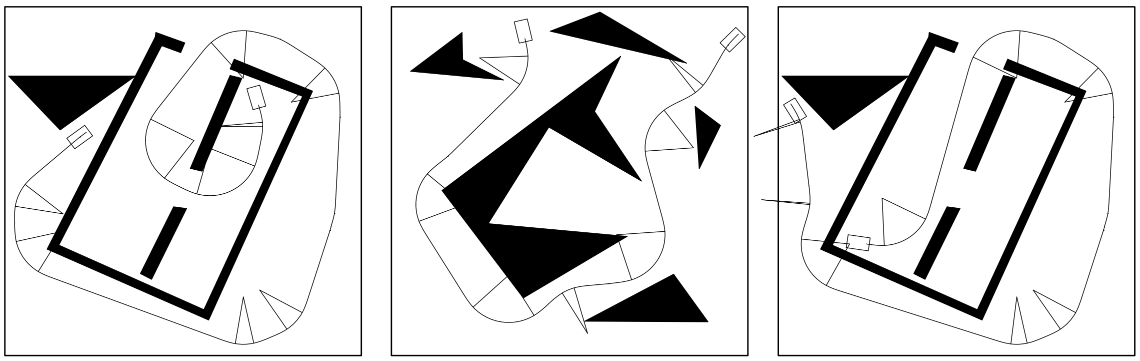

*Fig. 4.* Examples of continuous curvature paths.

tive is introduced; it is upper-bounded so as to reflect the fact that the vehicle can only reorient its front wheels with a finite velocity.

Addressing a similar problem (but without the maximum curvature constraint), (Boissonnat *et al.*, 1994) proves that the shortest path between two vehicle's configurations is made up of line segments and clothoids[1] of maximum curvature derivative. Unfortunately, (Kostov and Degtiariova-Kostova, 1995) later proves that these shortest paths are, in the general case, made up of an infinity of clothoids. These results also apply to the problem including the maximum curvature constraint. Therefore, in order to come up with a practical solution to the problem at hand, a set of paths that contain at most eight parts, each part being either a line segment, a circular arc, or a clothoid, has been defined. It is shown in (Scheuer and Laugier, 1998) that such paths are sub-optimal in length. They are used to design a *local path planner*, *i.e.* a non-complete collision-free path planner, which in turn is embedded in a global path planning scheme. The result is the first path planner for a car-like vehicle that generates collision-free paths with continuous curvature and upper-bounded curvature and curvature derivative. The reader is referred to (Scheuer and Fraichard, 1997) for a complete presentation of the continuous curvature path planner. Various experimental results are depicted in Fig. 4.

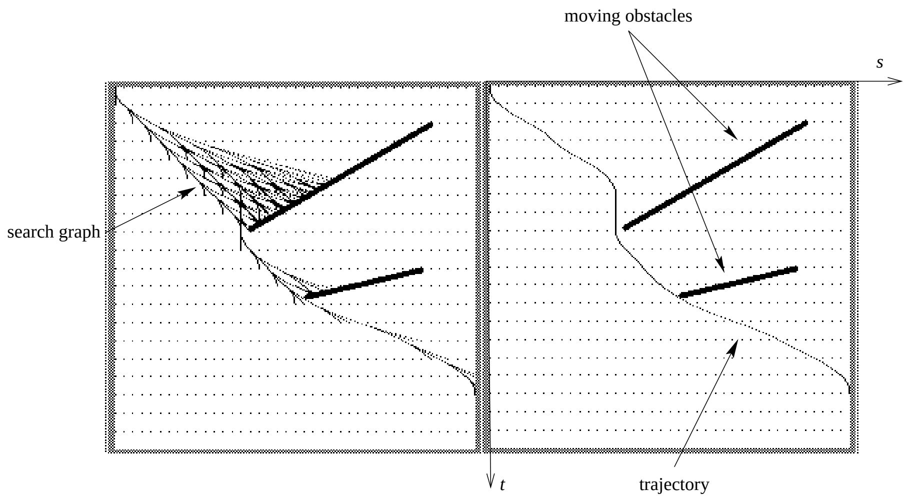

*Fig. 5.* An example of velocity planning.

## *4.3. Velocity Planning*

Given the nominal path generated by the path planner, the problem is to determine the trajectory of the vehicle along this path, *i.e.* its velocity profile; this profile must respect the dynamic constraints of the vehicle and yields no collision between the vehicle and the moving obstacles of the environment.

To address these two issues, *i.e.* moving obstacles and dynamic constraints, the concept of *state-time space*, has been introduced. It stems from two concepts that have been used before in order to deal respectively with moving obstacles and dynamic constraints, namely the concepts of *configuration-time space* (Erdmann and Lozano-Perez, 1987), and *state space*, *i.e.* the space of the configuration parameters and their derivatives. Merging these two concepts leads naturally to state-time space, *i.e.* the state space augmented of the time dimension (Fraichard, 1993). In this framework, the constraints imposed by both the moving obstacles and the dynamic constraints are represented by static forbidden regions of state-time space. Besides a trajectory maps to a curve in state-time space hence trajectory planning in dynamic workspaces simply consists in finding a curve in state-time space, *i.e.* a continuous sequence of state-times between the current state of the vehicle and a goal state. Such a curve must obviously respect additional constraints due to the fact that time is irreversible and that velocity and acceleration constraints translate to geometric constraints on the slope and the curvature along the time dimension. However it is possible to extend previous methods for path planning in configuration space in order to solve the problem at hand. In particular, a method derived from the one originally presented in (Canny *et al.*, 1988) has been designed to solve the problem at hand. It follows the paradigm of near-time-optimization: the search for the solution trajectory is performed over a restricted set of *canonical trajectories* hence the near-time-optimality of the solution. These canonical trajectories are defined as having piecewise constant acceleration that change its value at given times. Besides the acceleration is selected so as to be either minimum, null or maximum (bang controls). Under these assumptions, it is possible to transform the problem of finding the time-optimal canonical trajectory to finding the shortest path in a directed search graph embedded in the state-time space.

An example of velocity planning is depicted in Fig. 5. There are two windows: a trace window showing the part of the search graph which has been explored and a result window displaying the final trajectory. Any such window represents the $s{\times}t$ plane (the position axis is horizontal while the time axis is vertical; the frame origin is at the upper-left corner). The thick black segments represent the trails left by the moving obstacles and the little dots are nodes of the underlying state-time search graph. The obstacles are assumed to keep a constant velocity. The vehicle starts from position 0 (upper-left corner) with a null velocity, it is to reach position 1 (right border) with a null velocity. The reader is referred to (Fraichard, 1993) and (Fraichard and Scheuer, 1994) for more details about velocity planning.

## 5. Sensor-Based Manoeuvres

Recall that the control architecture proposed relies upon the concept of sensor-based manoeuvres (SBM). At a given time instant, the vehicle is carrying out a particular SBM that has been instantiated to fit the current execution context (see §2). SBMs are general templates encoding the knowledge of how a given motion task is to be performed. They combine real-time functions, control and sensing skills, that are either control programs or sensor data processing functions.

This section describes the two SBMs that have been developed and integrated in the control architecture proposed: trajectory following and parallel parking. These two manoeuvres have been implemented and successfully tested on a real automatic vehicle, the results of these experiments are presented in §6. The Orccad tool (Simon *et al.*, 1993) has been selected to implement both SBMs and skills. "Robot procedures" (in the Orccad formalism) are used to encode SBMs while "robot-tasks" encode skills. Robot procedures and robot tasks can both be represented as finite automata or transition diagrams. The "trajectory following" and "parallel parking" SBMs are depicted in Fig. 6 as transition diagrams. The control skills are represented by square boxes, *e.g.* "find parking place", whereas the sensing skills appear as predicates attached to the arcs of the diagram, *e.g.* "parking place detected", or conditional statements, *e.g.* "obstacle overtaken?". The next two sections describe how the two manoeuvres illustrated in Fig. 6 operates.

### *5.1. Trajectory Following*

The purpose of the trajectory following SBM is to allow the vehicle to follow a given nominal trajectory as closely as possible, while reacting appropriately to any unforeseen obstacle obstructing the way of the vehicle. Whenever such an obstacle is detected, the nominal trajectory is locally modified in real time, in order to avoid the collision. This local modification of the trajectory is done, in order to satisfy a set of different motion constraints: collision avoidance, time constraints, kinematic and dynamic constraints of the vehicle. In a previous approach, a fuzzy controller combining different basic behaviours (trajectory tracking, obstacle avoidance, etc.) was used to perform trajectory following (Garnier and Fraichard, 1996). However this approach proved unsatisfactory: it yields oscillating behaviours, and does not guarantee that all the aforementioned constraints are always satisfied.

The trajectory following SBM makes use of *local trajectories* to avoid the detected obstacles. These local trajectories allow the vehicle to move away from the obstructed nominal trajectory, and to catch up this nominal trajectory when the (stationary or moving) obstacle has been overtaken. All the local trajectories verify the motion constraints. This SBM relies upon two control skills, *trajectory tracking* and *lane changing* (*cf.* Fig. 6), that are detailed now.

*5.1.1. Trajectory Tracking* The purpose of this control skill is to issue the control commands that will allow the vehicle to track a given nominal trajectory. Several control methods for nonholonomic robots have been proposed in the literature. The method described in (Kanayama *et al.*, 1991) ensures stable tracking of a feasible trajectory by a car-like robot. It has been selected for its simplicity and efficiency. The vehicle's control commands are of the following form :

$$\dot{\theta} = \dot{\theta}_{ref} + v_{R,ref}(k_y y_e + k_\theta \sin\theta_e), \qquad (3)$$

$$v_R = v_{R,ref}\cos\theta_e + k_x x_e, \qquad (4)$$

where $q_e = (x_e, y_e, \theta_e)^T$ represents the error between the reference configuration $q_{ref}$ and the current configuration $q$ of the vehicle ($q_e = q_{ref} - q$), $\dot{\theta}_{ref}$ and $v_{R,ref}$ are the reference velocities, $v_R = v\cos\phi$ is the rear axle midpoint velocity, $k_x$,

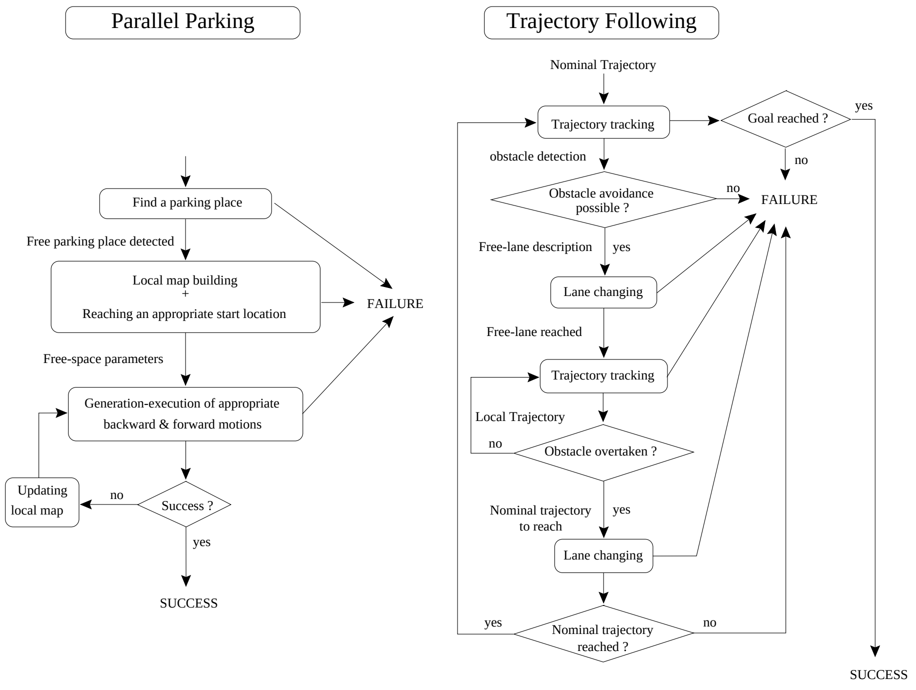

*Fig. 6.* The "parallel parking" and "trajectory following" SBMs.

$k_y$, $k_\theta$ are positive constants (the reader is referred to (Kanayama *et al.*, 1991) for full details about this control scheme).

*5.1.2. Lane Changing* This control skill is applied to execute a lane changing manoeuvre. The lane changing is carried out by generating and tracking an appropriate local trajectory. Let $\mathcal{T}$ be the nominal trajectory to track, $d_{\mathcal{T}}$ be the distance between $\mathcal{T}$ and the middle line of the free lane to reach, $s_{\mathcal{T}}$ be the curvilinear distance along $\mathcal{T}$ between the vehicle and the obstacle (or the selected end point for the lane change), and $s = s_t$ be the curvilinear abscissa along $\mathcal{T}$ since the starting point of the lane change (*cf.* Fig. 7).

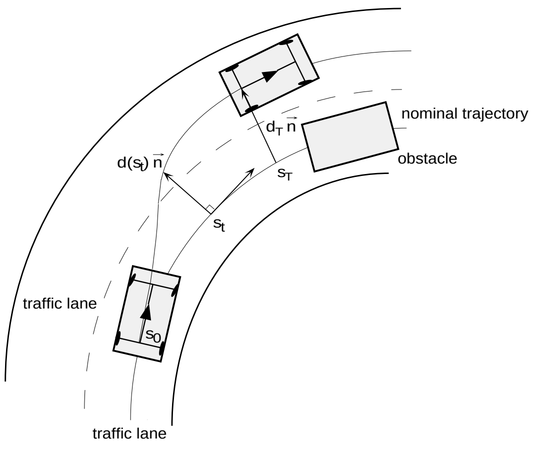

*Fig. 7.* Generation of smooth local trajectories for avoiding an obstacle.

A feasible smooth trajectory for executing a lane change can be obtained using the following quintic polynomial (*cf.* (Nelson, 1989)):

$$d(s) = d_{\mathcal{T}} \left( 10 \left( \frac{s}{s_{\mathcal{T}}} \right)^3 - 15 \left( \frac{s}{s_{\mathcal{T}}} \right)^4 + 6 \left( \frac{s}{s_{\mathcal{T}}} \right)^5 \right), \tag{5}$$

In this approach, the distance $d_{\mathcal{T}}$ is supposed to be known beforehand. Then the minimal value required for $s_{\mathcal{T}}$ can be estimated as follows:

$$s_{\mathcal{T},min} = \frac{\pi \sqrt{k\, d_{\mathcal{T}}}}{2\, \mathcal{C}_{max}}, \tag{6}$$

where $\mathcal{C}_{max}$ stands for the maximum allowed curvature:

$$\mathcal{C}_{max} = \mathbf{min}\left\{\frac{\tan(\phi_{max})}{L}, \frac{\gamma_{max}}{v^2_{R,ref}}\right\}, \qquad (7)$$

$\gamma_{max}$ is the maximum allowed lateral acceleration, and $k > 1$ is an empirical constant, *e.g.* $k = 1.17$ in our experiments.

At each time $t$ from the starting time $T_0$, the reference position $p_{ref}$ is translated along the vector $d(s_t).\vec{n}$, where $\vec{n}$ represents the unit normal vector to the nominal velocity vector along $\mathcal{T}$; the reference orientation $\theta_{ref}$ is converted into $\theta_{ref} + \arctan\left(\frac{\partial d}{\partial s}(s_t)\right)$, and the reference velocity $v_{R,ref}$ is obtained using the following equation:

$$v_{R,ref}(t) = \frac{dist(p_{ref}(t), p_{ref}(t + \Delta t))}{\Delta t}, \qquad (8)$$

where *dist* stands for the Euclidean distance. As shown in Fig. 6, this type of control skill can also be used to avoid a stationary obstacle, or to overtake another vehicle. As soon as the obstacle has been detected by the vehicle, a value $s_{T,min}$ is computed according to (6) and compared with the distance between the vehicle and the obstacle. The result of this computation is used to decide which behaviour to apply: avoid the obstacle, slow down or stop. In this approach, an obstacle avoidance or overtaking manoeuvre consists of lane changing manoeuvre towards a collision-free "virtual" parallel trajectory(see Fig. 7). The lane changing skill operates the following way:

1. Generate a smooth local trajectory $\tau_1$ which connects $\mathcal{T}$ with a collision-free local trajectory $\tau_2$ "parallel" to $\mathcal{T}$ ($\tau_2$ is obtained by translating appropriately the involved piece of $\mathcal{T}$).
2. Track $\tau_1$ and $\tau_2$ until the obstacle has been overtaken.
3. Generate a smooth local trajectory $\tau_3$ which connects $\tau_2$ with $\mathcal{T}$, and track $\tau_3$.

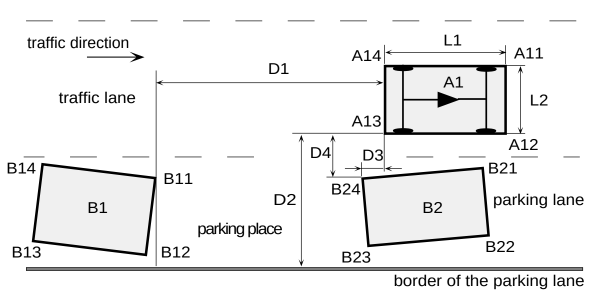

*Fig. 8.* Situation at the beginning of a parallel parking manoeuvre.

## *5.2. Parallel Parking*

Parallel parking comprises three main steps (*cf.* Fig. 6): localizing a free parking place, reaching an appropriate start location with respect to the parking place, and performing the parallel parking manoeuvre using iterative backward and forward motions until the vehicle is parked. During the first step, the vehicle moves slowly along the traffic lane and uses its range sensors to build a local map of the environment and detect obstacles. The local map is used to determine whether free parking space is available to park the vehicle.

A typical situation at the beginning of a parallel parking manoeuvre is depicted in Fig. 8. The autonomous vehicle $A1$ is in the traffic lane. The parking lane with parked vehicles $B1$, $B2$ and a parking place between them is on the right-hand side of $A1$. $L1$ and $L2$ are respectively the length and width of $A1$, and $D1$ and $D2$ are the distances available for longitudinal and lateral displacements of $A1$ within the place. $D3$ and $D4$ are the longitudinal and lateral displacements of the corner $A13$ of $A1$ relative to the corner $B24$ of $B2$.

Distances $D1$, $D2$, $D3$ and $D4$ are computed from data obtained by the sensor systems. The length ($D1 - D3$) and wide ($D2 - D4$) of the free parking place are compared with the length $L1$ and width $L2$ of $A1$ in order to determine whether the parking place is sufficiently large.

During parallel parking, iterative low-speed backward and forward motions with coordinated control of the steering angle and locomotion velocity are performed to produce a lateral displacement of the vehicle into the parking place. The number of such motions depends on the distances $D1$, $D2$, $D3$, $D4$ and the necessary parking depth (that depends on the width $L2$ of the vehicle $A1$). The start and end orientations of the vehicle are the same for each iterative motion.

For the $i$-th iterative motion (but omitting the index "$i$"), let the start coordinates of the vehicle be $x_0 = x(0)$, $y_0 = y(0)$, $\theta_0 = \theta(0)$ and the end coordinates be $x_T = x(T)$, $y_T = y(T)$, $\theta_T = \theta(T)$, where $T$ is duration of the motion. The "parallel parking" condition means that

$$\theta_0 - \delta_\theta \ < \ \theta_T \ < \ \theta_0 + \delta_\theta, \qquad (9)$$

where $\delta_\theta > 0$ is a small admissible error in orientation of the vehicle.

The following control commands of the steering angle $\phi$ and locomotion velocity $v$ provide the parallel parking manoeuvre (Paromtchik and Laugier, 1996*b*):

$$\phi(t) = \phi_{max}\, k_\phi\, A(t), \quad 0 \le t \le T, \qquad (10)$$

$$v(t) = v_{max}\, k_v\, B(t), \quad 0 \le t \le T, \qquad (11)$$

where $\phi_{max} > 0$ and $v_{max} > 0$ are the admissible magnitudes of the steering angle and locomotion velocity respectively, $k_\phi = \pm 1$ corresponds to a right side (+1) or left side (–1) parking place relative to the traffic lane, $k_v = \pm 1$ corresponds to forward (+1) or backward (–1) motion,

$$A(t) = \begin{cases} 1, & 0 \le t < t', \\ \cos \frac{\pi(t-t')}{T^*}, & t' \le t \le T - t', \\ -1, & T - t' < t \le T, \end{cases} \qquad (12)$$

$$B(t) = 0.5 \left(1 - \cos \frac{4\pi t}{T}\right), \quad 0 \le t \le T, \qquad (13)$$

where $t' = \frac{T-T^*}{2}$, $T^* < T$. The shape of the type of paths that corresponds to the controls (12) and (13) is shown in Fig. 9.

The commands (10) and (11) are open-loop in the $(x,\, y,\, \theta)$-coordinates. The steering wheel servo-system and locomotion servo-system must execute the commands (10) and (11), in order to provide the desired $(x,\, y)$-path and orientation $\theta$ of the vehicle. The resulting accuracy of the motion in the $(x,\, y,\, \theta)$-coordinates depends on the accuracy of these servo-systems. Possible errors are compensated by subsequent iterative motions.

For each pair of successive motions $(i,\, i+1)$, the coefficient $k_v$ in (11) has to satisfy the equation $k_{v,i+1} = -k_{v,i}$ that alternates between forward and backward directions. Between successive motions, when the velocity is null, the steering wheels turn to the opposite side in order to obtain a suitable steering angle $\phi_{max}$ or $-\phi_{max}$ to start the next iterative motion.

In this way, the form of the commands (10) and (11) is defined by (12) and (13) respectively. In order to evaluate (10)-(13) for the parallel parking manoeuvre, the durations $T^*$ and $T$, the magnitudes $\phi_{max}$ and $v_{max}$ must be known.

The value of $T^*$ is lower-bounded by the kinematic and dynamic constraints of the steering wheel servo-system. When the control command (10) is applied, the lower bound of $T^*$ is

$$T^*_{min} = \pi\, \mathbf{max} \left\{ \frac{\phi_{max}}{\dot{\phi}_{max}},\ \sqrt{\frac{\phi_{max}}{\ddot{\phi}_{max}}} \right\}, \qquad (14)$$

where $\dot{\phi}_{max}$ and $\ddot{\phi}_{max}$ are the maximal admissible steering rate and acceleration respectively for the steering wheel servo-system. The value of $T^*_{min}$ gives duration of the full turn of the steering wheels from $-\phi_{max}$ to $\phi_{max}$ or vice versa, *i.e.* one can choose $T^* = T^*_{min}$.

The value of $T$ is lower-bounded by the constraints on the velocity $v_{max}$ and acceleration $\dot{v}_{max}$ and by the condition $T^* < T$. When the control command (11) is applied, the lower bound of $T$ is

$$T_{min} = \mathbf{max} \left\{ \frac{2\pi\, v'(D1)}{\dot{v}_{max}},\ T^* \right\}, \qquad (15)$$

where $v'(\mathrm{D1}) \le v_{max}$, empirically-obtained function, serves to provide a smooth motion of the vehicle when the available distance $D1$ is small.

The computation of $T$ and $\phi_{max}$ aims to obtain the maximal values such that the following "longitudinal" and "lateral" conditions are still satisfied:

$$|\,(x_T - x_0)\, \cos\theta_0 + (y_T - y_0)\, \sin\theta_0\,| \ < \ D1, \quad (16)$$

$$|\,(x_0 - x_T)\, \sin\theta_0 + (y_T - y_0)\, \cos\theta_0\,| \ < \ D2. \quad (17)$$

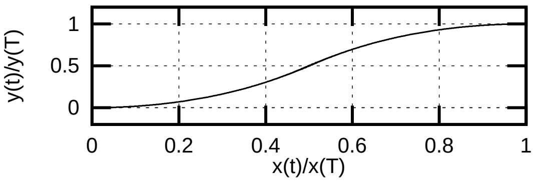

*Fig. 9.* Shape of a parallel forward/backward motion.

Using the maximal values of $T$ and $\phi_{max}$ assures that the longitudinal and, especially, lateral displacement of the vehicle is maximal within the available free parking space. The computation is carried out on the basis of the model (1) when the commands (10) and (11) are applied. In this computation, the value of $v_{max}$ must correspond to a safety requirement for parking manoeuvres, *e.g.* $v_{max} = 0.75\ m/s$ was found empirically.

At each iteration $i$ the parallel parking algorithm is summarized as follows:

1. Obtain available longitudinal and lateral displacements $D1$ and $D2$ respectively by processing the sensor data.
2. Search for maximal values $T$ and $\phi_{max}$ by evaluating the model (1) with controls (10), (11) so that conditions (16), (17) are still satisfied.
3. Steer the vehicle by controls (10), (11) while processing the range data for collision avoidance.
4. Obtain the vehicle's location relative to environmental objects at the parking place. If the "parked" location is reached, stop; else, go to step 1.

When the vehicle $A1$ moves backwards into the parking place from the start location shown in Fig. 8, the corner $A12$ (front right corner of the vehicle) must not collide with the corner $B24$ (front left corner of the place). The start location must ensure that the subsequent motions will be collision-free with objects limiting the parking place. To obtain a convenient start location, the vehicle has to stop at a distance $D3$ that will ensure a desired minimal safety distance $D5$ between the vehicle and the nearest corner of the parking place during the subsequent backward motion. The relation between the distances $D1$, $D2$, $D3$, $D4$ and $D5$ is described by a function $\mathcal{F}(D1, D2, D3, D4, D5) = 0$. This function can not be expressed in closed form, but it can be estimated for a given type of vehicle by using the model (1) when the commands (10) and (11) are applied. The computations are carried out off-line and the results are stored in a look-up table which is used on-line, to obtain an estimate of $D3$ corresponding to a desired minimal safety distance $D5$ for given $D1$, $D2$ and $D4$ (Paromtchik and Laugier, 1996*a*). When the necessary parking "depth" has been reached, clearance between the vehicle and the parked ones is provided, *i.e.* the vehicle moves forwards or backwards so as to be in the middle of the parking place between the two parked vehicles.

## 6. Experimental Results

The approach described in the paper has been implemented and tested on our experimental automatic vehicle (a modified Ligier electric car). This vehicle is equipped with the following capabilities:

1. a *sensor unit* to measure relative distances between the vehicle and environmental objects,
2. a *servo unit* to control the steering angle and the locomotion velocity and
3. a *control unit* that processes data from the sensor and servo units in order to "drive" the vehicle by issuing appropriate servo commands.

This vehicle can either be manually driven, or it can move autonomously using the *control unit* based on a Motorola VME162-CPU board and a transputer net. A VxWorks real-time operating system is used. The *sensor unit* of the vehicle makes use of a belt of ultrasonic range sensors (Polaroid 9000) and of a linear CCD-camera. The servo unit consists of a steering wheel servo-system, a locomotion servo-system for forward and backward motions, and a braking servo-system to slow down and stop the vehicle. The steering wheel servo-system is equipped with a direct current motor and an optical encoder to measure the steering angle. The locomotion servo-system of the vehicle is equipped with a 12 $kW$ asynchronous motor and two optical encoders located onto the rear wheels (for odometry data). The vehicle has an hydraulic braking servo-system. The Motion Controller monitors the current steering angle, locomotion velocity, travelled distance, coordinates of the vehicle and range data from the environment, calculates an appropriate local trajectory and issues the required servo commands. The Motion Controller has been implemented using the Orccad software tools (Simon *et al.*, 1993) running on a Sun workstation. The compiled code is transmitted via Ethernet to the VME162-CPU board.

The experimental car is equipped with 14 ultrasonic range sensors (Polaroid 9000), eight of them (a minimal configuration) are used for the current version of the automatic parking system: three ultrasonic sensors are at the front of the car (looking in the forward direction), two sensors are situated on each side of the car and one ultrasonic sensor is at the rear of the car (looking in the backward direction). The measurement range is $0.5 - 10.0m$, the sampling rate is $60ms$. The sensors are activated sequentially (four sensors are emitting/receiving signals at each instant (one for each side of the car). This sensor system is intended to test the control algorithms only and for low-speed motion only. Certainly, a more complex sensor system, *e.g.* a combination of vision and ultrasonic sensors, must be use to ensure reliable operation in a dynamic environment.

An experimental run of the "follow trajectory" SBM with obstacle avoidance on circular road (roundabout) is shown in Fig. 10. In this experiment, the Ligier vehicle follows a nominal trajectory along the curved traffic lane, and it finds on its way another vehicle moving at a lower velocity (see Fig. 10a). When the moving obstacle is detected, a local trajectory for a right lane change is generated by the system, and the Ligier performs the lane changing manoeuvre, as illustrated in Fig.10b. Afterwards, the Ligier moves along a trajectory parallel to its nominal trajectory, and a left lane change is performed as soon as the obstacle has been overtaken (Fig. 10c). Finally the Ligier catches up its nominal trajectory, as illustrated in Fig. 10d.

The corresponding motion of the vehicle is depicted in Fig. 11a. The steering and velocity controls applied during this manoeuvre are shown in Fig. 11b and Fig. 11c. It can be noticed in this example that the velocity of the vehicle has increased when moving along the local "parallel" trajectory (Fig. 11c); this is due to the fact that the vehicle has to satisfy the time constraints associated to its nominal trajectory.

An experimental run of the parallel parking SBM in a street is shown in Fig. 12. This manoeuvre can be carried out in environments including moving obstacles, *e.g.* pedestrians or some other vehicles (*cf.* the video (Paromtchik and Laugier, 1997)). In this experiment, the Ligier was manually driven to a position near the parking place, the driver started the autonomous parking mode and left the vehicle. Then, the Ligier moved forward autonomously in order to localize the parking place, obtained a convenient start location, and performed a parallel parking manoeuvre. When, during this motion a pedestrian crosses the street in a dangerous proximity to the vehicle, as shown in Fig. 12a, this moving obstacle is detected, the Ligier slows down and stops to avoid the collision. When the way is free, the Ligier continues its forward motion. Range data is used to detect the parking bay. A decision to carry out the parking maneuver is made and a convenient start position for the initial backward movement is obtained, as shown in Fig. 12b. Then, the Ligier moves backwards into the bay, as shown in Fig. 12c. During this backward motion, the front human-driven vehicle starts to move backwards, reducing the length of the bay. The change in the environment is detected and taken into account. The range data shows that the necessary "depth" in the bay has not been reached, so further iterative motions are carried out until it has been reached. Then, the Ligier moves to the middle between the rear and front vehicles, as shown in Fig. 12d. The parallel parking maneuver is completed.

The corresponding motion of the vehicle is depicted in Fig. 13a where the motion of the corners of the vehicle and the midpoint of the rear wheel axle are plotted. The control commands (10) and (11) for parallel parking into a parking place situated at the right side of the vehicle are shown in Fig. 13b and Fig. 13c respectively. The length of the vehicle is $L1 = 2.5m$, the width is $L2 = 1.4m$, and the wheelbase is $L = 1.785m$. The available distances are $D1 = 4.9m$, $D2 = 2.7m$ relative to the start location of the vehicle. The lateral distance $D4 = 0.6m$ was measured by the sensor unit. The longitudinal distance $D3 = 0.8m$ was estimated so as to ensure the minimal safety distance $D5 = 0.2m$. In this case, five iterative motions are performed to park the vehicle. As seen in Fig. 13, the durations $T$ of the iterative motions, magnitudes of the steering angle $\phi_{max}$ and locomotion velocity $v_{max}$ correspond to the available displacements $D1$ and $D2$ within the

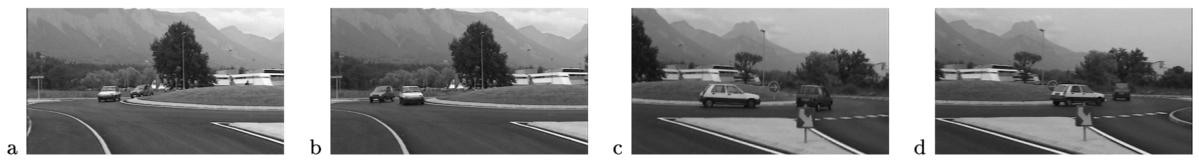

*Fig. 10.* Snapshots of trajectory following with obstacle avoidance in a roundabout: (a) following the nominal trajectory, (b) lane changing to the right and overtaking, (c) lane changing to the left, (d) catching up with the nominal trajectory.

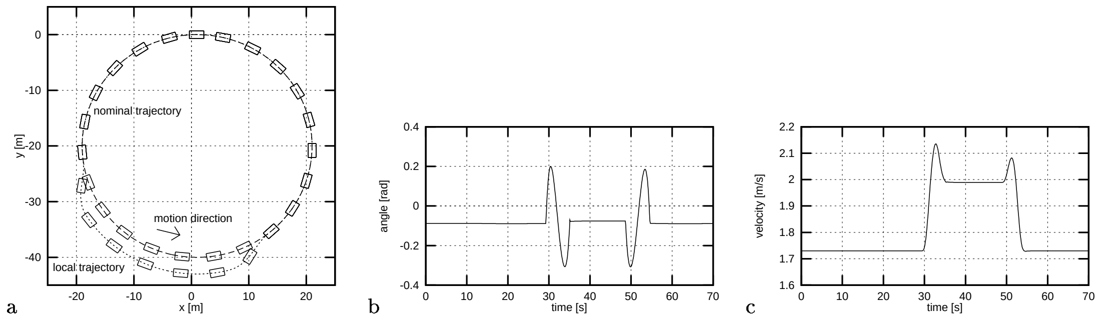

*Fig. 11.* Motion and control commands in the "roundabout" scenario: (a) motion, (b) steering angle and (c) velocity controls applied.

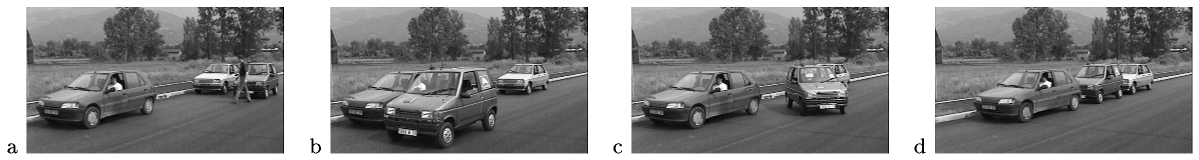

*Fig. 12.* Snapshots of a parallel parking: (a) localizing a free parking place, (b) selecting an appropriate start location, (c) performing a backward parking motion; (d) completing the parallel parking.

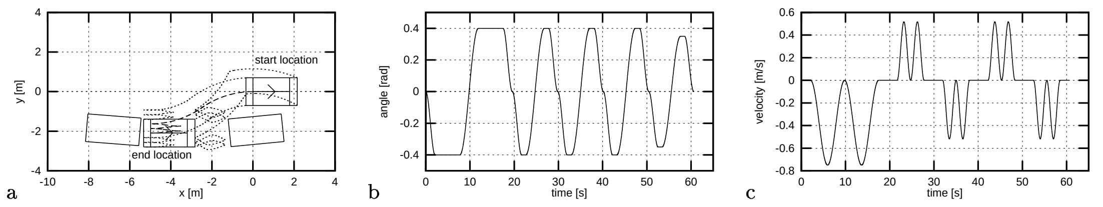

*Fig. 13.* Motion and control commands in the parallel parking scenario: (a) motion, (b) steering angle and (c) velocity controls applied.

parking place (*e.g.* the values of $T$, $\phi_{max}$ and $v_{max}$ differ for the first and last iterative motion).

## 7. Related Works

As mentioned in §1, motion autonomy has been a long standing issue in Robotics hence the important number of works presenting control architectures for robot systems. All these architectures are not reviewed here, the main trends are indicated instead.

Three main functions are to be found in any control architecture: perception, decision and action (hence the 'perception-decision-action' paradigm). After a careful examination of the existing control architectures, it appears that, to some extent, the difference between them lies in the decision function. Two types of approaches of completely opposite philosophy have appeared:

- *deliberative approaches:* in this type of approach, complex models of the environment of the robot are built from sensory data or *a priori* knowledge. These models are then used to perform high-level reasoning, *i.e.* planning, in order to determine which action to undertake. Maintaining these models and reasoning about them is, in most cases, a time-consuming process that makes these methods unable to deal with dynamic and uncertain environments. (Moravec, 1983; Nilsson, 1984) and (Waxman *et al.*, 1985) are good examples of this type of control architectures.
- *reactive approaches:* the philosophy of this type of approach is just the opposite: they favor reactivity. The decision function is reduced to a minimum. Action follows perception closely, almost like a reflex. This type of approach is most appropriate to dynamic and uncertain environments since unexpected events can be dealt with as soon as they are detected by the sensors of the robot. One drawback however, high-level reasoning is very difficult to achieve (if not impossible). (Brooks, 1990) is the canonical sensor-based control architecture; other examples are given in (Khatib and Chatila, 1995) or (Zapata *et al.*, 1990).

In an attempt to combine the advantages of both deliberative and reactive approaches, several authors have tried to combine high and low-level reasoning functions within a single control architecture. This idea permits to obtain *hybrid control architectures* with both high-level reasoning capabilities and reactivity.

The first hybrid architectures were obtained by simply putting together a deliberative and a reactive component. For instance, (Arkin, 1987) integrates a simple motion planner to a reactive architecture whereas (Gat *et al.*, 1990) sends the output of a task planner to a simple reactive execution controller: when a problem is detected at execution time, a reflex action is performed and the task planner is reinvoked. The performance of these approaches in terms of robustness, flexibility and reactivity are far from satisfactory. Better architectures have been proposed since, *e.g.* (Alami *et al.*, 1998; Gat, 1997) or (Simmons, 1994), they all combine three functional components:

- A set of elementary real-time functions (control loops, sensor data processing functions, etc.). A task is performed through the activation of such functions.
- A reactive execution mechanism that control and coordinates the execution of the real-time functions.
- A decision module that produces the task plan and supervises its execution. It may react to events from the execution function.

The control architecture presented in this paper clearly falls into this class of hybrid architectures. Skills are the real-time functions, the motion controller is the execution mechanism while the mission monitor is the decision module. With regard to these architectures, the main novelty of the approach proposed lies in the introduction of a meta-level of real-time functions, the sensor-based manoeuvres, that encapsulate high-level expert human knowledge and heuristics about the motion tasks to be performed, that permit to reduce the planning effort required to address a given motion task and thus to improve the overall response-time of the system.

## 8. Conclusion

This paper has presented an integrated control architecture endowing a car-like vehicle moving in a dynamic and partially known environment (the road network) with autonomous motion capabilities. Like most recent control architectures for autonomous robot systems, it combines three functional components: a set of basic real-time skills, a reactive execution mechanism and a decision module. The main novelty of the architecture proposed lies in the introduction of a fourth component akin to a meta-level of skills: the sensor-based manoeuvres, *i.e.* general templates that encode high-level expert human knowledge and heuristics about how a specific motion task is to be performed. The concept of sensor-based manoeuvres permit to reduce the planning effort required to address a given motion task, thus improving the overall response-time of the system, while retaining the good properties of a skill-based architecture, *i.e.* robustness, flexibility and reactivity.

After a general overview of the architecture proposed, the paper has covered in more details the trajectory planning function (which is an important part of the decision module) and two types of sensor-based manoeuvres: trajectory following and parallel parking. Experimental results with a real automatic car-like vehicle in different situations have been reported to demonstrate the efficiency of the approach. Future works will include the development and testing of other types of sensor-based manoeuvres.

## Acknowledgements

This work was partially supported by the Inria-Inrets[2] Praxitèle programme on urban public transport *[1994-1997]*, and the Inco-Copernicus ERB-IC15-CT96-0702 project "Multi-agent robot systems for industrial applications in the transport domain" *[1997-1999]*. The authors would like to thank E. Gauthier for his valuable contribution to the final version of the paper.

## Notes

1. A clothoid is a curve whose curvature is a linear function of its arc length.
2. Institut National de Recherche sur les Transports et leur Sécurité.

## References

Alami, R., R. Chatila, S. Fleury, M. Ghallab and F. Ingrand (1998). An architecture for autonomy. *Int. Journal of Robotics Research* **17**(4), 315–337. Special issue on integrated architectures for robot control and programming.

Arkin, R. C. (1987). Motor schema based navigation for a mobile robot. In: *Proc. of the IEEE Int. Conf. on Robotics and Automation*. Vol. 1. San Franciso, CA (US). pp. 264–271.

Barraquand, J. and J.-C. Latombe (1989). On nonholonomic mobile robots and optimal maneuvering. *Revue d'Intelligence Artificielle* **3**(2), 77–103.

Boissonnat, J.-D., A. Cérézo and J. Leblond (1994). A note on shortest paths in the plane subject to a constraint on the derivative of the curvature. Research Report 2160. Inst. Nat. de Recherche en Informatique et en Automatique. Rocquencourt (FR).

Brooks, R. A. (1990). A robust layered control system for a mobile robot. In: *Readings in Uncertain Reasoning* (G. Shafer and J. Perl, Eds.). pp. 204–213. Morgan Kaufmann.

Canny, J., B. Donald, J. Reif and P. Xavier (1988). On the complexity of kynodynamic planning. In: *Proc. of the IEEE Symp. on the Foundations of Computer Science*. White Plains, NY (USA). pp. 306–316.

Dubins, L. E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics* **79**, 497–517.

Erdmann, M. and T. Lozano-Perez (1987). On multiple moving objects. *Algorithmica* **2**, 477–521.

Fraichard, Th. (1993). Dynamic trajectory planning with dynamic constraints: a 'state-time space' approach. In: *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*. Vol. 2. Yokohama (JP). pp. 1394–1400.

Fraichard, Th. and A. Scheuer (1994). Car-like robots and moving obstacles. In: *Proc. of the IEEE Int. Conf. on Robotics and Automation*. Vol. 1. San Diego, CA (US). pp. 64–69.

Garnier, Ph. and Th. Fraichard (1996). A fuzzy motion controller for a car-like vehicle. In: *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*. Vol. 3. Osaka (JP). pp. 1171–1178.

Gat, E. (1997). On three-layer architectures. In: *Artificial Intelligence and Mobile Robots* (D. Kortenkamp, R. P. Bonnasso and R. Murphy, Eds.). MIT/AAAI Press.

Gat, E., M. G. Slack, D. P. Miller and R. J. Firby (1990). Path planning and execution monitoring for a planetary rover. In: *Proc. of the IEEE Int. Conf. on Robotics and Automation*. Vol. 1. Cincinatti, OH (US). pp. 20–25.

Kanayama, Y., Y. Kimura, F. Miyazaki and T. Noguchi (1991). A Stable Tracking Control Method for a Non-Holonomic Mobile Robot. In: *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*. Osaka (JP).

Khatib, M. and R. Chatila (1995). An Extended Potential Field Approach For Mobile Robot Sensor-Based Motions. In: *Proc. of Intelligent Autonomous Systems*. pp. 490–496.

Kostov, V. and E. Degtiariova-Kostova (1995). Some properties of clothoids. Research Report 2752. Inst. Nat. de Recherche en Informatique et en Automatique.

Latombe, J. C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers.

Laumond, J.-P., P. E. Jacobs, M. Taïx and R. M. Murray (1994). A motion planner for non-holonomic mobile robots. *IEEE Trans. Robotics and Automation* **10**(5), 577–593.

Moravec, H. P. (1983). The stanford cart and the CMU rover. *Proceedings of the IEEE* **71**(7), 872–884.

Nelson, W. L. (1989). Continuous curvature paths for autonomous vehicles. In: *Proc. of the IEEE Int. Conf. on Robotics and Automation*. Vol. 3. Scottsdale, AZ (US). pp. 1260–1264.

Nilsson, N. J. (1984). Shakey the robot. Technical note 323. AI Center, SRI International. Menlo Park, CA (US).

Parent, M. and P. Daviet (1996). Automated urban vehicles: towards a dual mode PRT (Personal Rapid Transit). In: *Proc. of the IEEE Int. Conf. on Robotics and Automation*. Minneapolis, MN (US). pp. 3129–3134.

Paromtchik, I. E. and C. Laugier (1996*a*). Autonomous parallel parking of a nonholonomic vehicle. In: *Proc.*

*of the IEEE Int. Symp. on Intelligent Vehicles.* Tokyo (JP). pp. 13–18.

Paromtchik, I. E. and C. Laugier (1996*b*). Motion generation and control for parking an autonomous vehicle. In: *Proc. of the IEEE Int. Conf. on Robotics and Automation.* Minneapolis, MN (US). pp. 3117–3122.

Paromtchik, I. E. and Ch. Laugier (1997). Automatic parallel car parking. In: Video-Proceedings of the IEEE Int. Conf. on Robotics and Automation. Albuquerque, NM (US). Produced by Inst. Nat. de Recherche en Informatique et en Automatique-Unité de Communication et Information Scientifique (3 min.).

Reeds, J. A. and L. A. Shepp (1990). Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics* **145**(2), 367–393.

Rich, E. and K. Knight (1983). *Artificial Intelligence.* McGraw-Hill.

Scheuer, A. and Ch. Laugier (1998). Planning sub-optimal and continuous-curvature paths for car-like robots. In: *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems.* Vol. 1. Victoria, BC (CA). pp. 25–31.

Scheuer, A. and Th. Fraichard (1997). Continuous-curvature path planning for car-like vehicles. In: *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems.* Vol. 2. Grenoble (FR). pp. 997–1003.

Simmons, R. G. (1994). Structured control for autonomous robots. *IEEE Trans. Robotics and Automation* **10**(1), 34–43.

Simon, D., B. Espiau, E. Castillo and K. Kapellos (1993). Computer-Aided Design of a Generic Robot Controller Handling Reactivity and Real-Time Control Issues. In: *IEEE Transactions on Control Systems Technology.* pp. 213–229.

Švestka, P. and M. H. Overmars (1995). Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In: *Proc. of the IEEE Int. Conf. on Robotics and Automation.* Vol. 2. Nagoya (JP). pp. 1631–1636.

Waxman, A. M., J. Le Moigne and B. Srinivasan (1985). Visual Navigation of Roadways. In: *Proc. of the IEEE Int. Conf. on Robotics and Automation.* Saint Louis, MI (US). pp. 862–867.

Zapata, R., B. Jouvencel and P. Lepinay (1990). Sensor-based Motion Control for Fast Mobile Robots. In: *IEEE Int. Workshop on Intelligent Motion Control.* Istambul (TR).

**Christian Laugier** received the M.Sc and Ph.D degrees in Computer Science from the university of Grenoble, France, in 1973 and 1976 respectively. He also received the "Docteur d'Etat" degree in Computer Science from the Institut National Polytechnique de Grenoble in 1987. Christian Laugier is Research Director at Inria and Director of the Sharp project at Inria Rhône-Alpes and at the Imag-Gravir laboratory. From 1974 to 1978, he worked in the field of computer Graphics and Computer Aided Design. In 1979, he joined the Lifia Laboratory (Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle) in Grenoble, where he worked until 1995 in the areas of automatic Robot programming, autonomous mobile Robots, and motion planning. From 1987 to 1992, he was Associate Director of Lifia, and from 1984 to 1995 he was Director of the Robotics Group at Lifia. His current research interests are in the areas of motion planning, telerobotics, autonomous vehicles, and dynamic simulation. He has published over 140 technical papers in the areas of Computer Graphics and Robotics. Christian Laugier was General Chairman of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems 1997.

**Thierry Fraichard** assumes the position of Research Associate at Inria Rhône-Alpes as a member of the Sharp project and the Imag-Gravir laboratory since December 1994. He received his Ph.D in Computer Science from the Institut National Polytechnique de Grenoble in April 1992 for his dissertation on "Motion planning for a non-holonomic mobile in a dynamic workspace". He was a Postdoctoral Fellow in the Manipulation Laboratory of the Robotics Institute at Carnegie Mellon University from December 1993 to November 1994. In 1997, he served as Secretary for the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems 1997. Dr. Fraichard's research focuses on motion autonomy for car-like vehicles with a special emphasis on motion planning for non-holonomic systems, motion planning in dynamic workspaces, motion planning in the presence of uncertainty and the design of control architectures for autonomous vehicles.

**Philippe Garnier** received the B.Sc degree in Computer Science from the university of Grenoble, France, in 1990. He received the M.Sc and Ph.D degrees in Computer Science from the Institut National Polytechnique de Grenoble in 1991 and 1995 respectively. From 1996 to 1997, he was a Postdoctoral Research Fellow at Inria Rhône-Alpes in Grenoble, France. His research interests include motion control for autonomous car-like vehicles in dynamic and structured environments. Since January 1998, Philippe Garnier works on the design, implementation and validation of low and high-level controllers for the autonomous car-like vehicles of Inria Rhône-Alpes.

**Igor Paromtchik** received the M.Sc degree in Radiophysics and Ph.D degree in System Analysis and Automatic Control from the Belarusian State University in 1985 and 1990 respectively. Dr. Paromtchik held positions of Research Scientist and Assistant Professor at this university until 1992. During 1992-1994, he worked as a Researcher at the Institute for Real-Time

Computer Systems and Robotics of the university of Karlsruhe, Germany. Since 1995, Dr. Paromtchik has been working at Inria Rhône-Alpes in Grenoble, France. From 1997, jointly with his position of Expert Engineer at Inria, Dr. Paromtchik serves as a Visiting Researcher at the Institute of Physical and Chemical Research (Riken) in Japan. His main research interests are control systems for mobile robots, system analysis and conception, software engineering for real-time computer systems and intelligent transportation systems.

**Alexis Scheuer** entered the Ecole Normale Supérieure de Lyon, France in 1989. He passed in June 1994 the Agrégation (second highest teaching examination in France) in Mathematics, and completed in January 1998 a Ph.D in Computer Science from the Institut National Polytechnique de Grenoble, France, about "Continuous-curvature path planning for non-holonomic mobile robot". After seven months as a Postdoctoral Research Fellow in the Autonomous Vehicle Laboratory of the Nanyang Technological University in Singapore, he is currently a Teaching Assistant at the university of Grenoble and a member of the Sharp project at Inria Rhône-Alpes and the Imag-Gravir laboratory. His research interests include motion planning, non-holonomic constraints, controllability and optimality.