Int. Conf. on Intelligent Autonomous Systems
July 25–27, 2000. Venezia (IT). pp. 1033–1040

Using Artificial Neural Networks to Improve Sensor-Based Maneuvers for a Car-Like Vehicle

F. Large, J. Hermosillo, S. Sekhavat, C. Laugier
Inria[a] Rhône-Alpes & Gravir[b]
655 av. de l'Europe, 38330 Montbonnot, France
`Frederic.Large@inrialpes.fr`

July 2000

Abstract — This paper presents an Artificial Neural Network (ANN) approach aiming to improve the performance of sensor-based maneuvers. The key idea is to adapt the parameters of the controller accordingly to the perception of the *real* state of the vehicle from noisy and uncertain sensor data. A particular implementation is discussed around a case study consisting in following a nominal trajectory and avoiding unexpected obstacles. Experimental results obtained are presented to illustrate the advantages of our approach.

Keywords — mobile-robot, artificial-neural-networks, adaptive-learning.

Acknowledgements — This work was partially supported by the French programme "La Route Automatisée": `<http://www.lara.prd.fr/>`.

[a]Institut National de Recherche en Informatique et en Automatique.
[b]Lab. Graphisme, Vision et Robotique.

Using Artificial Neural Networks to Improve Sensor-Based Maneuvers for a Car-Like Vehicle

F. Large, J. Hermosillo, S. Sekhavat, C. Laugier
Inria[*]*Rhône-Alpes & Gravir*[†]
655 av. de l'Europe. 38330 Montbonnot, France
Frederic.Large@inrialpes.fr

Abstract
This paper presents an Artificial Neural Network (ANN) approach aiming to improve the performance of sensor-based maneuvers. The key idea is to adapt the parameters of the controller accordingly to the perception of the real *state of the vehicle from noisy and uncertain sensor data. A particular implementation is discussed around a case study consisting in following a nominal trajectory and avoiding unexpected obstacles. Experimental results obtained are presented to illustrate the advantages of our approach.*

1 Introduction

Motion autonomy for various types of vehicles has already been widely studied. The state of the art on this topic shows approaches of various complexity, combining in different ways purely reactive methods with more traditional hierarchical decisional schemes. A generalized approach to solve the motion autonomy problem for a car-like vehicle moving in a partially known environment is to combine an *off-line global path/trajectory planner*, with a *reactive execution controller* capable to track the nominal trajectory while avoiding collisions with unexpected obstacles.

In a previous approach, we have used a fuzzy behavior merging process for combining a trajectory tracking behavior with a collision avoidance behavior. However, this approach lead to an oscillatory motion of the vehicle [2] and was not able to guarantee that all the constraints (goal, timing, kinematic and dynamic constraints) would be satisfied. In recent work, we presented a novel control architecture [7], aiming to generate *smooth and safe motions* for an autonomous car-like vehicle while satisfying multiple goals and constraints. In this architecture the reactivity of the system relies on the paradigm of *sensor-based maneuver* (*SBM*). A similar approach has been developed in [8] for the AMR robot at Laas.

This paper presents an Artificial Neural Network (ANN) approach aiming to improve the performance of sensor-based maneuvers for autonomous navigation. We argue that using a controller based on a simplified model of the real vehicle, requires good model matching and parameter identification which are classical drawbacks in this kind of approach; specially when dealing with "non academic" experimental platforms such as a real car like ours.

Our approach consists in using a more flexible control system, which does not compute data from an approximative analytical model anymore, but from the real vehicle itself. The key idea is to adapt the parameters of the controller accordingly to the perception of the *real* state of the vehicle.

The paper is organized as follows: Section 2 outlines the framework of our work, describing our wheeled mobile robot and a basic maneuver of following a nominal trajectory and avoiding unexpected obstacles. In section 3 we discuss some weakness of the model based approaches (including ours) regarding the robustness, and we describe how we try to improve the performance of our system using *Artificial Neural Networks*. Finally, experimental results are discussed in section 4.

2 Framework of study

2.1 Simplified model of the Vehicle

Our experimental platform consists in a mobile robot called the Cycab (See Fig. 1a). Kinematically, the Cycab differs from a classical car in that all wheels are steerable. A steering angle of ϕ on the front wheels induces a steering angle of $-k\phi$ on the rear wheels (k is approximatively constant).

[*]Inst. Nat. de Recherche en Informatique et en Automatique.
[†]Lab. Graphisme, Vision et Robotique.

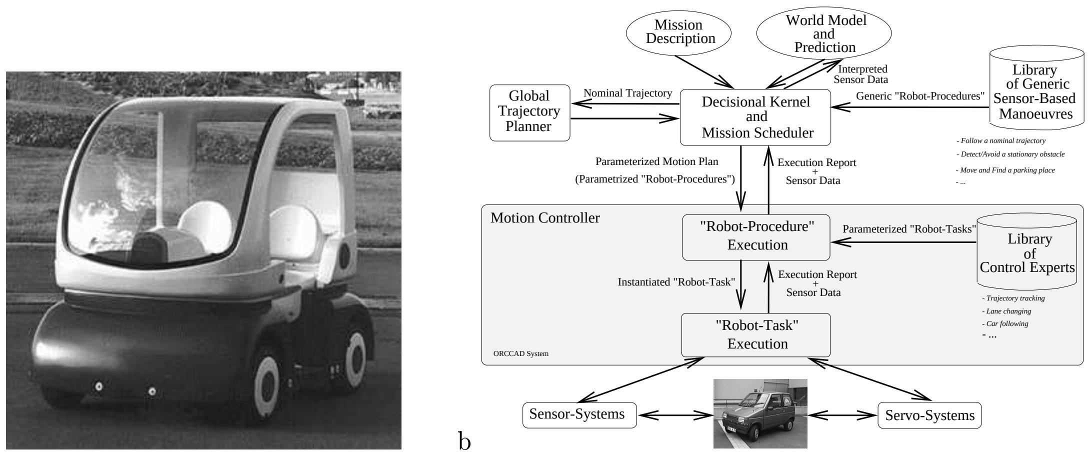
a  b
Mission Description
World Model and Prediction
Interpreted Sensor Data
Global Trajectory Planner
Nominal Trajectory
Decisional Kernel and Mission Scheduler
Generic "Robot-Procedures"
Library of Generic Sensor-Based Manoeuvres
- Follow a nominal trajectory
- Detect/Avoid a stationary obstacle
- Move and Find a parking place
- ...
Parameterized Motion Plan (Parametrized "Robot-Procedures")
Execution Report + Sensor Data
Motion Controller
"Robot-Procedure" Execution
Parameterized "Robot-Tasks"
Library of Control Experts
Instantiated "Robot-Task"
Execution Report + Sensor Data
- Trajectory tracking
- Lane changing
- Car following
- ...
"Robot-Task" Execution
ORCCAD System
Sensor-Systems
Servo-Systems

Figure 1: Our framework of study: (a) Our experimental platform : The Cycab vehicle, (b) Our control architecture implementing the SBM paradigm.

The following equations represent the kinematic model of the Cycab :

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

where x and y are the coordinates of the front axle midpoint, θ is the orientation of the vehicle and L is the wheel base. The controls are ϕ the steering angle and v the velocity of the front wheels. Also, bounds on velocity and acceleration are considered by the planning and control functions.

2.2 Sensor-Based Maneuvers

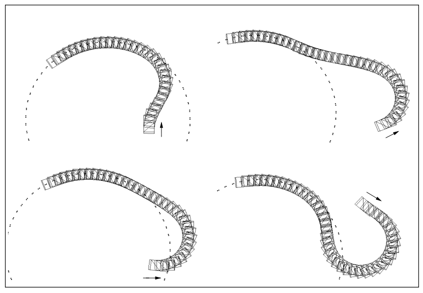
Our work fits into the frame of our control architecture (see Fig. 1b) implementing the *sensor-based maneuver* (*SBM*) paradigm as described in [7]. The architecture includes an off-line global trajectory planner[1], a decisional kernel that selects appropriate *Sensor-Based Maneuvers (SBM)* in real-time, and a reactive motion controller that makes use of a set of *Control Experts*[2] to execute the required sensor-based maneuvers. The underlying idea is to select an appropriate *generic SBM* in a library, and to instantiate this generic *SBM* into an executable maneuver using a set of *Control Experts* and of *sensor operations* (see Fig. 1b).

A *SBM* is basically a *safe and smooth motion* of the vehicle, which is executed using some predefined sensor modalities and controls; it allows the vehicle to perform in a reactive way a particular type of maneuver, while adapting the control parameters to the current execution context. The *SBM*s are actually implemented using Orccad software [11] (a *SBM* corresponds to a "robot-procedure" in Orccad formalism).

A basic SBM for any mobile robot is "Following a Nominal Trajectory". The purpose of this *SBM* is to allow the robot to follow a reference trajectory as closely as possible, while reacting appropriately to any unforeseen obstacle obstructing the way of the robot. 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 accordingly to the context (sensor data), in order to satisfy a set of different constraints: collision avoidance, time constraints, kinematic and dynamic constraints. In a previous approach, we have used a fuzzy behavior merging process for combining a trajectory tracking behavior with a collision avoidance behavior. However, this approach has exhibited non consistent behaviors: it generates oscillations, and it does not guaranty that all the previous constraints are always satisfied [2]. The current *SBM* approach makes use of *smooth local trajectories* for avoiding 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 these local trajectories verify the motion constraints and enable the vehicle to follow a smooth path. This type of maneuver is executed using two Control Experts: "trajectory tracking" and "lane changing".

[1]A trajectory represents both a geometric path (*i.e.* a smooth curve) and its associated velocity profile.

[2]A *Control Expert* is a parameterized control program adapted to the execution of a particular type of sensor-based motion.

a 

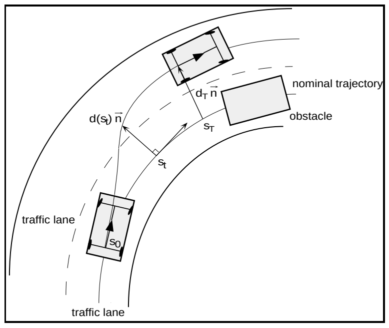
b 

Figure 2: Trajectory tracking and Lane changing: (a) Examples of intermediate paths when the vehicle is far from the reference trajectory.(b) Generation of smooth local trajectories to avoid an obstacle.

2.2.1 Trajectory Tracking

In order to realize the trajectory tracking, we impose on the middle point of the front axle to follow its reference trajectory. To do so, we use Kanayama's control law [4] applied to the (x, y) point (see Eq. 1) which guarantees the stable convergence of (x, y) to their reference values. One can show for the case of the car-like for instance, that in forward motion and for the practical range of use, this approach will also induce the convergence of the car orientation to its reference value [6]. Therefore the vehicle controls are deduced from the model of the robot and the following equations :

$$\dot{\alpha} = \dot{\alpha}_{ref} + v_{ref}(k_y y_e + k_\alpha \sin\alpha_e), \tag{2}$$

$$v = v_{ref}\cos\alpha_e + k_x x_e, \tag{3}$$

where $\alpha = \theta + \phi$ is the absolute angle of the front wheels, $(x_e, y_e, \alpha_e)^T$ represents the error vector and k_x, k_y, k_α are positive constants. The actual controls of the robot are v and $\dot{\phi} = \dot{\alpha} - v\frac{\sin(\phi+k\phi)}{L\cos(k\phi)}$.

When the reference trajectory is considered too far, a second degree polynomial function is used to calculate an intermediate tracked path, tangentially connected to the initial trajectory (see Fig.2a). The criterion used in this case is the difference between the maximum allowed linear and angular speeds and the commands generated by the Kanayama's control law. The intermediate trajectory is calculated only if that difference is negative.

2.2.2 Lane Changing

This maneuver is carried out by generating and tracking an appropriate smooth local trajectory. Let $\mathcal{T}$ be the nominal trajectory to track (see Fig. 2b), d_T be the distance between $\mathcal{T}$ and the middle line of the free lane to reach, s_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. A feasible smooth trajectory for executing a lane change can be obtained using the following quintic polynomial [10]:

$$d(s) = d_T\left(10\left(\frac{s}{s_T}\right)^3 - 15\left(\frac{s}{s_T}\right)^4 + 6\left(\frac{s}{s_T}\right)^5\right), \tag{4}$$

Then, the minimal value required for s_T can be estimated as follows:

$$s_{T,min} = \frac{\pi\sqrt{k\,d_T}}{2\,\mathcal{C}_{max}}, \tag{5}$$

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

$$\mathcal{C}_{max} = \mathbf{min}\left\{\frac{\tan(\phi_{max})}{L}, \frac{\gamma_{max}}{v^2_{R,ref}}\right\}, \tag{6}$$

γ_{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 θ_{ref} is converted into $\theta_{ref} + \arctan(\partial d(s_t)/\partial s)$, and the reference velocity $v_{R,ref}$ is obtained using the following equation:

$$v_{R,ref}(t) = \frac{dist(p_{ref}(t), p_{ref}(t + \Delta t))}{\Delta t}, \tag{7}$$

where *dist* stands for the Euclidean distance.

This type of Control Expert is used to avoid a stationary obstacle, or to overtake another vehicle. In this case, as soon as the obstacle has been detected by the vehicle (*e.g.* during tracking the nominal trajectory), a value $s_{T,min}$ is calculated according to (5) and compared with the distance between the vehicle and the obstacle. The result of this computation is used by the SBM procedure to decide which behavior to apply: avoid the obstacle, slow down or stop.

3 Improving the performance of our system

3.1 *Why using ANN ?*

Previous maneuvers use a controller based on a simplified modelisation of the real vehicle. Good model matching and parameter identification are classical obstacles in this kind of approach, specially when dealing with "non academic" experimental platforms. Consider the example of the Cycab. To obtain the simplified kinematic model of the robot (Eq. 1) we assume that all the wheels role without slippage. This assumption implies that the perpendiculars to all wheels intersect at the same point which is the instantaneous center of rotation of the robot. This assumption is mechanically hard to realize and the current version of our platform just guarantees that the perpendiculars to the wheels of each axle intersect at a point which belongs to the other axle (see Fig. 3). As a consequence, the wheels do slip a bit. Moreover, the constant k of the model between front and rear steering angles is not exactly constant. In other respects, the real controls of the Cycab are $(v_1, v_2, v_3, v_4, \phi)$, the velocities of each wheel and the steering angle of the front wheels. The translation of the output (v, α) of our controller into real controls goes also through the same hypothesis of rolling without slippage. Hence, we will have to deal with errors at both planning and execution level due to the gap between our simplified model and the reality. Parameter identification (the wheel base, wheel radius, k, etc) also introduces some errors. Finally, there is the problem of the adaptivity of the control to the evolution of the parameters and the experimental context. Just consider the case of the accuracy of sensors or a simple wheel which the diameter might change because of its oldness and/or the temperature and/or the pressure of its tire. It is easy for example to notice a close relation between the repartition of load in the Cycab and the slippage of the wheels.

For all these reasons, our approach consists in using a more flexible control system, which does not compute data from an approximative analytical model anymore, but from the real vehicle itself, adapting in real-time to its minor changes, the same way as a human driver would do unconsciously. *Artificial Neural Networks* (ANN) offer such possibilities.

3.2 *Modelling the Vehicle with ANN*

We propose to use a *Radial Base Function (RBF) artificial neural network* [9] to simulate the model of the vehicle. We choose this kind of ANN mainly because of its ability to perform incremental on-line learning (by only modifying the relevant learned parameters). Such a property is needed to define an adaptive controller (i.e. a controller having the capability to tune on-line its parameters in order to compensate the current execution errors).

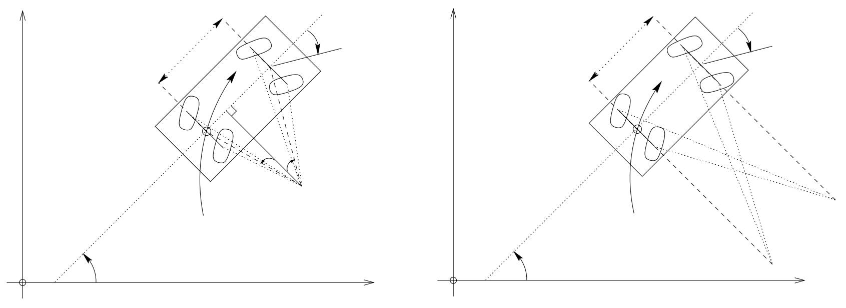


Figure 3: The ideal model of the Cycab (left) and its current mechanical realization (right).

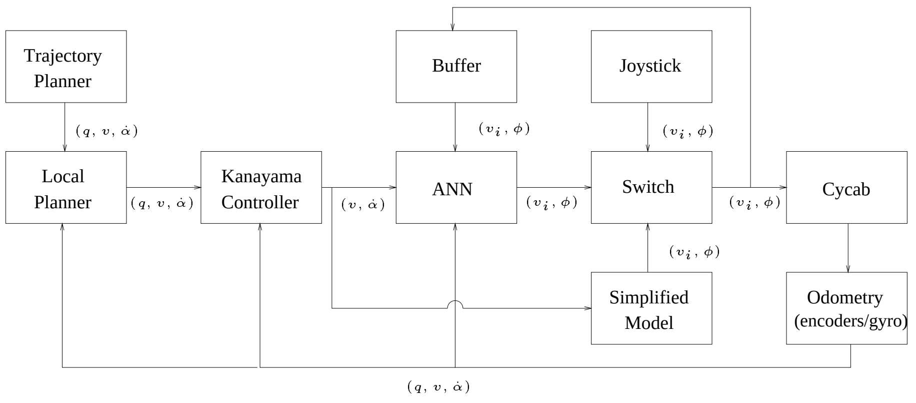


Figure 4: Trajectory following application scheme

The Control Expert of the Cycab for trajectory tracking including the ANN is presented in Fig. 4. This unique module allows the user to do the initial learning of the ANN by simply moving the robot through the joystick command[3]. When the joystick is inactive for more than 5 seconds, the robot takes into account the controls coming from the planner. The output of the global planner is filtered by a local planner which decides whether the error is too big for the Kanayama control law. In which case a local path towards a mobile point of the nominal trajectory is computed and updated at each iteration. The output $(v, \dot{\alpha})$ of our Kanayama controller are translated into Cycab controls $(v_1, v_2, v_3, v_4, \phi)$ by the ANN substituting the kinematic model of the Cycab. We use the simplified model of the Cycab as an extra security to check the validity of the ANN output before sending them to the Cycab. Indeed if the ANN outputs are "too far" from the model based estimation of the controls, the model values will have priority. The feedback of the localization module to the ANN module and the memorization of its last output are done for *in-line learning* purposes [3]. This one is processed only when the average of the error absolute value on a given time horizon is larger than a given threshold. This operation aims at compensating the discrepancy and matching again with the reality. This step guarantees a more adaptive system, even when parameters are changing significantly (see section 4).

3.3 Choosing behaviours using ANN

The previous section describes our trajectory tracking module which is a basic Control Expert also used in other Control Experts such as lane changing, emergency stop, etc. These kind of experts correspond to the *Motion Controller* level of our architecture [7].

At the top level of the architecture, the *Mission Scheduler* level has to choose between such basic behaviors of the vehicle, depending on the context captured by the sensors. For the "Following the Trajectory" SBM, we use at the MS level a particular *recurrent neural network* known as *versatile neural network* [5] to schedule different Control Experts in-line (see Fig. 5).

[3]The Cycab is steered by a joystick instead of a steering wheel.

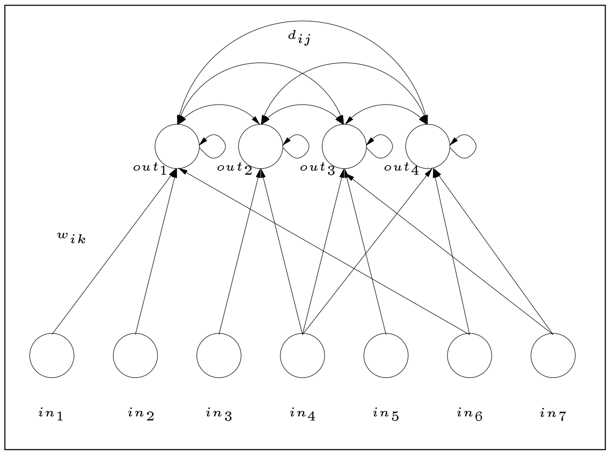


Figure 5: Versatile Neural Network

This ANN codes rules of type :

If Input_F in [range] and ... and Input_B then Decision,

where Input_F is a float input and Input_B a binary input. Good results have been already obtained (see section 4) with a network of 31 inputs (such as the position of the robot relatively to is nominal trajectory, its speed, the occupation of the neighborhood, a.s.o.) used by up to 81 rules. The system has got 25 outputs corresponding to the 25 different possible behaviors of the vehicle (e.g. normal trajectory following at different speeds, trajectory following with fixed/moving obstacle...).

We use ANN at the MS level because of its fast reactivity as a real time expert system and because of its ability to generalize its knowledge of known cases to propose a solution in unforeseen cases (for example, when an input is missing because of a failure). Indeed, if the context does not correspond to any of the rules, the versatile neural network will give the result of a competition between existing rules (see [3] for details).

4 Results

The approach described in this paper has been implemented and tested on simulation. In order to evaluate the performance of our approach, we needed to compare it to a classical system based on a simplified model of the vehicle. For this end, we needed to train the network in an off-line process in order to expect the performance of our system to be the closest possible to a classical approach (using a simplified model).

Fig. 6 shows the evolution of the ANN error having 25 hidden units, during the off-line learning phase of a trajectory tracking behaviour. We can observe that one larning step is enough to divide the error by 10, and two steps to divide it by 90.

When confronted to new examples during a test phase, the resulting performance of our system is illustrated in Fig. 7. The figure shows the errors in distance[4] obtained without and with our ANN approach, with no on-line learning. The Kanayama's law was tested with time steps of 0.1 s and then 0.5 s. In any case the error varies between 40 cm and 2 m.

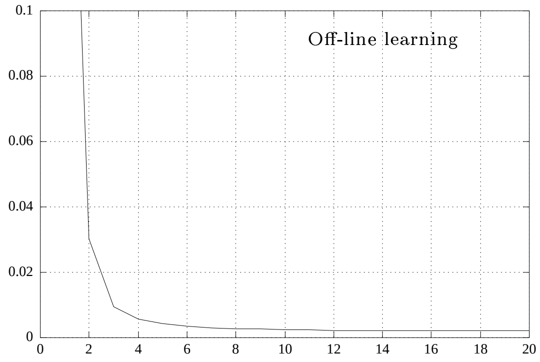


Figure 6: Error curve (in per-cent.) during the off-line learning phase, as a function of learning steps

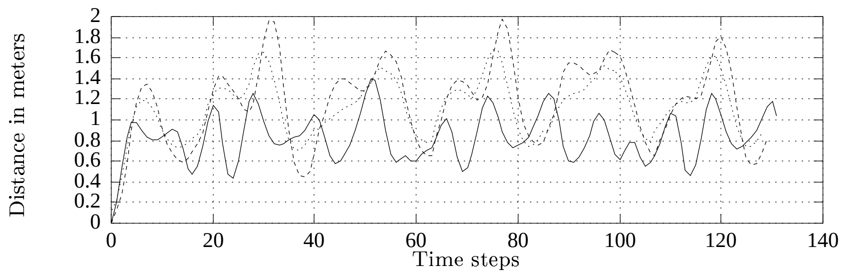


Figure 7: Distance with respect to the reference path during tracking. The continuous line corresponds to our ANN system without on-line learning. The dotted and dashed lines correspond to the performance of the system without ANN, using the Kanayama's control law with time steps of 0.1 and 0.5 seconds respectively.

[4]Distance between the reference point of the vehicle and the closest point of the path.

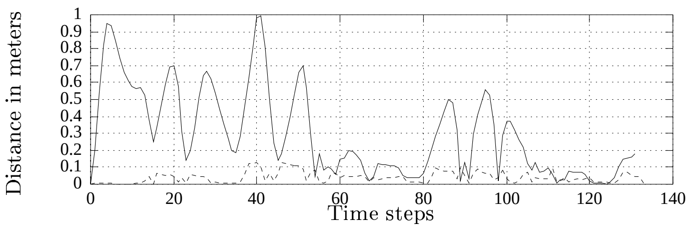


Figure 8: Distance to a nominal path during tracking. The continuous line corresponds to our improved system with on-line learning. The dashed line corresponds to a second passage over the same nominal path.

Aiming to improve this performance, we make a second passage over the nominal trajectory, this time using the on-line learning process [3]. Fig. 8 shows the evolution of the same error (distance with respect to the reference path); the continuous line corresponds to a first passage of the vehicle over the nominal path, while the dotted line shows the obtained error when a second passage is performed. One can observe that the distance between the vehicle and the tracked path decreases rapidly and becomes negligeable in the second passage over the nominal path.

In order to evaluate the performance of our system in the presence of command errors, we have simulated possible ways of malfunctionning of the vehicle. In the next example we show an error due to the steering angle ϕ. Such an error could be introduced by a bad calibration of the measuring device or by a malfunction of it. Qualitatively one can notice that such an error would be produced also by tires with different pressures, causing wheels' radi to be different. We have chosen for this test a relatively important error of 0.15 radians. The path to be tracked is a dubins path [1] and the speed command was set constant to 3 $m.s^{-1}$. Fig. 9 illustrates how the performance of our system is improved by the integration of ANN and on-line learning.

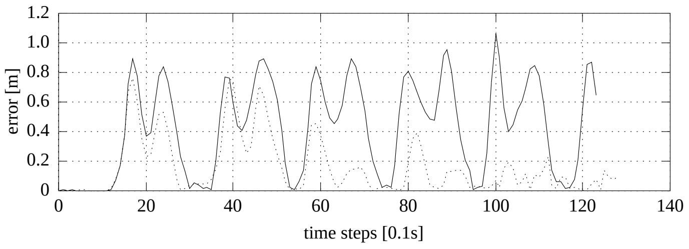


Figure 9: An error of 0.15 rad on the steering angle has been introduced after the initial training of the ANN. The plain curve represents the evolution of the position error (along the tracked trajectory) when no on-line tuning of the control parameters is available. The dotted curve represents the evolution of this error when the on-line learning function is applied.

Finally, some simulation results concerning the versatile network are presented in Fig. 10. Three cases are considered: a normal overtaking process takes place when the vehicle detects a moving obstacle and the passage lane is clear; an overtaking process is started when the obstacle is detected, but the vehicle senses another one obstructing the passage; finally, the situation is similar to the previous one, except that the vehicle is at the end of the overtaking process and a compelling cut-in behaviour has to be instantiated.

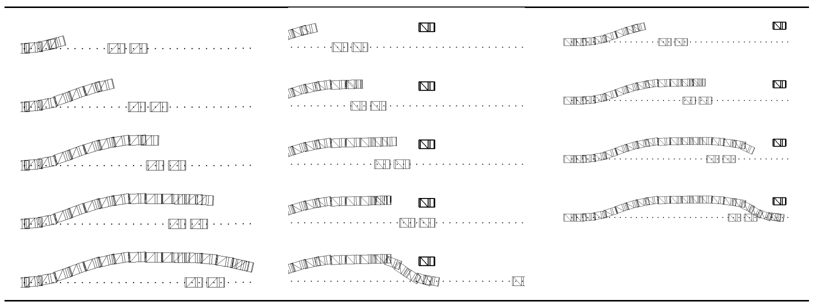


Figure 10: Different contexts of overtaking moving obstacles. With respect to the distance to the unpredicted fix obstacle on the left lane and the velocities, the MS decides to schedule a "slowing down motion" and then "going to right lane motion" (left column) or to execute the overtaking (right column).

5 Conclusion

This paper describes an approach using the Artificial Neural Networks to handle model matching problems and parameters identification problems in order to improve the performance of the system in autonomous *sensor-based maneuvers SBM*. The way in which this approach works has been illustrated by a case study consisting in following a nominal trajectory and avoiding unexpected obstacles. These two *SBM* require control and decisional schemes integrating adaptive capabilities. A *Trajectory tracking SBM* was implemented using a *Radial Base Function (RBF) ANN* in order to perform incremental on-line learning, by only modifying the relevant parameters.

As future work, motivated by the first results obtained in simulation, we plan to implement our approach on the experimental vehicle, and extend it in order to perform more complex maneuvers, such as *automated parking* or *platooning*.

Acknowledgments. This work was partially supported by the *LaRA* french program on automated roads *1999* and by the CONACYT (Consejo Nacional de Ciencia y Tecnología) in Mexico. The authors would like to aknowledge Eric Gauthier for his contributions to this paper.

References

[1] L. E. Dubins. 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, 1957.

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

[3] E. Gauthier. *Utilisation des Réseaux de Neurones pour la Commande d'un véhicule Autonome.* PhD thesis, Inst. Nat. Polytechnique de Grenoble, 1999.

[4] Y. Kanayama, Y. Kimura, F. Myazaki, and T. Noguchi. A stable tracking control method for a non-holonomic mobile robot. In *Proc. of the IEEE-RSJ Int. Workshop on Intelligent Robots and Systems*, volume 2, pages 1236–1241, Osaka (JP), 1991.

[5] A. Labbi. Neural networks for decision making in dynamic environments. In *Proc. of the Int. Conf. on Artificial Neural networks*, Amsterdam (NL), 1993.

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

[7] C. Laugier, Th. Fraichard, Ph. Garnier, I. E. Paromtchik, and A. Scheuer. Sensor-based control architecture for a car-like vehicle. *Autonomous Robots*, 6(2):165–185, April 1999.

[8] J.-P. Laumond, A. de Saint Vincent, R. Alami, R. Chatila, and V. Pérébaskine. Supervision and control of the AMR intervention robot. In *Proc. of the Int. Conf. on Advanced Robotics*, volume 2, pages 1057–1062, Pisa (IT), June 1991.

[9] J. Moddy and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–294, 1989.

[10] W. L. Nelson. Continuous curvature paths for autonomous vehicles. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, volume 3, pages 1260–1264, Scottsdale, AZ (US), May 1989.

[11] 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*, 1(4):213–229, December 1993.