# A Fuzzy Motion Controller for a Car-Like Vehicle

Philippe Garnier, Thierry Fraichard

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

# *A Fuzzy Motion Controller for a Car-Like Vehicle*

Philippe Garnier and Thierry Fraichard

**N° 3200**

June 25, 1997

THÈME 3


ISSN 0249-6399

# A Fuzzy Motion Controller for a Car-Like Vehicle

Philippe Garnier and Thierry Fraichard*

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet Sharp

Rapport de recherche n°3200 — June 25, 1997 — 32 pages

**Abstract:**

this paper describes an *'execution monitor'* (EM); it is the key component of a motion control architecture for a vehicle moving in a dynamic and partially known environment. EM endows the vehicle with the reactive capabilities required in an uncertain environment. Its purpose is to generate the commands for the servo-systems of the vehicle so as to follow a given nominal trajectory while reacting in real-time to unexpected events. EM is designed as a fuzzy controller, i.e. a control system based upon fuzzy logic, thus permitting approximate reasoning and a human-like description of the vehicle's reactive behaviour. The main components of EM are an inference engine and a set of 'linguistic rules'. The global behaviour of the vehicle results from the combination of several basic behaviours (trajectory following, obstacle avoidance, etc.), each of which is encoded by a specific set of rules. EM differs from classical fuzzy controllers in two novel ways: first, it introduces a new defuzzification technique, the *Barycentre of the Centres Of Area*, that permits to better take into account the influence of each and every rule. Second, weighing coefficients are attached to the rules thus permitting a fine tuning of the influence of each basic behaviour. Furthermore it is shown how supervised learning, i.e. learning through samples, can be used to automate the determination of these weights thus suppressing the ever delicate problem of finding such coefficients (the identification problem). EM has been implemented and tested on an electric car-like vehicle prototype. Promising results of preliminary experiments featuring trajectory following and unexpected obstacle avoidance are presented.

**Key-words:** mobile robot, control architecture, motion execution, fuzzy logic

*(Résumé : tsvp)*

* Email: Thierry.Fraichard@inria.fr

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN (France)
Téléphone : (33) 04 76 61 52 00 – Télécopie : (33) 04 76 61 52 52

# Un contrôleur flou pour un véhicule de type voiture

**Résumé :** ce papier décrit un *moniteur d'exécution* (ME); c'est l'élément clé d'une architecture destinée à contrôler les mouvements d'un véhicule évoluant dans un environnement dynamique et partiellement connu. ME dote le véhicule des capacités de réaction nécessaires dans un environnement incertain. Sa tâche est de déterminer les commandes à envoyer aux actionneurs du véhicule de façon à suivre une trajectoire nominale donnée tout en réagissant en temps réel aux événements imprévus. ME repose sur l'utilisation de la logique floue ce qui permet le raisonnement approximatif et la spécification en terme humain des capacités de réaction du véhicule. ME comporte essentiellement un moteur d'inférence flou et une base de 'règles linguistiques'. Le comportement global du véhicule résulte de la combinaison d'un ensemble de comportements de base (suivi de trajectoire, évitement d'obstacles, etc.), chaque comportement de base étant représenté par un ensemble particulier de règles. ME diffère des contrôleurs flous classiques de deux manières : tout d'abord, il introduit une nouvelle technique de defuzzification qui favorise une meilleure prise en compte de l'influence de chaque règle. Ensuite, des poids sont associés à chaque règle afin de permettre un meilleur ajustement de l'influence respective des différents comportement de base. Enfin, il est montré que l'apprentissage supervisé, i.e. par des exemples, peut être utilisé pour déterminer automatiquement ces poids, supprimant de ce fait le problème toujours délicat de l'identification de ce type de coefficients. ME a été implanté et testé sur un prototype de voiture électrique. Les résultats de premières expériences sur le suivi de trajectoire et l'évitement d'obstacles imprévus sont présentés.

**Mots-clé :** robot mobile, architecture de contrôle, exécution de mouvement, logique floue

# Contents

# List of Figures

# 1 Introduction

As noted by Latombe [15], motion planning, i.e. computing the sequence of configurations[1] allowing a robot to move from here to there, is a central problem in the development of autonomous robots. This is true indeed, but when the environment of the robot becomes complex, i.e. uncertain, partially known, with moving obstacles or other robots, etc., it takes much more than motion planning to achieve motion autonomy. In this case, the ability to detect unexpected events and react accordingly becomes essential.

Motion autonomy, for car-like vehicles in particular, is one of our main research goals. It has been pursued since 1986 within two different research programmes on road transport: Prometheus[2] and Praxitèle[3]. The road network is a perfect example of a complex environment; it features moving obstacles (other vehicles, pedestrians, etc.) that can move fast and whose future behaviour cannot be know a priori let alone predicted reliably. In this framework, we have developed a control architecture designed to plan and control the motion of a car-like vehicle moving in a dynamic and partially known environment such as the road network (see [7, 10, 11]). This control architecture is presented in §2; it relies upon two main complementary modules: a *trajectory planner* that computes a nominal trajectory between the current configuration of the vehicle and its goal, and an *execution monitor* whose purpose is to pilot the vehicle and endow it with the required reactive capabilities.

This paper focuses on the execution monitor. i.e. the module that actually pilots the vehicle. It has to fulfill the following functions:

- Generating the commands for the actuators of the vehicle so as to follow the nominal trajectory as closely as possible.
- Monitoring the execution of the trajectory.
- Reacting in real-time to unexpected events by locally adapting the trajectory actually followed by the vehicle.

Additionally the execution monitor may also have to reinvoke the trajectory planner when it becomes impossible to catch up with the nominal trajectory.

The execution monitor is designed as a fuzzy controller, i.e. a reactive system based upon fuzzy logic [16]. Its main components are an inference engine and a set of 'linguistic rules' that encode the reactive behaviour of the vehicle. These rules are of the type `if condition then action`. The condition and action parts of a rule include 'symbolic variables' such as `slow`, `fast`, etc., that are represented by fuzzy sets[4]. Linguistic rules permit to describe the

[1] The configuration of a robotic system is an independent set of parametres that uniquely specify the position and orientation of every part of the system.

[2] Eurêka EU-153 European research programme *[1986-1994]*. Prometheus stands for 'PROgraMme for a European Traffic with Highest Efficiency and Unprecedented Safety'.

[3] Inria-Inrets French research programme on Public Individual Transport *[1994-1997]*.

[4] In ordinary set theory, an element belongs to a given set or not. In fuzzy set theory, an element belongs to a set with a certain 'membership degree', i.e. a value in the range $[0, 1]$. The fuzzy set theory can be seen as a generalization of the ordinary set theory (where the membership degree belongs to $\{0, 1\}$).

behaviour of the vehicle the way a human expert would do it. Actually the global behaviour of the vehicle results from the combination of several basic behaviours (trajectory following, obstacle avoidance, etc.), each of which is encoded by a specific set of rules.

As we will see further down, the execution monitor differs from classical fuzzy controllers in two novel ways: the first one is a bit technical and regards the so-called defuzzification process. At this point, suffice it to say that, thanks to the novel defuzzification process we have introduced, the influence of each and every rule is better taken into consideration. The second difference lies in the fact that weighing coefficients are attached to the rules thus permitting a fine tuning of the influence of each basic behaviour. We borrowed this idea from [25] but took it a step further by using supervised learning [6] to automate the determination of these weights thus suppressing the ever delicate problem of finding such coefficients.

**Outline of the Paper.** To begin with §2 gives a short presentation of the overall control architecture of the vehicle. The trajectory planner is then sketched in §3. Afterwards the execution monitor is presented in detail in §4. Finally experimental results and future developments are presented in §5.

# 2 The Control Architecture

To begin with, this section briefly reviews the main types of control architectures for robots that can be found in the literature. Then it sketches the control architecture we have developed and within which is embedded the execution monitor.

## 2.1 Types of Control Architectures

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. Thus it is possible to differentiate between:

- *Model-based 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. 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. [18] and [28] are good examples of this type of control architectures.
- *Sensor-based approaches:* the philosophy of this type of approach is just the opposite: they favor reactivity. The decision function is reduced to a minimum. Thus 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) in these methods. [3] is the canonical sensor-based control architecture; other examples are given in [13] or [31].

- *Hybrid approaches:* in an attempt to combine the advantages of both model and sensor-based approaches, it has been suggested to put together high and low-level reasoning functions within a single control architecture (usually they run in parallel with different time frequencies). This idea permits to obtain control architectures with both high-level reasoning capabilities and reactivity. Refs. [1] and [24] are representative of this type of control architectures.

### 2.2 Our Control Architecture

Motion autonomy in a dynamic and partially known environment requires both high-level reasoning capabilities and reactivity (so as to be able to deal with unexpected events in due time). Accordingly we opted for a hybrid control architecture. It fits the 'perception-decision-action' paradigm and is depicted in Fig. 1. The focus in this section is on the decision part and the reader is referred to [10] for a complete presentation of the architecture.

The decision part comprises two main modules that are complementary: the *trajectory planner* (TP) and the *execution monitor* (EM).

The purpose of TP is to compute a nominal trajectory for the vehicle, i.e. a time-ordered sequence of (configuration, velocity) couples between the current configuration of the vehicle and its goal. The determination of this trajectory relies upon:

- A priori information on the environment of the vehicle (e.g. position of the stationary obstacles).
- Sensory data (e.g. position and velocity of the moving obstacles).
- Hypotheses about the future evolution of the workspace (e.g. prediction of the future behaviour of the moving obstacles).

Of course the predictions made may not be reliable, it is therefore necessary to give the vehicle the ability to deal with unpredicted events. This is the purpose of EM, it generates commands for the actuators of the vehicle so as to follow the nominal trajectory as closely as possible while reacting in real-time to unexpected events by locally adapting the trajectory actually followed by the vehicle. TP is sketched in §3 while EM is detailed in §4.

## 3 The Trajectory Planner (TP)

A detailed presentation of TP is beyond the scope of this paper; it is only sketched in this section. The reader is referred to [8] and [9] for more details. The purpose of TP is to compute a nominal trajectory between the current configuration of the vehicle and its goal. The nominal trajectory is a time-ordered sequence of states, i.e. (configuration, velocity) couples, it has to respect different types of constraints :

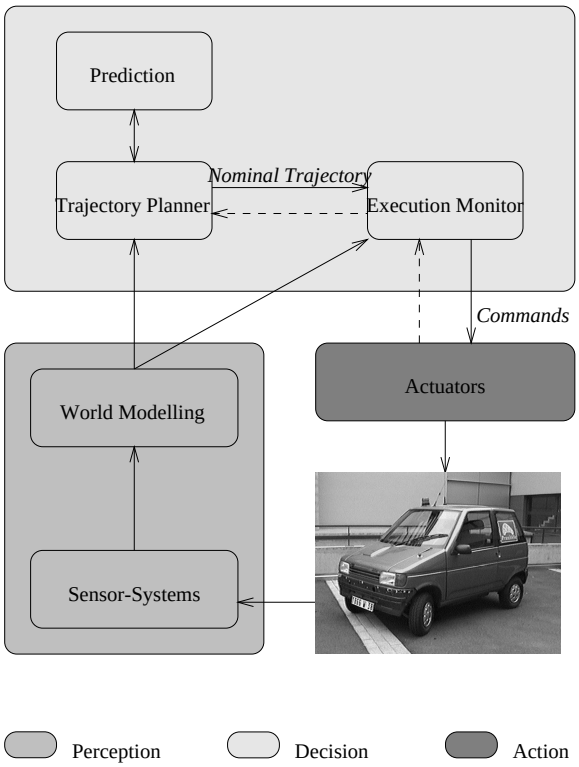


Figure 1: the control architecture of the vehicle.

- *Kinematic constraints:* a car-like vehicle has a limited steering range that restrict the geometric shape of its motion.
- *Dynamic constraints:* these constraints due to engine power, ground-wheel interaction, etc., restrict the accelerations and velocities that can be applied to the vehicle.
- *Collision avoidance constraints:* collision with the stationary and moving obstacles of the environment are forbidden.

Given that a trajectory can be represented also by a geometric path and a velocity profile along this path, TP addresses the problem at hand in two complementary steps:

1. *Path planning:* a geometric path leading the vehicle to its goal is computed. The left-hand side of Fig. 2 depicts an example of such a path: it is a continuous curve whose curvature is upper-bounded (so as to respect the kinematic constraints of a car-like vehicle). Besides it is collision-free with the stationary obstacles of the environment.

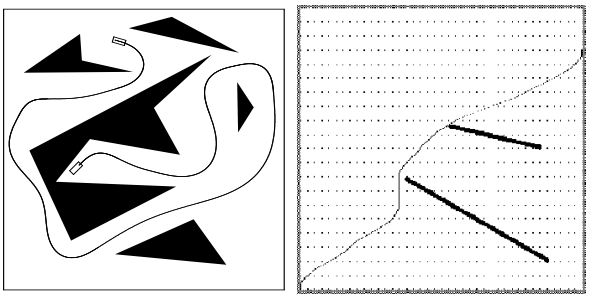

Figure 2: (a) path planning and (b) velocity planning.

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 collision between the vehicle and the moving obstacles of the environment. Velocity planning requires the knowledge of the future behaviour of the moving obstacles; this information is provided by a prediction module. The right-hand side of Fig. 2 illustrates the velocity planning: it 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.

# 4 The Execution Monitor (EM)

As mentioned earlier, the purpose of EM is to generate commands for the actuators of the vehicle so as to follow the nominal trajectory as closely as possible while reacting in real-time to unexpected events by locally adapting the trajectory actually followed by the vehicle. EM is designed as a fuzzy controller, i.e. a control system based upon fuzzy logic. Before detailing EM and its main features, let us briefly recall what a fuzzy controller is.

## 4.1 Fuzzy Controllers

Mamdani's pioneering research on fuzzy controllers [17] was motivated by Zadeh's work on the theory of fuzzy sets[3] [30]. Ever since, fuzzy controllers (FCs) have become increasingly popular (the reader is referred to [16] for a review on this topic). The general architecture of a FC is depicted in Fig. 3. It is a control system whose input is the output of the process to be controlled (sensory data, internal state). Its outputs are commands for the actuators of the process. A FC is made up of four components:

1. The *knowledge base:* it encodes the desired behaviour of the process, it is made up of:
   - Fuzzy sets associated with the input and output variables of the process.

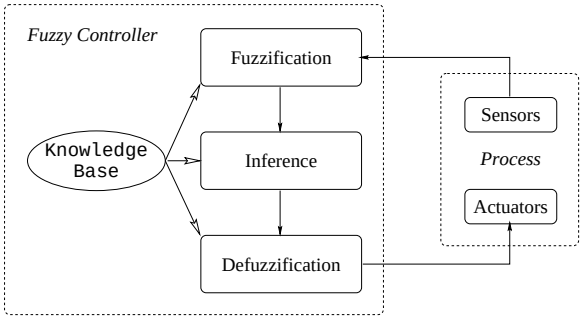


Figure 3: the architecture of a fuzzy controller.

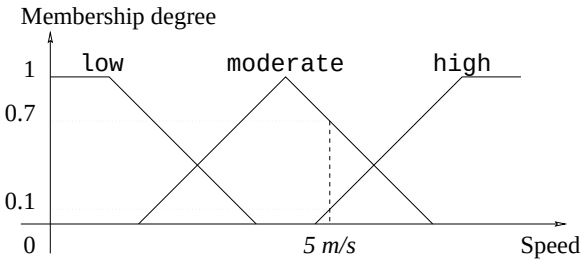


Figure 4: the three fuzzy sets, labeled `low`, `moderate` and `high`, that are associated with the speed variable.

- Linguistic rules, i.e. rules of the type: `if condition then action`.

2. The *fuzzification module:* it turns real input values into fuzzy values, i.e. a set of (fuzzy set label, membership degree) couples.
3. The *inference engine:* it is the kernel of the FC. By using the knowledge base and fuzzy logic, it determines the fuzzy commands to apply to the process.
4. The *defuzzification module:* it turns fuzzy commands into actual commands for the actuators of the process.

The interest of such an approach is twofold: 1) fuzzy sets and fuzzy inference permit approximate reasoning, and 2) a complete model of the controlled process is not necessary, a description of the way it works in human-like terms is used instead.

Let us briefly illustrate the way a classical FC works through the example of a car-like vehicle whose output variables are speed and distance, i.e. the distance to the car in front. and whose input variables are acceleration and braking.

Three fuzzy sets labeled `low`, `moderate` and `high` are associated with the speed variable, and similarly with the distance, acceleration and braking variables. Fig. 4 depicts the fuzzy

sets associated with the speed variable. Classically, they are represented by trapezoid regions in the speed vs. membership degree space.

Given the actual speed of the car, the *fuzzification* module yields a set of (fuzzy set label, membership degree) couples. For instance a $5m/s$ speed is `moderate` with a membership degree of 0.7 and `high` with a membership degree of 0.1, meaning that this speed is closer to `moderate` than `high` (Fig. 4).

In order to specify a simple behaviour for the car, namely that it should not drive too close to the car in front, the following linguistic rules could 'naturally' be written:

```
Rule 1: if   (distance is low) and
             (speed is moderate)
        then (braking is low)
Rule 2: if   (distance is low) and
             (speed is high)
        then (braking is high)
Rule 3: etc.
```

Now, the *fuzzy inference* works the following way: the actual input variables, i.e. speed and distance, are first fuzzified into a set of (fuzzy set label, membership degree) couples. Then the 'activation degree' of each rule is computed; it is a function of the membership degrees of the input variables present in the condition part of the rule; function whose definition depends on the operators used to combine the input variables, e.g. 'and' means taking the minimum of the membership degrees. The activation degree of a given rule is then used to compute the result of the fuzzy inference for the rule considered. This result is a new fuzzy set and the two main techniques used to obtain it are:

- Mamdani's inference (a.k.a. minimum inference [17]) in which the result set is the part of the fuzzy set associated with the output variable that lies beneath the activation degree (Fig. 5).
- Larsen's inference (a.k.a. product inference [14]) in which the result set is obtained by scaling the fuzzy set associated with the output variable with respect to the activation degree.

Finally, given all the fuzzy sets inferred for a given output variable, i.e. acceleration or braking, it is up to the *defuzzification* module to determine the actual command sent to the car. This particular point is detailed in §4.3.

## 4.2 Main Features of EM

EM is designed as a FC and, as such, it includes the four components described above. It applies Mamdani's inference. However it differs from classical FCs in two novel ways:

- To begin with, we introduced a new defuzzification method that permits to better take into consideration the influence of each and every linguistic rule of the knowledge base. This new method is called *barycentre of the centres of area* (see §4.3).

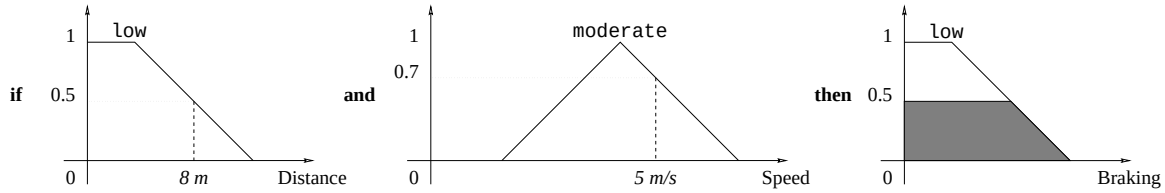


Figure 5: an example of fuzzy inference with rule 1: the activation degree of the rule is the minimum ('and' operator) of the membership degrees of the input variables of the left-hand side of the rule. Applying Mamdani's inference yields the part of the fuzzy set associated with the output variable that lies beneath the activation degree (grey area).

- Then, we attached *weighing coefficients* to the linguistic rules. These weights represent the importance of each rule with respect to the others and permit a finer tuning of EM (cf. §4.4). We borrowed this idea from [25] but took it a step further by using *supervised learning* [6] to automate the determination of these weights thus facilitating the design of EM. This point is detailed in §4.6.

Additionally, we opted for a modular organization of the base of linguistic rules. In other words, the global behaviour of the vehicle is obtained through the combination of several *basic behaviours*, each of which is encoded by a specific set of linguistic rules. This approach permits an incremental construction of the knowledge base and also to develop and test the basic behaviours separately. The different basic behaviours present in EM are described in §4.5.

## 4.3 A Novel Defuzzification Method

Recall that the purpose of defuzzification is to compute an actual value from the fuzzy sets inferred for a given command variable. Several defuzzification methods can be found in the literature (see [16]). The basic idea is to compute the union of these fuzzy sets, to pick up a point in the resulting fuzzy set and to take its abscissa as the actual command.

Existing defuzzification methods differ mostly in the way they pick up the point in the final fuzzy set. Two main approaches stand out however: the Mean Of Maxima (MOM) and the Centre Of Area (COA). MOM considers all the points whose membership degree is maximum and selects the mean of these points. COA picks up the center of area of the final fuzzy set. Another interesting defuzzification method is proposed in [29], the Centroïd of Largest Area (CLA). Instead of considering the union of the fuzzy sets for a given command variable, it takes the largest one and applies COA to this set.

MOM and COA consider the union of the fuzzy sets corresponding to a given command variable. By doing so, they merge the fuzzy sets with the same labels and thus ignore the fact that they may have resulted from the activation of several rules. CLA on the other hand discards some fuzzy sets. In all cases, this can adversely affect the command finally selected by giving an unjustified importance to a given rule.

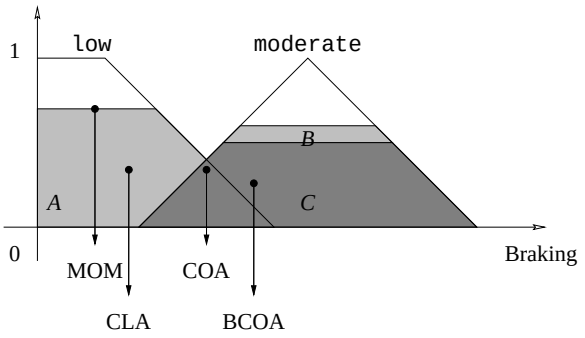


Figure 6: results of the different defuzzification methods.

This is illustrated in Fig. 6 that depicts an example where three different rules have been activated resulting in three different fuzzy sets (labeled $A$, $B$ and $C$) for the braking command variable. The key thing in this example is that $B$ and $C$ overlap. Two rules have voted for a moderate braking but, when applying MOM, COA or even CLA, this information is lost and an unjustified importance is given to a low braking. To remove this drawback, we do not use the union of the fuzzy sets, instead the *Barycentre of the Centres Of Area* (BCOA) of the different fuzzy sets is computed so as to better take into account the influence of each and every rule. In the example of Fig. 6, one would expect moderate braking to be the right decision since two rules out of three have voted for it; BCOA is the defuzzification method that gives the best result.

Formally, the BCOA defuzzification method is defined as follows: let $\{f_1, \ldots, f_k\}$ be the set of fuzzy sets inferred for a given command variable, the actual value delivered by BCOA is:

$$\frac{\sum_{i=1}^{k} area(f_i)\, coa(f_i)}{\sum_{i=1}^{k} area(f_i)} \tag{1}$$

where $area(f)$ denotes the area of the fuzzy set $f$, and $coa(f)$ its centre of area.

## 4.4 Weighing Coefficients

Following [25], we attached *weighing coefficients* to the linguistic rules. These weights represent the importance of each rule with respect to the others. Formally, they are taken

into account in the BCOA defuzzification method that now delivers the following value:

$$\frac{\sum_{i=1}^{k} area(f_i)coa(f_i)w_i}{\sum_{i=1}^{k} area(f_i)} \tag{2}$$

where $w_i$ is the weighing coefficient attached to the rule that inferred the fuzzy set $f_i$.

On the one hand, the weighing coefficients permit a finer tuning of the desired behaviour of EM but, on the other hand, their empirical determination is tedious especially when the number of linguistic rules increases. For this reason, we decided to automate their determination through the use of supervised learning [6] . This point is detailed in §4.6.

## 4.5 EM's Knowledge Base

As mentioned above, the global behaviour of the vehicle is obtained through the combination of several basic behaviours, each of which is encoded by a specific set of linguistic rules. Recall that the main purpose of EM is to follow the nominal trajectory provided by the trajectory planner while reacting in real-time to unexpected events, i.e. mostly by avoiding collision with unexpected obstacles. Accordingly, two basic behaviours were developed first: *trajectory following* and *obstacle avoidance*. They are respectively presented in §4.5.1 and §4.5.2. Other basic behaviours were added afterwards; to meet the requirements of the highway code in particular (see §4.5.3). EM commands the longitudinal acceleration and the steering velocity of the car-like vehicle considered. Each basic behaviour[5] takes as input a particular subset of the output variables of the vehicle (sensory data, internal state, etc.). The input of each basic behaviour is presented along with the behaviour in the next sections. The linguistic rules corresponding to each basic behaviours are not included here[6]. Instead the principles underlying the different basic behaviours are outlined and the way they work is illustrated by results obtained on a car-like vehicle simulator[7]. This simulator was essential in the design of EM; it was used as a primary test and debugging tool. It allowed a first validation of EM before the experiments on the real vehicle (see §5).

### 4.5.1 Trajectory Following

The primary purpose of EM is to ensure that the vehicle follows the nominal trajectory provided by the trajectory planner. Trajectory following is therefore the first behaviour we designed; it contains twenty linguistic rules. The nominal trajectory is a time-ordered sequence of states, i.e. (configuration[8], velocity) couples. At each time step, EM is activated

[5] More precisely, the linguistic rules associated with it.

[6] The reader is referred to [10] for a more detailed presentation of EM's knowledge base.

[7] This simulator was developed under the ACT robot simulation package.

[8] In the case of a car-like vehicle, a configuration is defined by a triple $(x, y, \theta)$, where $(x, y)$ is the position of a vehicle's reference point and $\theta$ the vehicle's orientation.

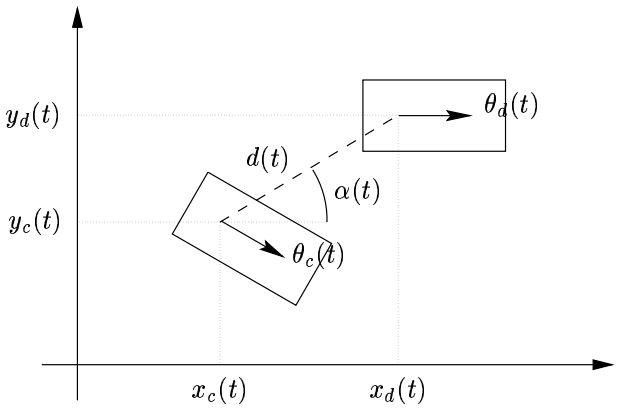


Figure 7: current and desired configurations of the vehicle at time $t$. $(x_c(t), y_c(t), \theta_c(t))$ denotes the current position and orientation of the vehicle, whereas $(x_d(t), y_d(t), \theta_d(t))$ denotes the desired one.

and selects the command, i.e. an (acceleration, steering velocity) couple, that minimize the error between the current state of the vehicle and the desired one. This minimization is achieved thanks to three sets of rules whose respective purposes are to minimize the error in position, orientation and velocity.

The first set of rules aims at minimizing the error in position. It takes as input the angular error $\alpha(t)$ and the distance $d(t)$ between the current position of the vehicle and the desired one at time $t$ (Fig. 7). It outputs acceleration and steering velocity commands. It includes rules of the following type:

— `if` ($\alpha(t)$ `is positive-low`)
`then (steering is left-low)`
— `if` ($d(t)$ `is positive-high`)
`then (acceleration is positive-high)`

where `positive-low`, `left-low`, etc., denote fuzzy sets associated with the different input and output variables of the vehicle.

The second set of rules aims at minimizing the error between the current orientation of the vehicle and the desired one. It takes as input $\Delta\theta(t)$, i.e. the difference between the current orientation of the vehicle and the desired one at time $t$ (Fig. 7). It outputs a steering velocity command thanks to rules such as:

— `if` ($\Delta\theta(t)$ `is positive-low`)
`then (steering is left-low)`

The third and last set of rules aims at minimizing the error between the current velocity of the vehicle and the desired one. It takes as input $\Delta v(t)$, i.e. the difference between the

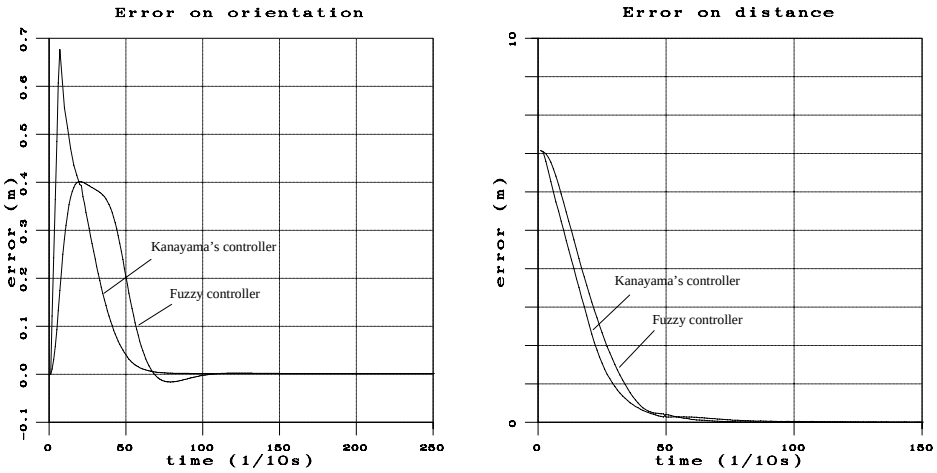


Figure 8: trajectory following with EM and Kanayama's controller.

current velocity of the vehicle and the desired one at time $t$. It outputs an acceleration command thanks to rules such as:

— `if` $(\Delta v(t)$ `is negative-low)`
`then (acceleration is negative-low)`

Of course there is no way to theoretically prove the convergence or stability of the trajectory following behaviour of EM. However, in an attempt to validate EM, it was compared with a trajectory following controller based upon a control theoretic approach, namely Kanayama's controller [12]. Both controllers were implemented and tested in simulation. Fig. 8 depicts a typical simulation result: given a straight nominal trajectory to be followed at constant speed, and an initial configuration error in position and orientation, the two charts show the evolution of the error in distance and in orientation. As one can see, the results are rather similar. However our controller has two advantages over Kanayama's: a) it can follow a trajectory including stops and b) it can easily incorporate other behaviours.

### 4.5.2 Obstacle Avoidance

Recall that the nominal trajectory relies upon hypotheses made about the future evolution of the environment and, in particular, the future behaviour of the moving obstacles (§3). Since moving obstacles may 'misbehave', avoiding collision with them is also a fundamental part of EM's task, hence the obstacle avoidance basic behaviour.

Obstacle avoidance requires a model of the actual environment of the vehicle. This model can be built from a priori information (maps of the stationary obstacles. etc.), or sensory

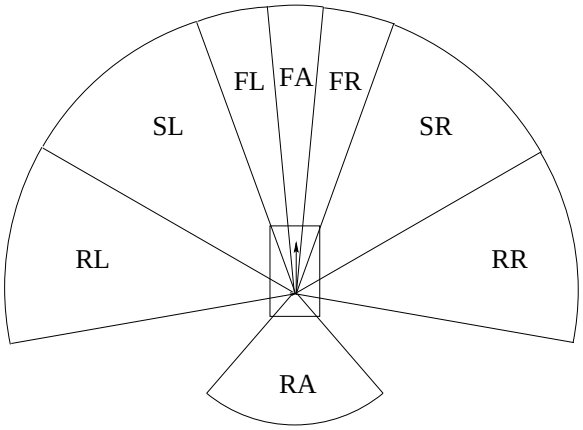


Figure 9: the eight polar regions of the vehicle's local environment.

data obtained on-line (information about the moving obstacles). Of course, for the sake of safety, this model must be updated at each time step.

Since getting a complete and accurate model of a given environment remains a costly process[9], the obstacle avoidance behaviour was designed so as to rely on a relatively poor but easy to get model of the local environment of the vehicle. This model is illustrated in Fig. 9. The local environment of the vehicle is divided in eight *polar regions*, and the only information used by the obstacle avoidance behaviour is the distance to the closest obstacle in the region. Poor as it may seem, this model proved sufficient to ensure obstacle avoidance.

The obstacle avoidance behaviour contains twenty linguistic rules that takes as input the current state of the vehicle and the model of its local environment. It outputs acceleration and steering velocity commands. It includes rules such as:

— `if (velocity is positive-high)`
`and (obstacle in FA is near)`
`and (obstacle in FL is near)`
`and (obstacle in FR is near)`
`then (acceleration is negative-high)`

After the obstacle avoidance behaviour was developed, it became possible to test the capabilities of EM regarding local adaptation of the nominal trajectory so as to react to unexpected events. In order to do so, several environments were considered. To begin with, EM was tested on simple environments such as the one depicted in Fig 10 that contains four square obstacles. A clothoïd curve to be followed at constant speed was provided to EM as a nominal trajectory. This trajectory was intersecting one of the squares and Fig 10 illustrates how the vehicle avoided the collision. Then, we considered more complex

[9]Especially when cameras are used.

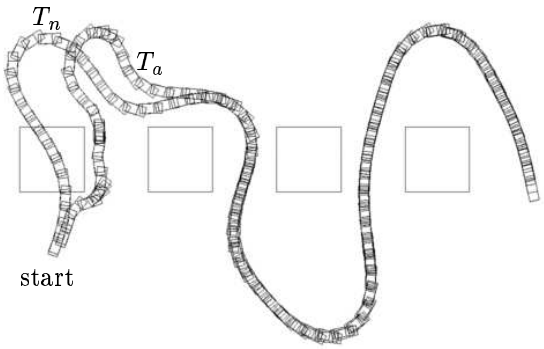


Figure 10: a collision avoidance test: $T_n$ (resp. $T_a$) denotes the nominal (resp. actual) trajectory of the vehicle.

environments such as the road network. Several scenarios including moving obstacles were designed, e.g. crossing an intersection, overtaking an other vehicle, etc. One of these scenario is illustrated in Fig. 11. Three vehicles are approaching an intersection. The white one is controlled by EM ('our vehicle'); the other two are regarded as moving obstacles. Based upon a prediction of the motion of the two other vehicles (constant speed), the trajectory planner computes a nominal trajectory such that our vehicle gives way to the vehicle on its left and crosses the intersection before the vehicle on its right. However, in the course of the execution of the trajectory, it turns out that the vehicle on the right suddenly accelerates. As soon as EM realizes that there is a risk of collision with the accelerating vehicle, obstacle avoidance becomes more important than trajectory following and our vehicle will stop so as to give way to the accelerating vehicle and then will reaccelerate so as to catch up with the nominal trajectory.

### 4.5.3 Additional Behaviours

EM includes additional basic behaviours. One of them in particular monitors whether it remains possible to catch up with the nominal trajectory. If not, i.e. if the difference between the current state and the desired one becomes too important, the trajectory planner is reinvoked so as to get a new nominal trajectory.

Drawing upon Zeghal's experience in the domain of collision avoidance between aircrafts [32], an 'avoid to the left' policy was added to the obstacle avoidance behaviour. In other words, when the vehicle faces a moving obstacle, it tries to get round it on the left side. Arbitrary as it may seem, this policy is nonetheless useful to reduce the deadlocks that may arise in situations such as the one depicted in the left-hand side of Fig. 12. In this situation, each vehicle has no choice but to avoid the other vehicle by turning to the right. If they both do so then a deadlock occurs. On the other hand, if they both (or, at least, one

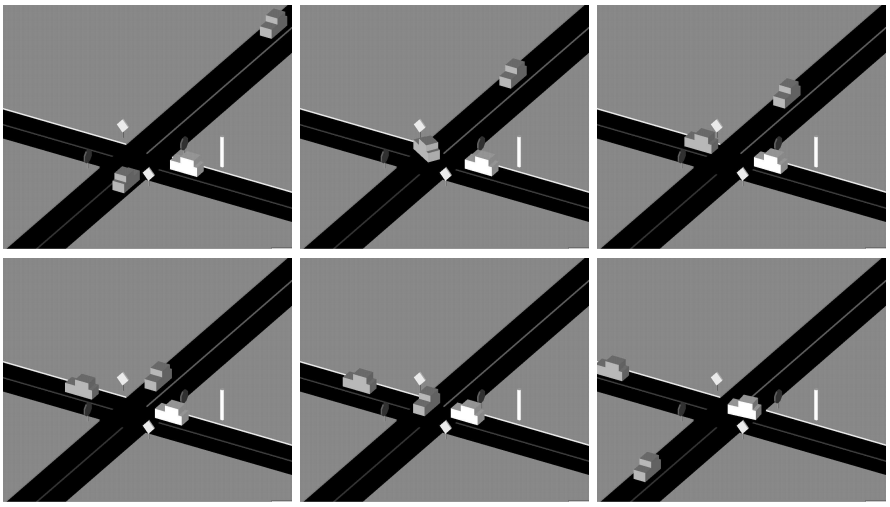

Figure 11: an intersection crossing scenario.

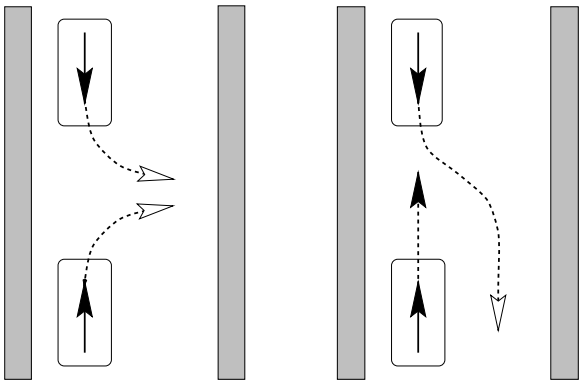

Figure 12: a deadlock situation and its resolution thanks to an 'avoid to the left' policy.

of them) follow an 'avoid to the left' policy then the conflict will be solved as illustrated in the right-hand side of Fig. 12.

Finally, we have added behaviours that implement certain highway-code regulations, e.g. give way to the vehicle coming to your right, or that take advantage of the current context, e.g. crossing an intersection. These behaviours are specific to the case of a vehicle evolving on the road network and they will not be detailed here (the reader is referred to [10] for more details).

## 4.6 Supervised Learning

As mentioned earlier in §4.4, *weighing coefficients* are attached to the linguistic rules. These weights represent the importance of each rule with respect to the others. They permit a finer tuning of the desired behaviour of EM. However their empirical determination may prove tedious especially when the number of linguistic rules increases. For this reason, we decided to automate their determination through the use of supervised learning [6] . In this type of learning, samples of the form (input, desired output) are presented to the learning process whose purpose is then to determine the value of the weights that minimize, for each sample, the error between the desired output and the output actually computed by EM.

As we will see in the next two sections, it is possible to cast the problem of learning the weights into the framework of linear programming and therefore to use a classical Simplex algorithm to solve the problem at hand efficiently [4].

### 4.6.1 Linear Programming

A linear program is an optimization problem in which:

1. The unknown variables of the problem have to satisfy a set of linear equations or inequations (the *constraints*).
2. The value of a linear function (the *objective function*) has to be optimized, i.e. minimized or maximized.

Typically, a linear program is formulated as:

$$\begin{cases} Ax = b \\ \min(f(x)) \end{cases}$$

where $A$ is a $m \times n$ matrix, $b$ a $m$-vector, $x$ an $n$-vector and $f$ an objective function [26]. $x$ is the vector of unknown variables and the goal is to find $x$ so that it satisfies the constraints while optimizing the objective function. Once a problem is formulated as a linear program, it is possible to use the Simplex algorithm to solve it efficiently [4].

### 4.6.2 The Learning Problem

Let $s_i$ be a sample, $s_i = (I_i, O_i)$ where $I_i$ is the input data of EM and $O_i$ the desired output data. Let $W$ be the set of weights attached to the linguistic rules of EM. Let $\mathrm{EM}(I_i, W)$ denote the output data actually computed by EM; it depends on the input data $I_i$ and the weights $W$. For a given sample $s_i = (I_i, O_i)$, it is then possible to write:

$$\mathrm{EM}(I_i, W) + \delta_i = O_i \tag{3}$$

where $\delta_i$ represents the error between the actual and the desired output data. Note that (3) is linear with respect to $\delta_i$ and the weights of $W$ (see Eq. (2) in §4.4); it may be viewed as a linear programming constraint.

Given $n$ samples, $\{\delta_1, \delta_2, \ldots, \delta_n\}$ is the set of errors between the actual and desired output data; it is the set of values that are to be minimized by an appropriate choice of $W$. In order to do so, the following objective function is defined:

$$\sum_{i=1}^{n}(|\delta_i| + n\max(|\delta_i|)) \tag{4}$$

This objective function, that is to be minimized, is the combination of two norms written in linear form, each of which has an interesting property: the left-hand side of the sum aims at minimizing each error individually while the right-hand side ensures a minimization of the worst error [5].

By choosing the errors $\delta_j$ and the weights of $W$ as unknown variables, (3) as the constraints, and (4) as the objective function, it is possible to cast the problem of learning the weights into the framework of linear programming (note that in this formulation, the objective function solely depends on the errors). Accordingly the Simplex algorithm can be used to solve the problem at hand, meaning that, given a set of samples $s_i$, it is possible to *efficiently* and *automatically* compute the weights of the linguistic rules of EM.

### 4.6.3 Learning Results

Testing the supervised learning algorithm requires a data base of learning samples $s_i = (I_i, O_i)$ , i.e. couples of (input, desired output). We decided to run EM (with empirically determined weights) on the different scenarios developed on our car-like vehicle simulator in order to generate the required data base. Then the weights of the linguistic rules of EM were reset to one and the samples were fed to the learning algorithm; it was found that the learned weights were close to the empirical ones (which is in some way reassuring). Next we run EM with different weights (unit, empirical and learned) on several scenarios so as to demonstrate the importance of the weights in the global behaviour of the vehicle. These experiments showed first that, even with unit weights, the vehicle behaves satisfactorily in the sense that it is able to avoid collisions while following its nominal trajectory (see Fig 13, top). Second they showed that empirical or learned weights do improve EM by smoothing the global behaviour of the vehicle (see Fig 13, middle and bottom).

Finally the convergence of the learning process was studied. Sets $S_i$ of increasing size of learning samples randomly selected from the data base were fed to the learning algorithm. The following criterion was chosen to show the stabilization of the learned weights:

$$\sum_j |w_j^{i+1} - w_j^i| \tag{5}$$

where $w_j^i$ is the $j^{th}$ learned weight for the set $S_i$ of learning samples. Fig. 14 depicts one example of the evolution of this criterion for sets $S_i$ of size 100 to 2000. Several experimental runs showed a convergence of the learned weights after about 1100 samples.

The outcome of these experiments is a primary validation of the learning process. They show that the weighing coefficients of EM's rules can be determined automatically through

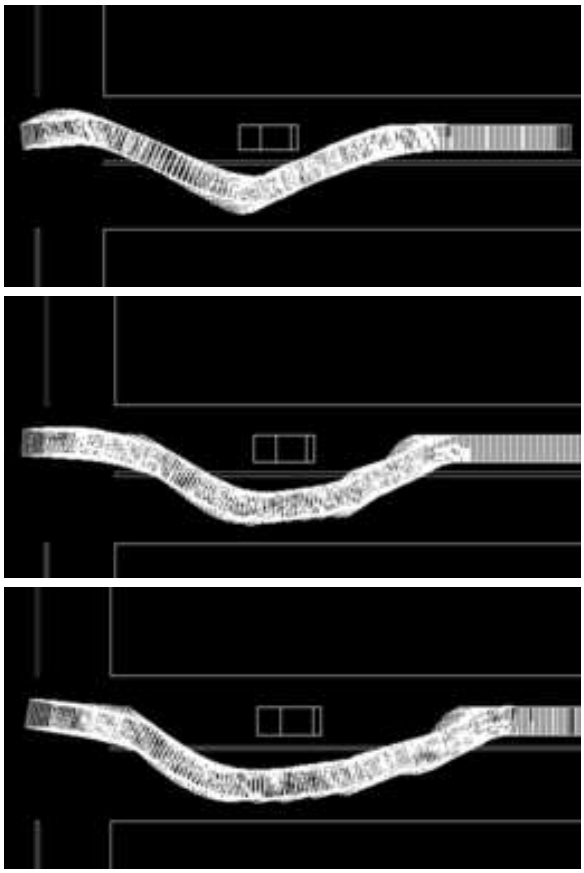

Figure 13: EM's behaviour during an overtaking manœuvre: with unit weights (top), empirical weights (middle) and learned weights (bottom).

supervised learning. However more experiments are necessary. In particular actual learning samples obtained thanks to a real vehicle equipped with sensors should be used to further test the learning process.

# 5 Experimental Results

Up until now, EM has been developed and tested on a car-like vehicle simulator. The next step was to use a real vehicle. We have started to work with an electric car that we are developing within the framework of Praxitèle, a French research programme on road transport. The next four sections will respectively present the Praxitèle programme, the

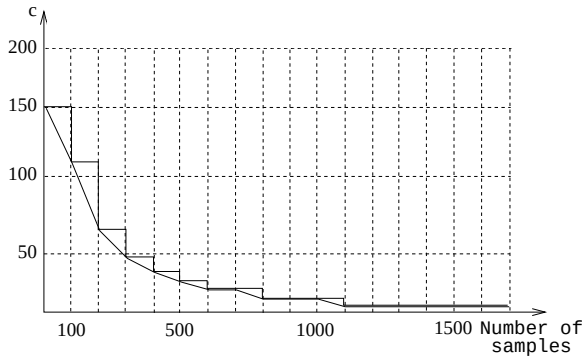


Figure 14: convergence of the learned weights.

experimental vehicle used, the experiments carried out so far and the future developments planned.

## 5.1 The Praxitèle Programme

Praxitèle is a French research programme that investigates the concept of Public Individual Transport. It is a novel transportation system based upon a fleet of small public cars that are supervised by a central computer [22, 2, 21]. Its aim is to develop a complement to public transport which would be a better alternative to the use of private automobiles in urban areas in terms of congestion, pollution and ease of use. Such a public transportation system can be an extension of a public network where the demand does not justify the investment and the operation of large capacity systems. It can also be justified for local trips in specific neighborhoods, or as an alternative to the development of 'park and ride' systems. Within such a transport system, the cars are driven by their users but their operation will be automated in certain circumstances, e.g. to move empty cars where they are needed, or within specific environments, e.g. parking-maintenance stations, special tracks, etc. This programme, that is highly multidisciplinary, is a perfect opportunity for us to test our motion control architecture in real situations.

## 5.2 The Experimental Vehicle

The experimental validation of our control architecture, EM included, is carried out on a prototype based upon a commercial electric car, namely a Ligier Optima (Fig. 15). It is a 2.5 m long and 1.4 m wide vehicle that weighs around 650 kg. It can accommodate two people. Like a regular car, it has two rear wheels and two directional front wheels with a maximum steering angle of 23 deg. It is equipped with a 12 kw asynchronous electric motor powered by lead batteries that can drive the car up to 70 km/h with an average range of 80 km.

Figure 15: the Ligier prototype.

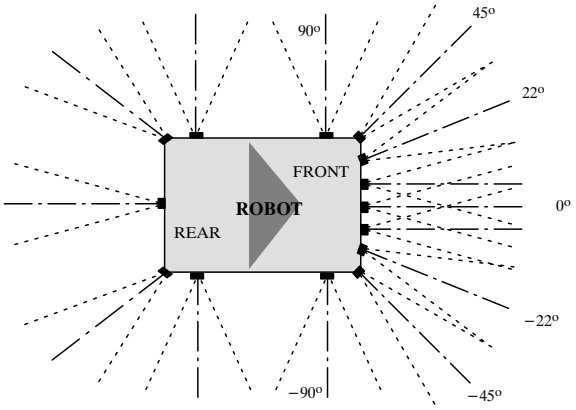


Figure 16: the ultrasonic sensors layout.

The control system of the vehicle includes a Motorola VME bus with a MVME 162 CPU board (68040 processor). This board drives three servo-motors and a three-phase controller. One servo-motor controls the steering wheel, the other two are in charge of the brakes. The three-phase controller drives the electric motor. As far as proprioception is concerned, an optical encoder measures the steering angle while two optical encoders mounted on the rear wheels provide the longitudinal velocity of the car and a position estimation (odometry).

The ligier is equipped with a range measurement system of fourteen Polaroid 9000 ultrasonic sensors whose layout is depicted in Fig. 16. These sensors can detect obstacles in a range of 10 m. with a 1 cm. resolution. They are driven and synchronized thanks to a

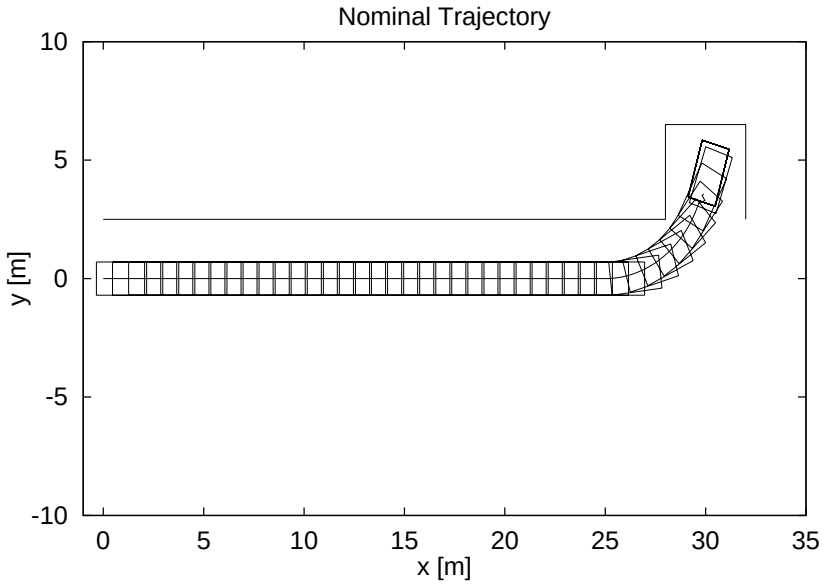


Figure 17: a trajectory following experiment.

custom-made board including a network of four transputers dynamically linked by a C004 integrated programmable circuit.

As far as software is concerned, the ORCCAD[10] software [27] has been used to implement the different components of our control architecture.

## 5.3 Trajectory Following With Obstacle Avoidance

Because testing behaviours such as the 'avoid to the left' or 'give way to the right' policies requires at least two vehicles, our experiments have focused primarily on testing the trajectory following and obstacle avoidance behaviours. The limited range of the ultrasonic sensors further restricted us to slow speed (of the order of 1 m/s). An actual parking lot was chosen to carry out the experiments. A 40 m. long nominal parking trajectory was computed and fed to EM. At first the trajectory following behaviour was tested alone. Fig 17 shows a bird view of the trace of the experiment.

Then unexpected obstacles were introduced in order to test the obstacle avoidance behaviour in various situations. The outcome of an experiment with an unexpected obstacle right on the path of the vehicle is depicted in Fig 18. The bottom part Fig 18 shows the actual velocity followed by the vehicle. Various snapshots of this experiment are presented in Fig. 19.

[10]Open Robot Controller Computer-Aided Design.

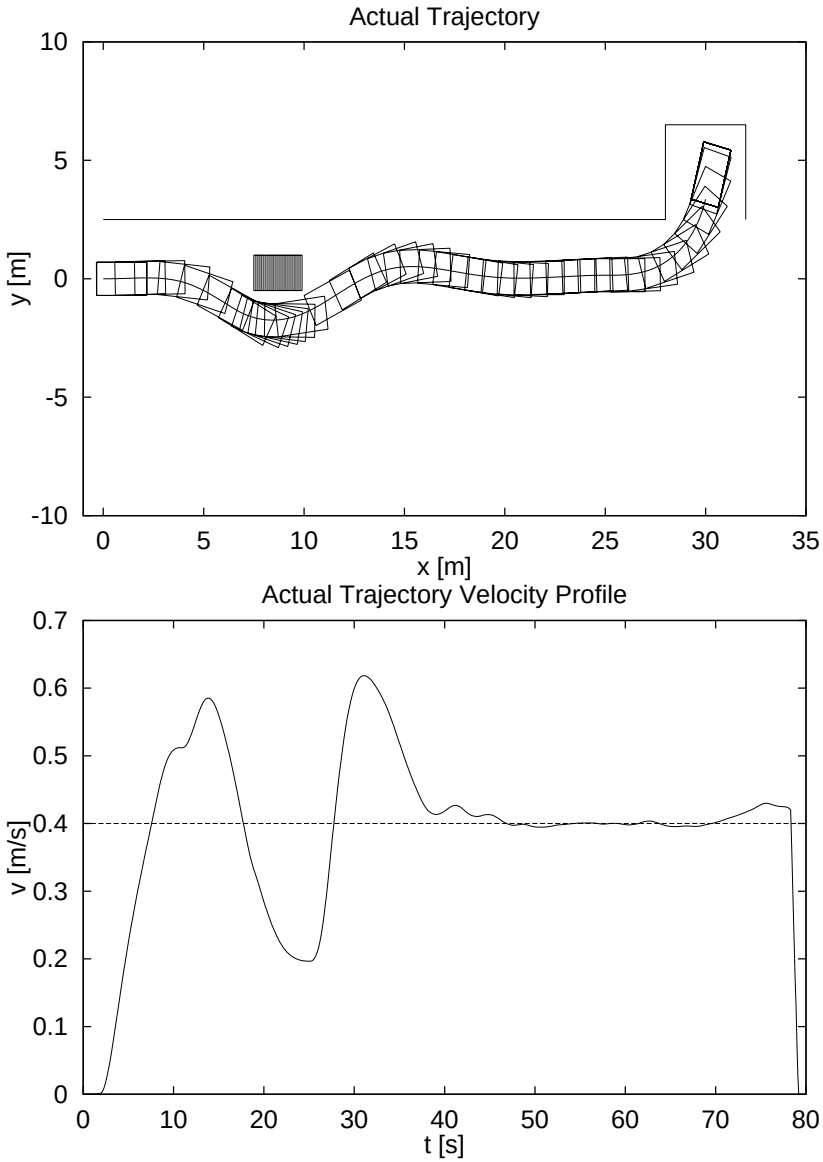


Figure 18: an obstacle avoidance experiment : trace of the experiment (top), and actual velocity profile followed by the vehicle (bottom).

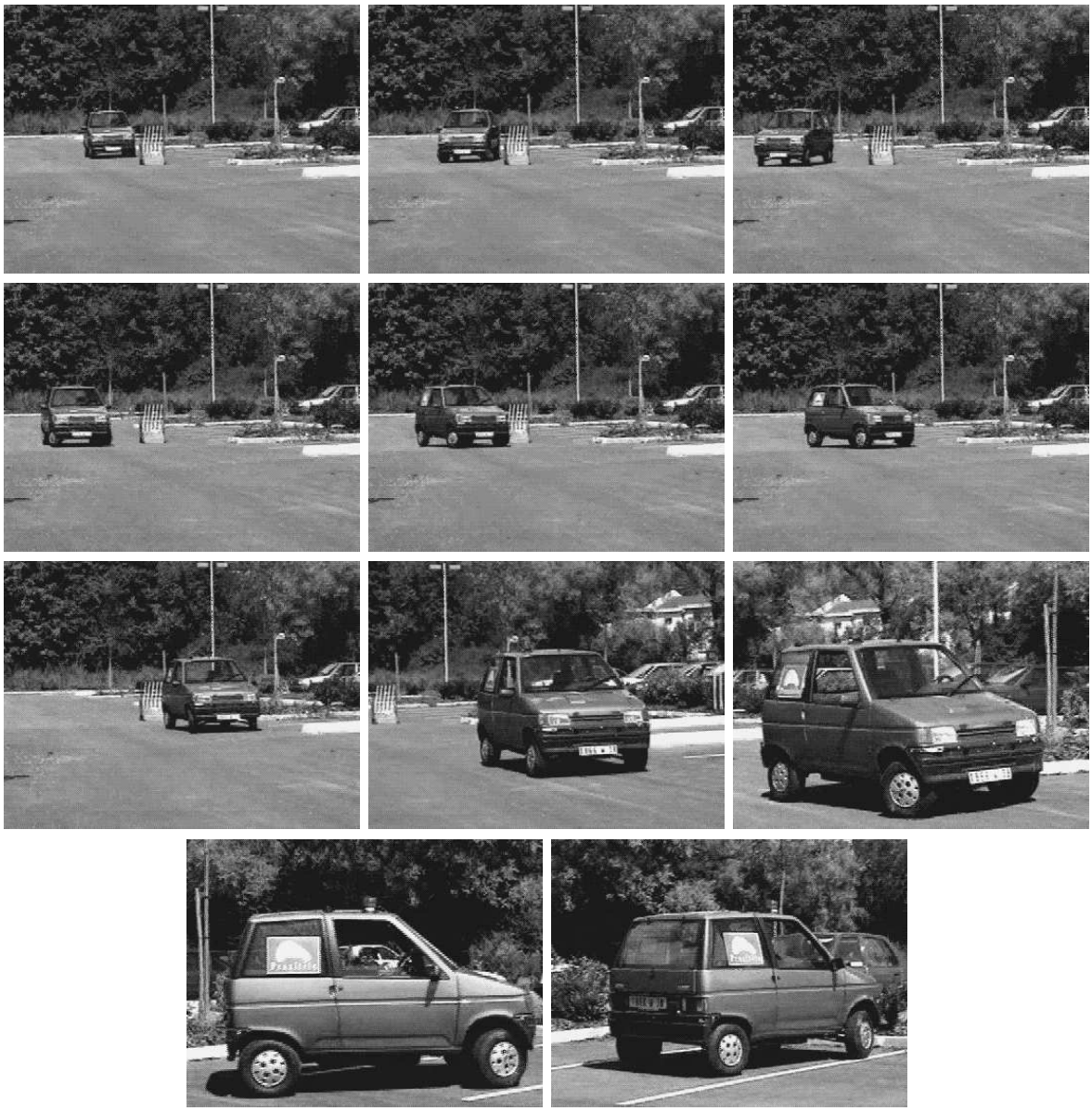

Figure 19: snapshots of the experiment depicted in Fig. 18.

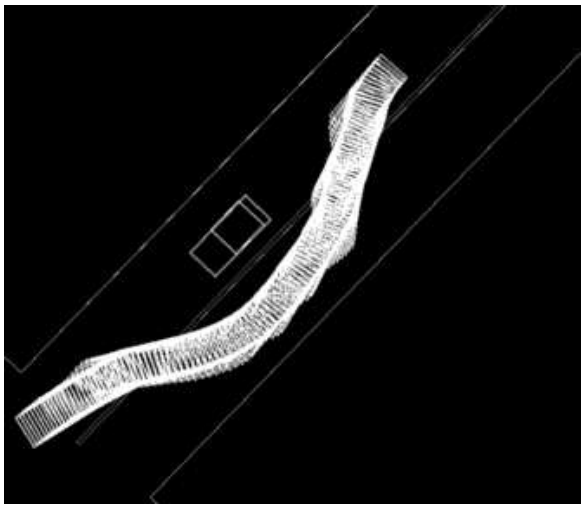

Figure 20: an oscillatory behaviour.

## 5.4 Future Developments

As mentioned earlier, the limited range of the ultrasonic sensors mounted on our experimental vehicle restricted us to slow speed (of the order of 1 m/s). The next step is to test EM at higher speed. In order to do so, sensors with a better range are required. We are considering the use of a linear camera coupled with an infrared flashlight. Provided that the obstacles are equipped with reflectors (it is the case for other cars), it will be possible to detect them from a distance of up to 30m. Thus, it should enable us to carry out experiments at higher speed (of the order of 3 m/s).

Regarding the experiments carried out so far, the overall behaviour of EM proved satisfactory. However a problem appeared in certain situations. This problem is depicted in Fig. 20 where our vehicle has to avoid a vehicle which is directly on its nominal trajectory. In the course of the overtaking manœuvre, a competition arises between trajectory following and obstacle avoidance. The result of this competition is an oscillatory behaviour: at one time step, EM detects an obstacle on its right and decides to steer to the left in order to avoid it. But at the next time step, the obstacle becomes less dangerous so EM decides to steer to the right in order to catch up with the nominal trajectory and so forth. Several reasons account for this oscillation problem (limitation of the sensory data, crude characterization of the fuzzy sets associated with the steering-acceleration commands) but it is mostly due to the reflex nature of EM.

To deal with this problem, we have decided to introduce an intermediate level between the trajectory planner and EM: the *local navigator* [20]. The purpose of the local navigator is to compute a set of so-called 'escape-lines', i.e. trajectories obtained by applying simple commands (typically constant) to the vehicle over a fixed period of time. These escape lines

are then checked for possible collision with the obstacles detected by the vehicle's sensors. Finally the best (typically the closest to the nominal trajectory) collision-free escape line is selected as the trajectory to be followed by EM over the next period of time. The activation frequency of the local navigator is one order of magnitude lower than that of the execution monitor so as to ensure continuity in the vehicle's behaviour and to avoid oscillations. A detailed presentation of the local navigator and a report on preliminary experiments can be found in [19].

In addition, we have started to experiment with the concept of *generic manœuvres*. A generic manœuvre is a parameterized trajectory, local in nature, that can be used in response to a specific situation. This concept seems very appropriate to a structured environment such as the road network. In this case, it is indeed straightforward to consider manœuvres such as lane following, lane changing, overtaking, parking, etc. These manœuvres can then be used whenever the current situation requires it. This concept along with preliminary results are presented in [23].

# 6 Conclusion

This paper has presented the *execution monitor*, i.e. the reactive component of a motion control architecture for a car-like vehicle. The execution monitor (EM) fulfills the following functions:

- Generating the commands for the actuators of the vehicle so as to follow a nominal trajectory as closely as possible.
- Monitoring the execution of the trajectory.
- Reacting in real-time to unexpected events by locally adapting the trajectory actually followed by the vehicle.

EM is designed as a fuzzy controller, i.e. a reactive system based upon fuzzy logic [16]. Its main components are an inference engine and a set of 'linguistic rules' that encode the reactive behaviour of the vehicle. The global behaviour of the vehicle results from the combination of several basic behaviours (trajectory following, obstacle avoidance, etc.), each of which is encoded by a specific set of rules. EM differs from classical fuzzy controllers in two novel ways: to begin with, we have introduced a new defuzzification technique, the *Barycentre of the Centres Of Area*, that permits to better take into account the influence of each and every rule. The second difference lies in the fact that weighing coefficients are attached to the rules thus permitting a fine tuning of the influence of each basic behaviour. We borrowed this idea from [25] but took it a step further by using supervised learning [6] to automate the determination of these weights thus suppressing the ever delicate problem of finding such coefficients.

At first, EM was designed and tested on a car-like vehicle simulator. Then it was implemented on an electric vehicle prototype that we are currently developing within the

framework of Praxitèle, a French research programme on road transport. Promising results of preliminary experiments featuring trajectory following and unexpected obstacle avoidance have been presented. Further experiments are underway.

### Acknowledgements

This work was partially supported by the Inria-Inrets[11] Praxitèle programme on individual urban public transport.

## References

[1] R. C. Arkin. Motor schema based navigation for a mobile robot. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 264–271, Raleigh (NC), April 1987.

[2] D. Augello, E. Bénéjam, J.-P. Nerrière, and M. Parent. Complementarity between public transport and a car sharing service. In *Proc. of the World Congress on Application of Transport Telematics and Intelligent Vehicle Highway System*, Paris (FR), November–December 1994.

[3] R. A. Brooks. A robust layered control system for a mobile robot. In G. Shafer and J. Perl, editors, *Readings in Uncertain Reasoning*, pages 204–213. Morgan Kaufmann, 1990.

[4] G. B. Dantzig. *Linear programming and extensions.* Princetown University Press, 1963.

[5] J. Dixmier. *Cours de mathématiques du premier cycle.* Gauthier-Villars, 1976.

[6] S. Faibish. Neural reflexive control of a mobile robot. In *Proc. of the IEEE Int. Symp. on Intelligent Control*, pages 144–146, Glasgow (GB), August 1992.

[7] Th. Fraichard and C. Laugier. On line reactive planning for a non holonomic mobile in a dynamic world. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 432–437, Sacramento (CA), April 1991.

[8] Th. Fraichard and C. Laugier. Dynamic trajectory planning, path-velocity decomposition and adjacent paths. In *Proc. of the Int. Joint Conf. on Artificial Intelligence*, volume 2, pages 1592–1597, Chambéry (FR), September 1993.

[9] Th. Fraichard and A. Scheuer. Car-like robots and moving obstacles. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, volume 1, pages 64–69, San Diego (CA), May 1994.

[11] Institut National de REcherche sur les Transports et leur Sécurité.

[10] Ph. Garnier. *Contrôle d'exécution réactif de mouvement de véhicules en environnement dynamique structuré.* Thèse de doctorat, Inst. Nat. Polytechnique de Grenoble, Grenoble (FR), December 1995.

[11] M. Hassoun and C. Laugier. An architecture for motion planning and motion control of a car-like vehicle. *Mathl. Comp. Modelling*, 22(4–7):329–341, 1995.

[12] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi. A Stable Tracking Control Method for a Non-Holonomic Mobile Robot. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1236–1241, Osaka (JP), November 1991.

[13] M. Khatib and R. Chatila. An Extended Potential Field Approach For Mobile Robot Sensor-Based Motions. In *Proc. of Intelligent Autonomous Systems*, pages 490–496, March 1995.

[14] P. M. Larsen. Industrial applications of fuzzy logic control. *Int. Jour. of Man-Machine Studies*, 12, 1980.

[15] J.-C. Latombe. *Robot motion planning.* Kluwer Academic Press, 1990.

[16] C. C. Lee. Fuzzy logic in control systems: fuzzy logic controller—part I and II. *IEEE Trans. Systems, Man and Cybernetics*, 20(2):404–418,419–435, March 1990.

[17] E. H. Mamdani. Applications of fuzzy algorithms for simple dynamic plants. *Proc. IEEE*, 121(12):1585–1588, 1974.

[18] N. J. Nilsson. Shakey the robot. Technical Report 347, Artificial Intelligence Center, SRI International, Memlo Park (CA), 1984.

[19] C. Novales, D. Pallard, and C. Laugier. Controlling the motions of an autonomous vehicle using a local navigator. In *Proc. of the Int. Symp. on Robotics and Manufacturing*, Montpellier (FR), May 1996.

[20] C. Novales and R. Zapata. A local architecture for controlling the movements of fast mobile robots. In *Proc. of the Int. Symp. on Robotics and Manufacturing*, Hawaï (USA), 1994.

[21] M. Parent and P. Daviet. Automated urban vehicles: towards a dual mode PRT (Personnal Rapid Transit). In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, volume 4, pages 3129–3134, Minneapolis (MN), April 1996.

[22] M. Parent and P.-Y. Texier. A public transport system based on light electric cars. In *Proc. of the Int. Conf. on Automated People Movers*, Irving (TX), March 1993.

[23] I. Paromtchik, Ph. Garnier, and Ch. Laugier. Autonomous maneuvers of a nonholonomic vehicle. In *Proc. of the Int. Symp. on Experimental Robotics*, Barcelona (ES), June 1997.

[24] D. W. Payton. An Architecture for Refexive Autonomous Vehicle Control. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, San Franciso (CA), April 1986.

[25] F. G. Pin, H. Watanabe, J. Symon, and R. S. Pattay. Autonomous Navigation of a Mobile Robot Using Custom-Designed Qualitative Reasoning VLSI Chips and Boards. *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1992.

[26] M. Sakarowitch. *Linear programming.* Springer Verlag, 1983.

[27] D. Simon, B. Espiau, K. Castillo, and K. Kapellos. Computer-aided design of a generic robot controller handling reactivity and real-time control issues. *IEEE Trans. Control Systems Technology*, pages 213–229, December 1993.

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

[29] J. Yen and N. Pfluger. A Fuzzy Logic Based Extension to Payton and Rosenblatt's Command Fusion Method for Mobile Robot Navigation. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(6):971–978, June 1995.

[30] L. A. Zadeh. Outline of a New Approach to the Analysis of Complex Systems and Decision-Making Approach. *IEEE Trans. on Systems, Man, and Cybernetics SME-3(1)*, pages 28–45, January 1973.

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

[32] K. Zeghal and J. Ferber. CRAASH : A Coordinated Collision Avoidance System. In *European Simulation Multiconference*, Lyon (FR), June 1993.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399