IEEE Conference on Intelligent Vehicles, october 24-26 1994, Paris, France.

MODELING AND CONTROL OF MOBILE ROBOTS AND INTELLIGENT VEHICLES BY NEURAL NETWORKS

Isabelle RIVALS*, Daniel CANAS*, Léon PERSONNAZ** and Gérard DREYFUS**.

* SAGEM Eragny, Unité R&D, Avenue du Gros Chêne, 95 610 Eragny, France.
Phone : 33 1 34 30 52 07 ; Fax : 33 1 34 30 50 96 ; E-mail : rivals@sagem.fr.

** ESPCI, Laboratoire d'Électronique, 10 rue Vauquelin, 75 005 Paris, France.
Phone : 33 1 40 79 44 62 ; Fax : 33 1 40 79 44 25 ; E-mail : personna@neurones.espci.fr.

Abstract : This paper introduces the four-wheel-drive vehicle REMI, a testbed developed by SAGEM for research purposes in mobile robotics and intelligent car systems. The motion control architecture of the robot is presented, with an emphasis on the guidance and piloting modules. The latter relies on neural network techniques, and the principles underlying its design are outlined. A robust neural control scheme using an internal model of the process is developed. Experimental results are presented.

Keywords : intelligent cruise control, internal model control, neural networks, nonlinear identification and control, path following, wheeled mobile robots.

1. Introduction.

This paper describes the main features of an autonomous outdoor robot with neural network control. This robot is a fully automated standard 4WD vehicle called REMI (Robot Evaluator for Mobile Investigations) that is used as a testbed for motion control. The main components of the motion control architecture are path planning, guidance and piloting modules. Their hierarchical organization will be presented in section 2. In section 3, we will focus on the guidance module. Classical, robust, and neural piloting control techniques have been successfully experimented on REMI. We will however devote section 4 to a piloting module that is entirely designed using neural network techniques, which have achieved very good performances and offer new perspectives for intelligent vehicle control.

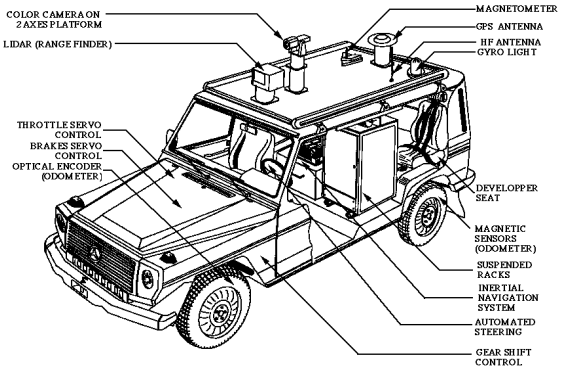

Figure 1.
REMI : the motion control testbed.

Typical automotive applications range from fully autonomous robots (open mine industry, construction, forest exploitation, agriculture [VDB93]) to automated functions on standard vehicles, such as intelligent cruise control systems, controlling the accelerator and braking systems with neural controllers.

2. Functional architecture.

The functional motion control architecture is divided in four separate modules, which are hierarchicaly organized, as shown in figure 2.

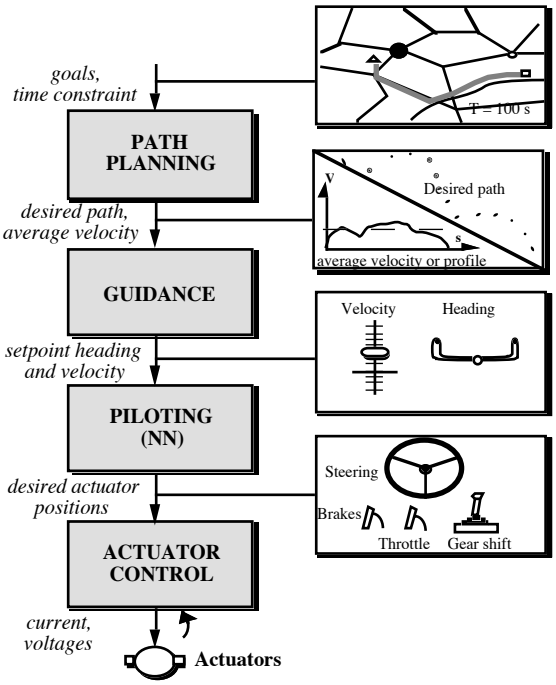

Figure 2.
Motion control architecture.

These modules achieve the following tasks : the path planning, the guidance, the neural network piloting, and the actuator control.

The path planning module.

A final goal position being given by the operator, the path planning module computes a path as a set of points satisfying this endpoint constraint. This path is associated to a time constraint specified by the operator in terms of an average velocity. A " teaching by doing " procedure can replace this module. In this case, path and velocity are " tought " by driving the vehicle manually while its position and velocity are recorded. The set of points and the average velocity are sent to the guidance module.

The guidance module.

The guidance module first generates a continuous reference trajectory and a velocity profile. Then, at each cycle of operation, this module chooses a target-point on the reference trajectory and computes a setpoint heading ψ_c from vehicle and target-point postures (i. e. position and heading). This setpoint heading and the velocity associated to the target-point, the setpoint velocity v_c, are sent to the piloting module.

The neural network piloting module.

The piloting module consists of two separate neural controllers. The heading controller computes a desired steering wheel position α. The velocity controller generates a desired position for the throttle of the engine θ and a desired pressure for the braking system π. These control intputs, or desired actuators positions, are sent to the next module at 20 Hz.

The actuator control module.

This module is interfaced with the actuators through power boards. Three digital-analog servo-loops running at 100 Hz monitor the angular position of the steering wheel, the throttle valve angular position, and the pressure in the braking system.

A localisation module computes the position, the attitude and the velocity of the vehicle, using an inertial dead-reckoning unit and an odometric sensor. Localisation and motion control modules are implemented on a 68030 board running under the real-time operating system OS9 [VDB93].

3. Guidance module.

First a continuous reference trajectory with an associated velocity profile is computed from the set of points and average velocity determined by the path planning module.

Then, at each sampling time, a setpoint heading and a setpoint velocity are generated for the piloting module. Our control strategy is based on a target-point approach. A target-point is defined for a point of the vehicle body, the control-point, which is chosen at the center of the rear axle. This location is of course weakly controllable, due to the nonholonomy constraint, but presents the following advantages :

- it coincides with the localisation point (the position computed by the dead-reckoning unit),
- steering and velocity control of this point can be decoupled,
- its turning radius is the smallest,
- the heading ψ of the vehicle at this point is tangent to the path.

The following section is devoted to the trajectory generation, section 3.2 to the choice of the target-point, and section 3.3 to the computation of the setpoint heading ψ_c.

3.1. Trajectory generator.

The aim of this function is to generate a smooth, feasible reference trajectory, interpolating the set of points sent by the path planning module. Previous work established criteria for trajectories to be suited to tracking [FRA 93]. B-splines and clothoïds both show continuity in position, heading and curvature. But B-splines are based on polynomial functions which are easely computed, integrated and differentiated, and are therefore the best candidates for real-time implementation. The trajectory generator also generates a velocity profile respecting a maximum allowable lateral acceleration and the kinematic constraints of the vehicle.

3.2. Choice of the target-point.

It is chosen at a so called " lookahead " distance D in front of the vehicle. D is a function of vehicle velocity, which must be chosen so as to guarantee that :

- D is not too small : otherwise, the vehicle might reach the target point between two computations, or oscillations might appear.
- D is not too big : if this condition is not fulfilled, the vehicle might cut corners.

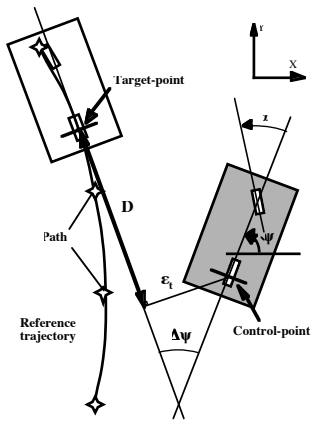

Figure 3.
Target-point and control-point postures.

Reference trajectory, lookahead distance D, control-point and target-point are shown in figure 3. $\Delta\psi$ denotes the orientation error, and ε_t the transversal error.

3.3. Setpoint heading generator.

The aim is to define a setpoint heading ψ_c for the heading controller of the piloting module. Thus ψ_c must be defined in order that if ψ converges to ψ_c, then ε_t and $\Delta\psi$ both converge to 0. Different choices of ψ_c are described in [FRA93] and [RIV93].

4. Neural network piloting module.

We address the lateral and longitudinal piloting of the vehicle, that is how to have the vehicle follow the setpoint heading and velocity determined by the guidance module. Why is a neural network approach advantageous, and how to achieve the modeling and control of the vehicle using recurrent neural networks ?

4.1. Why?

Wheeled mobile robots are nonlinear dynamical systems ; their kinematics involves geometrical nonlinearities, and their actuators introduce dynamical nonlinearities, e. g. the saturations of the steering wheel actuator and the nonlinear dynamics of the thermal engine. Thus, the identification and control of these processes require nonlinear models and controllers.

Optimal control theory has been widely used to solve such nonlinear, constrained problems. But the conventional scheme of optimal control has its own drawbacks. Very often, finding the optimal control trajectory is time consuming, and the solution obtained consists of a sequence of open-loop control vectors.

Neural networks offer interesting solutions of nonlinear control problems. Their approximation properties make them a useful tool for modeling nonlinear processes, and they provide a solution to the problem of getting a closed-form expression for optimal control laws. Besides, robustness considerations led us to develop an internal model based approach that was successfully applied to the longitudinal control of the vehicle, as will be shown in part 5. Finally, generic training algorithms were established, regardless of model and controller complexity. They will be presented in the next section.

4.2. How?

A general framework for training recurrent networks for nonlinear modeling and control is now outlined.

4.2.1. Training the neural model.

We assume that the process to be controlled can be described by the following model :

$$(1)\qquad \begin{cases} S_p(n+1) = f\left[S_p(n),\, D_p(n),\, U(n)\right] \\ y_p(n+1) = g\left[S_p(n+1)\right] + w(n+1) \end{cases}$$

where y_p is the output of the process, S_p its state, D_p are measured disturbances, and U is the control input. f and g are unknown functions. w is an output additive white noise modeling a measurement noise : it has been shown [NER94] that the optimal predictor for such a process is recursive. Thus, the model must be trained as the following feedback predictor :

$$(2)\qquad \begin{cases} S(n+1) = f_{NN}\left[S(n),\, D(n),\, U(n);\, C_m\right] \\ y(n+1) = g_{NN}\left[S(n+1);\, C_m\right] \end{cases}$$

where y is the output of the model, S its state, and f_{NN} and g_{NN} are the functions implemented by the static part of the neural net with weights C_m, which must be estimated.

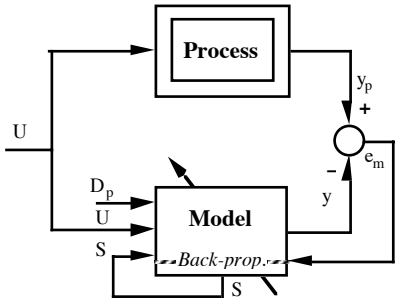

Figure 4.1.
Model training architecture.

The training of the model (see figure 4.1), i.e. the estimation of its weights C_m, is performed by minimizing the cost function J_m :

$$J_m = \frac{1}{2}\sum_{n=1}^{N}\left(e_m(n)\right)^2 = \frac{1}{2}\sum_{n=1}^{N}\left(y_p(n) - y(n)\right)^2$$

The minimization is carried out by a gradient-based technique. The gradient is computed by back-propagation on a fixed window of size N (non recursive, iterative training) using a semi-directed algorithm [NER92]. The size N of the window corresponds to the amount of training data available from the process.

4.2.2. Training the neural controller.

We perform the training of the controller using the neural model and a reference model, as shown in figure 4.2. The neural net controller with weights C_c implements the nonlinear state-feedback :

$$(3)\qquad U(n) = h_{NN}\left[U(n\text{-}1),\, S(n),\, D(n),\, R(n);\, C_c\right]$$

where D are simulated disturbances and {R(n)} is the setpoint sequence.

The weights of the neural model (2) are set to the values computed during its training.

A reference model is designed to generate the desired output sequence, or reference sequence, $\{y_r(n)\}$:

$$(4) \qquad \begin{cases} S_r(n+1) = f_r\left[S_r(n),\, R(n)\right] \\ y_r(n+1) = g_r\left[S_r(n+1)\right] \end{cases}$$

where S_r is the state of the reference model ; f_r and g_r are chosen by the designer.

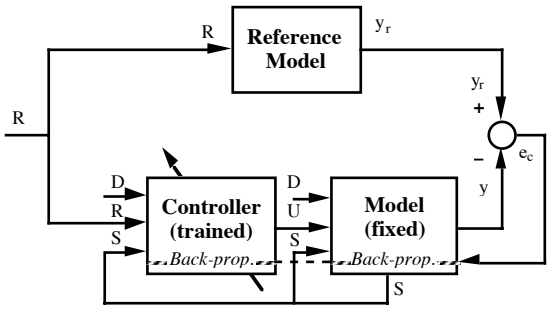

Figure 4.2.
Controller training architecture.

The weights of the controller C_c are computed so as to minimize the cost-function J_c :

$$J_c = \frac{1}{2}\sum_{n=1}^{N}\left(e_c(n)\right)^2 = \frac{1}{2}\sum_{n=1}^{N}\left(y_r(n)-y(n)\right)^2$$

Again, back-propagation on the fixed window of size N, fixed by the designer, is used to compute the gradient of J_c.

4.2.3. Using the neural controller.

a) Simple state-feedback control (SFC).

The operating control system is shown in figure 4.3. The controller, with fixed weights, computes its output from the process state, measured disturbance and setpoint :

$$(5) \qquad U(n) = h_{NN}\left[U(n\text{-}1),\, S_p(n),\, D_p(n),\, R(n)\right]$$

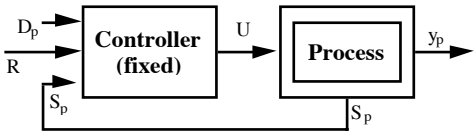

Figure 4.3.
Operating phase : SFC system.

For this control system to be stable and show good tracking and regulation properties, the neural model used for the training of the controller has to be quite accurate.

b) Internal model control (IMC).

The IMC structure incorporates a model of the process to be controlled that is simulated on-line in the control computer. This recursive model is only fed with the process inputs (controls and disturbances). Using neural networks, the model must be the feedback neural model the controller was trained with. The basic IMC structure is shown in figure 4.4. The controller computes its output as follows :

$$(6) \qquad U(n) = h_{NN}\left[U(n\text{-}1),\, S(n),\, D_p(n),\, R^*(n)\right]$$

where S is the state of the model, $R^*=R-e_{im}$, and $e_{im}=y_p-y$. Thanks to the second loop, model uncertainty and measured output disturbances are taken into account.

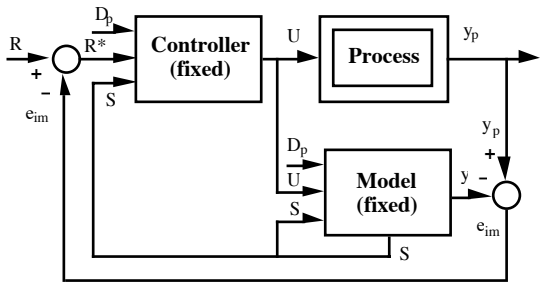

Figure 4.4.
Operating phase : IMC system.

IMC has many desirable properties, which were extensively analyzed for linear systems in [MOR89], and can be generalized to nonlinear systems [ECO86]. Thus, our neural IMC system has the two following characteristics :

(a) If the plant and the controller are input-output stable, and if the model is perfectly accurate, then the closed loop system is also input-output stable.

(b) If the controller steady-state gain is equal to the inverse of the model steady-state gain (assuming its existence), and if the closed-loop system is stable with this controller, offset-free control is obtained for constants inputs.

If the neural controller is trained using a reference model with unity steady-state gain, then the first condition of (b) is met.

One must be aware that the validity domain of the neural controller is restricted to the region of the state space it was trained in. This training region must be larger than the operating region desired for the process. As a matter of fact, during the operating phase, the controller controls the model (its input is the state of the model), which might evolve in a larger region of the state space than the process, due to the mismatch between model and process and to perturbations. In the case the model should leave the validity domain of the neural controller during the operating phase, the model state must be reset to the process state.

One can also train the controller to implement the inverse of the operator describing the plant model (if it exists) by using a reference model consisting of a simple delay for the training. In this case, a filter must be introduced between setpoint and controller in order to achieve the desirable robustness against perturbations, and the desired

tracking response [MOR89]. If the model is perfectly accurate, perfect tracking can be achieved.

5. Experimental results.

The piloting problem is split into two separate control problems (see part 2), a lateral or heading control problem, and a longitudinal or velocity control problem. The lateral problem has been presented more extensively in [RIV93].

5.1. Lateral modeling and control.

The usually adopted "bicycle model " proved to be only a rough approximation of REMI's lateral behaviour. We thus performed an identification of the nonlinear relationship between the vehicle heading ψ_p and the steering wheel control input α. Incorporating *a priori* knowledge (the basic structure of the bicycle model) into the neural model led to a good generalization in regions where training data was not sufficient (at high velocity). In addition, we could model the nonlinear first order dynamics with two saturations (in angle and velocity) of the steering wheel actuator. The neural model consists of two subnetworks NNβ and NNψ :

$$\begin{cases} \beta(n+1) = f_{NN\beta}\left[\beta(n),\, v_p(n),\, \alpha(n)\right] \\ \psi(n+1) = \psi(n) + v_p(n)\, f_{NN\psi}\left[\beta(n)\right] \end{cases}$$

where $\beta(n)$ is the model steering wheel angle. The velocity of the vehicle v_p is considered as a measured disturbance.

The heading controller was designed as a static state-feedback regulator :

$$\alpha(n) = \rho_{NN}\left[e_\psi(n),\, \beta(n),\, v(n)\right]$$

$e_\psi(n)=\psi_c(n)-\psi(n)$, $\psi_c(n)$ being the heading setpoint given by the guidance module.
The reference model used for the training of the controller (figure 4.2) was designed to compute minimal-time rallying sequences $\{\psi_r(n)\}$ rallying ψ to ψ_c for the linearized model, at various velocities (0-80 km/h) [RIV93]. Thus the neural controller was trained to be a minimal-time controller (see [RIV93] part 3.2.1 : the heading-based approach).

The model being very accurate, the heading controller could be used with very good performances in a SFC system (figure 4.3). Since the process itself is an integrator, zero steady state error could be achieved.

Experimental lateral control results (path following) are shown in figure 5.1. As described in section 3.3, the setpoint heading was computed by the guidance module so as to bring the vehicle onto the reference trajectory. The transversal error shown is the distance to the closest point of the reference trajectory, and the heading error is defined with respect to the tangent to the trajectory at this point. The velocity was monitored by a human operator, with a mean value of 4,5 m/s. The transversal error did not exceed 40 cm, with curvature values of 0.1 m^{-1} in the sharp curves.

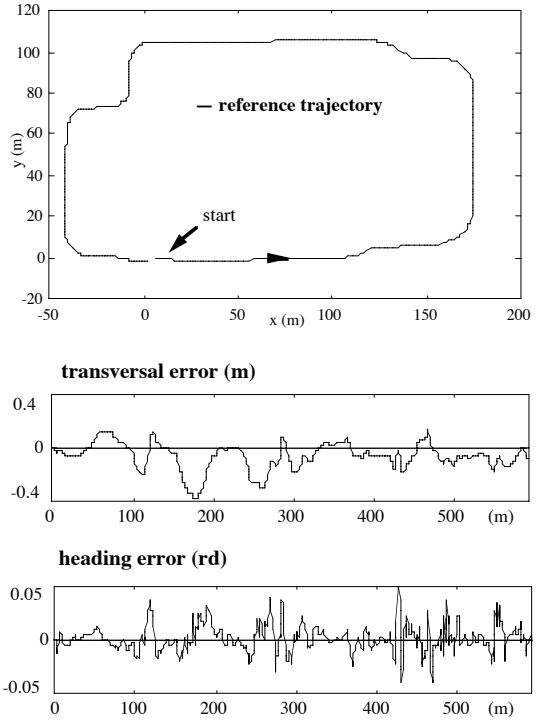

Figure 5.1.
Lateral control result (SFC system).

5.2. Longitudinal modeling and control.

The model of the longitudinal dynamical behaviour of the vehicle is a second order neural model :

$$v(n+1) = g_{NN}\left[v(n),\, v(n\text{-}1),\, \rho_p(n),\, \theta(n),\, \pi(n)\right] - G\, sin\left[s_p(n)\right]\; k\left[\rho_p(n)\right]$$

where v is the velocity, θ the throttle angular position and π the brake pressure. The (automatic) gear transmission ratio ρ_p is considered as a measured disturbance. G is the gravity constant, k a known inertia factor, and s_p the measured slope of the terrain.

The velocity controller was designed as a dynamic state-feedback controller implementing the inverse of the model (i. e. it was trained with a reference model consisting of a unit delay, within the training architecture shown in figure 4.2) :

$$U(n)=\kappa_{NN}\left[U(n\text{-}1),v_c(n),v_c(n\text{-}1),\rho(n),s(n),v(n),v(n\text{-}1)\right]$$

where $\{v_c(n)\}$ is the velocity setpoint sequence. The following switching logic was used to control throttle and brakes alternatively :
if $u(n)>0$ then $\theta(n) = u(n)$ (throttle), and $\pi(n) = 0$.
if $u(n)<0$ then $\pi(n) = u(n)$ (brakes), and $\theta(n) = 0$.

The longitudinal model being less accurate than the lateral one, we chose to use the controller trained with this model as a part of an IMC system (figure 4.4). We chose a damped second order low-pass linear filter with faster dynamics than the vehicle :

$v_r(n+1) = a_1\, v_r(n) + a_2\, v_r(n\text{-}1) + b_1\, v_c(n) + b_2\, v_c(n\text{-}1)$

where {vr(n)} is the reference sequence.

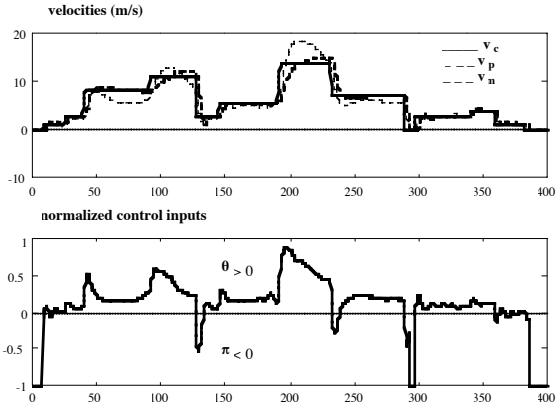

Figure 5.2.
Longitudinal control results (IMC system).

Experimental results are shown in figure 5.2. v_m, the thin dotted line, denotes the internal model velocity. These graphs show (i) the mismatch between model and process (i. e. between v_m and v_p, the thick dotted line), and (ii) that, despite this, zero steady-state error and good tracking dynamics are achieved.

6. Conclusion.

On the example of the piloting control system of a 4WD vehicle, we show the applicability of the neural network techniques to a complex control problem in automotive industry. The design of the control system was achieved using dynamical neural network models of the vehicle, and neural controllers implementing an optimal control law (lateral control) and a control law related to the inverse of the neural model in an internal model control scheme (longitudinal control). One of the salient features of our methodology is the use of *a priori* knowledge in the design of the model and controller networks [PLO94].

This piloting system was integrated in a fully automated robot ; systematic tests in rough terrain as well as comparisons to other approaches are in progress.

Acknowledgments.

This work was financed by SAGEM in the frame of a PhD study at ESPCI. We wish to thank Mrs Catherine Fargeon from DRET for her financial support to the MINERVE program, which was the starting base for the neural network project.

References.

[ECO86] Economou C. G., Morari M. & Palsson B. O. (1986) " Internal model control. 5. Extension to nonlinear systems ", Ind. Eng. Chem. Process Des. Dev., vol.25, pp. 403-411.

[FRA93] Frappier G. (1993) " MINERVE : navigation - guidage - pilotage de véhicules autonomes ", Journée thématique DRET : " Vers une plus grande autonomie des robots mobiles ", janvier 1993, Paris.

[MOR89] Morari M. & Zafiriou E. (1989) Robust process control, Prentice-Hall International Editions.

[NER92] Nerrand O., Roussel-Ragot P., Personnaz L., Dreyfus G. & Marcos S. (1992) " Neural networks and nonlinear adaptive filtering : unifying concepts and new algorithms ", Neural Computation Vol.5, pp. 165-199.

[NER94] Nerrand O., Roussel-Ragot P., Personnaz L. & Dreyfus G. (1994) " Training recurrent neural networks : why and how ? An illustration in process modeling ", IEEE Trans. on Neural Networks Vol.5, pp. 178-184.

The same methodology has been used in :

[PLO94] Ploix J.-L., Dreyfus G., Corriou J.-P., Pascal D. (1994) " From knowledge-based models to recurrent networks : an application to an industrial distillation process ", submitted to " Neural Information Processing Systems 94 ".

[RIV93] Rivals I., Personnaz L., Dreyfus G. & Canas D. (1993) " Real-time control of an autonomous vehicle : a neural network approach to the path following problem ", 5th International Conference on Neural Networks and their Applications (NeuroNîmes'93), pp. 219-229 .

[VDB93] Van den Bogaert T., Lemoine P., Vacherand F. & Do S. (1993) " Obstacle avoidance in PANORAMA ESPRIT II project ", 1st IFAC International Workshop on Intelligent Autonomous Vehicles, pp. 48-53.