Sixth International Conference on Neural Networks and Their Applications (Nîmes, 1993)

REAL-TIME CONTROL OF AN AUTONOMOUS VEHICLE : A NEURAL NETWORK APPROACH TO THE PATH FOLLOWING PROBLEM

Isabelle Rivals*, Léon Personnaz**, Gérard Dreyfus** and Daniel Canas*.

* SAGEM Eragny, Unité R&D, Avenue du Gros Chêne, 95 610 Eragny, France.
Phone : 33 1 34 30 51 19 ; Fax : 33 1 34 30 50 96.

** ESPCI, Laboratoire d'Électronique, 10 rue Vauquelin, 75 005 Paris, France.
Phone : 33 1 40 79 45 45 ; Fax : 33 1 40 79 44 25.

Abstract : A neural-network based approach to the control of non-linear dynamical systems such as wheeled mobile robots is presented. A general framework for the training of neural controllers is outlined, and applied to the lateral control of a vehicle for the path following and trajectory servoing problems. Simulation as well as experimental results on a four-wheel drive vehicle equipped with actuators and sensors are shown.

Key-words : autonomously guided vehicles (AGVs), mobile robots, model reference control, non-linear control, optimal control, path following, recurrent neural networks, trajectory servoing.

Résumé : Nous décrivons une approche neuronale de la commande de processus dynamiques non-linéaires tels que les robots mobiles à roues. Nous esquissons les grands traits d'un cadre général pour l'apprentissage de correcteurs neuronaux, et appliquons nos méthodes à la commande de la direction d'un véhicule pour son asservissement sur trajectoire. Nous illustrons notre propos à l'aide de simulations, et présentons les résultats expérimentaux obtenus sur un véhicule tout-terrain équipé des capteurs de navigation et des actionneurs nécessaires au pilotage.

Mots-Clés : asservissement sur trajectoire, commande avec modèle de référence, commande non-linéaire, commande optimale, robots mobiles, réseaux neuronaux bouclés, suivi de trajectoire, véhicules autonomes.

1. INTRODUCTION

We address the lateral control of an autonomous vehicle along a predefined trajectory using neural networks. Experimental results on a full-scale outdoor robot, a standard four-wheel drive car equipped with the sensors and actuators needed for navigation and control, demonstrate that neural techniques can be applied to real-world problems in robotics.

Classical control theory provides many design techniques to achieve performances specified in terms of rise and settling-time, gain and phase margin, bandwidth… These methodologies are well suited to the design of linear controllers for linear systems, with guaranteed stability and robustness. They are, however, of restricted use for control problems involving non-linear dynamic processes with inequality constraints on state and control variables, such as wheeled mobile robots with actuator limitations.

Optimal control theory has been widely used to solve such non-linear, constrained problems. But the conventional scheme of optimal control has its own drawbacks. Finding the control trajectory that minimizes the performance measure often requires the solution of non-linear differential equations. This can be achieved by using iterative numerical methods (quasi-linearization, steepest descent…) which are time consuming, or by dynamic programming ; both miss closed-form expressions for the feedback control laws.

Neural networks offer an alternative to the usual formulations and solutions of constrained optimization problems. Their approximation capabilities make them of possible use as models of the process to be controlled, as well as suitable controllers parameterizing non-linear optimal feedback control laws. In addition, the performance measure which, when minimized, corresponds to the optimal behaviour, can be defined with respect to a reference model. Finally, generic algorithms using a gradient-based approach to achieve the minimization of the performance measure - regardless of model and controller complexity - have been established.

In the second part of this paper, we present a general framework for the training of neural networks for control purposes. Part 3 is devoted to our application : the lateral control of a vehicle for the path following problem. We first present our test-vehicle and its neural model. We subsequently apply the control scheme developed in part 2 using two different approaches of the path following problem. Simulation and experimental results are shown in both cases. They are discussed in part 4.

2. TRAINING OF RECURRENT NEURAL NETWORKS FOR NON-LINEAR CONTROL

We assume that the process to be controlled is described by the following discrete-time model :

$$\begin{cases} S(k+1) = f(S(k), U(k)) \\ Y(k) = g(S(k)) \end{cases}$$

where $S(k)$, $Y(k)$ and $U(k)$ are the state, output and control vectors at time k respectively, and f and g are unknown non-linear functions.

The control system consists of the following components :

- a *neural network predictor model* with state S_m and output Y_m, is first trained :

$$\begin{cases} S_m(k+1) = f_{NN}(X_m(k), U(k)) \\ Y_m(k) = g_{NN}(S_m(k)) \end{cases}$$

where f_{NN} and g_{NN} are non-linear functions implemented by neural networks, and where the state input X_m may take different values depending on the particular predictor choice. For a recursive predictor, $X_m(k) = S_m(k)$; for a non-recursive predictor, $X_m(k)$ is often taken to be equal to the process state $X_p(k)$. The rationale of this choice has been discussed in [NER92a].

- a *reference model* (possibly a simple delay) is designed, which generates the desired output sequence $\{Y_r(k)\}$ from the setpoint sequence $\{R(k)\}$:

$$\begin{cases} S_r(k+1) = f_r(S_r(k),R(k)) \\ Y_r(k) = g_r(S_r(k)) \end{cases}$$

where $S_r(k)$ is the state of the reference model at time k, $Y_r(k)$ its output, $R(k)$ the setpoint vector, and f_r and g_r are known (possibly linear) functions.

- the *neural controller*, with weight matrix C, computes the control sequence $\{U(k)\}$ from the setpoint sequence and the model state input, and can therefore implement any suitable non-linear state-feedback control law $U(k) = \psi(X_m(k), R(k))$.

The aim of the training is to compute the weights of the neural controller, either adaptatively or non adaptively, so that the output of the process becomes as close as possible to the output of the reference model. We restrict the scope of our presentation to a non-adaptive context in which the process itself is not taken into account during the training of the controller (for a general presentation, including adaptive control schemes, see [NER 93a]). This approach of neural control is well suited to the off-line validation of controller structures, provided the neural model of the process is accurate enough, and is often encountered in the literature (e. g. [NAR 91]).

2.1 Training phase

The training algorithms aim at minimizing a cost-function J by a gradient-based technique. J involves the squared difference E_m between the output of the model and the output of the reference model over a time horizon of length N_c :

$$J = \frac{1}{2} \sum_{k=N_t-N_c+1}^{N_t} \|E_m(k)\|^2 = \frac{1}{2} \sum_{k=N_t-N_c+1}^{N_t} [Y_r(k) - Y_m(k)]^T W [Y_r(k) - Y_m(k)]$$

where W is a weighting matrix, $N_c \leq N_t$ (for example $N_c=1$ is chosen if a desired value exists for the final output only) and N_t is the number of time steps used for the evaluation of the gradient of the cost-function ([NER92b][NER93b]).
The weights will be modified iteratively in the direction opposite to that of the gradient :

$$\Delta Cij = - \mu \frac{\partial J}{\partial Cij} = - \mu \frac{\partial}{\partial Cij} \left(\frac{1}{2} \sum_{k=N_t-N_c+1}^{N_t} \|E_m(k)\|^2 \right)$$

where μ is the gradient step.

The value of the cost-function J, hence of its gradient, depends on the NN predictor model used for the computation of Y_m. The NN predictor model is either recursive or non-recursive, each case leading to a specific training algorithm.

a) The UD control algorithm (" UnDirected " algorithm, see figure 1)

If the NN predictor model is *recursive*, the state input $X_m(k)$ takes the values :

$$X_m(0) \text{ arbitrary, and } X_m(k) = S_m(k) \ \forall \ k>0.$$

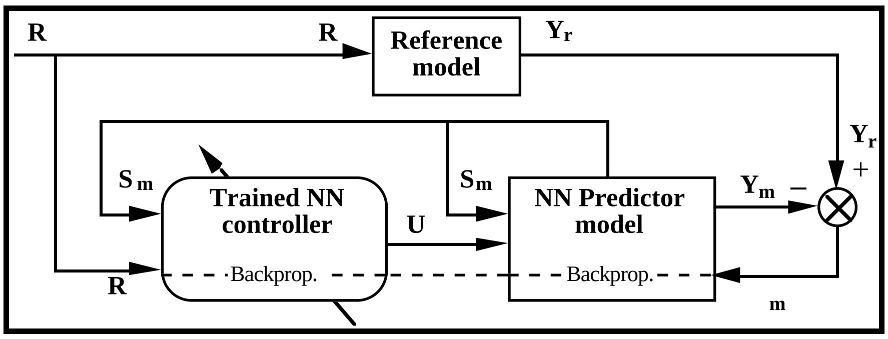


FIGURE 1
Controller training architecture associated to an UD control algorithm.

At $k=0$ only, the state inputs of the model are initialized to arbitrary values. The controller and the predictor build up a *recurrent network*, and the computation of the gradient - with respect to the controller weights - is achieved using *dynamic back-propagation* over the time-horizon N_t ([NER92b], [NAR90], [NAR91]).

b) The D control algorithm (" Directed " algorithm, see figure 2)

If the NN predictor model is *non-recursive*, the state input $X_m(k)$ takes the values :

$$X_m(k) = S_r(k) \ \forall \ k.$$

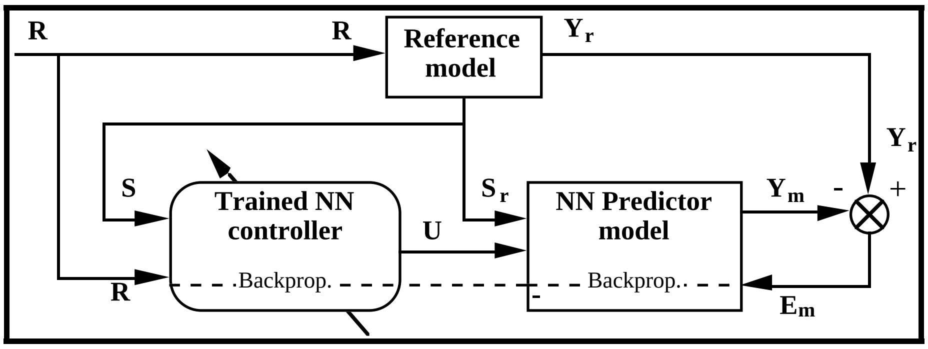


FIGURE 2
Controller training architecture associated to a D control algorithm.

Thus, at every time step, the state inputs of the model are taken equal to the values of the state variables of the reference model. The controller and the predictor now build up a *feedforward network*, and the computation of the gradient can be achieved using *static back-propagation*. For this algorithm to be applicable, it is necessary that the reference model specifies the whole state (not only the desired output).

2.2 Operating phase

During the operating phase, which follows the training phase, the weights are frozen and the controller is used as shown in figure 3. The reference model being *implicit* (that is, its output is not fed to the neural controller during training [LAN79]), it is not part of the operating control system itself.

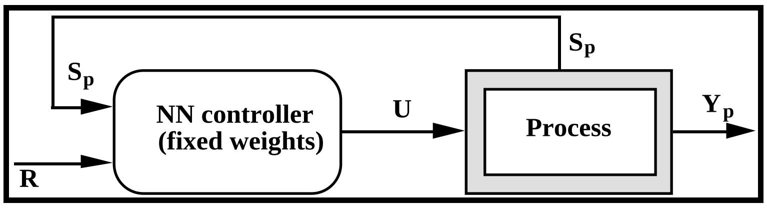


FIGURE 3
Control system during the operating phase.

3. APPLICATION TO THE LATERAL CONTROL OF AN AUTONOMOUS VEHICLE

3.1 The vehicle REMI and its model

Our testbed is the SAGEM autonomous navigation test vehicle REMI (Robot Evaluator for Mobile Investigations), a four-wheel drive vehicle equipped with actuators and sensors. Our aim being here the lateral control, we are concerned with its steering system only. The steering actuator monitors the angular position of the steering wheel, and the velocity is controlled by a human operator using the brake and throttle pedals. Thanks to an inertial dead-reckoning unit and an odometric sensor, a navigation module computes position, orientation and velocity of the vehicle. The navigation and piloting modules are implemented on a 68030 board running under the operating system OS9. The interfaces with the hardware are achieved with specific boards. More details about REMI can be found in [VDB93][FRA93]. As pointed out in part 2, an accurate model of the vehicle is required for the training, since the controller is non-adaptive. Thus, both the kinematics of the vehicle and its dynamics were identified, using two neural networks.

For the kinematic part, the classically adopted " bicycle " model [JUR93][SIN90] proving to be unsatisfactory, we performed a (non-adaptive) identification of the non-linear relationship (3) - given below - between the heading ψ of the vehicle and the position β of the steering wheel (Figure 4). We used a model based on the bicycle model, including a neural network which gives a better estimate of the geometrical non-linearity (ϕ_{NN}) than the *tan(.)* relationship of the bicycle model ($\psi(k+1) = \psi(k) + v(k)\,(\Delta T/L)\tan(\beta(k))$).

The vehicle dynamics can be separated into *actuator dynamics* and v*ehicle-ground interactions* . The latter are due to the flexing and slipping in the contact between the vehicle and the ground, and are negligible at low speed (<4 *m/s*). But we assume that, for a given trajectory, the upper limit of speed for which the vehicle will not slip will not be reached. Moreover, we consider that the ground is flat (nevertheless, we achieved good performances on uneven ground). Thus, the dynamics reduces to the steering actuator dynamics. We identified the relationship between steering command α and steering wheel position β with a second neural net consisting of two neurons with linear saturated activation function, neglecting other dynamical effects than a dead-time and the angle and velocity saturations of the actuator. Measurement noise being dominant, both identifications were performed with recursive predictors [NER93b].

The overall model, with state $S(k) = [x(k), y(k), \psi(k), \beta(k), v(k)]$ is described by the following state equations :

(1) $x(k+1) = x(k) + v(k)\,\Delta T \cos \psi(k)$

(2) $y(k+1) = y(k) + v(k)\,\Delta T \sin \psi(k)$

(3) $\psi(k+1) = \psi(k) + \dfrac{v(k)\,\Delta T}{L}\,\phi_{NN}(\beta(k))$

(4) $\beta(k+1) = sat_{\beta max}\,(\,\beta(k) + sat_{\Delta\beta max}\,(\,\alpha(k-N) - \beta(k)\,)\,)$

(5) $v(k+1) = v(k) + f_u(S(k), \Theta(k), B(k))$

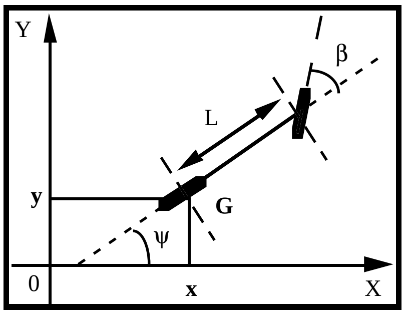


FIGURE 4
The variables of the model.

where :
- x, y are the coordinates of the guide-point G located at the center of the rear axle (for the justification of this choice, see [FRA93][SIN90]) in (X,Y) ;
- ψ is the heading of the vehicle ;
- β is the steering wheel angle ;
- α is the steering command ;
- v is the vehicle velocity measured at the guide-point G ;
- $L=2.85\ m$ is the distance between axles ;
- $sat_X(.)$ is the saturation function between $-X$ and $+X$; β_{max} and $\Delta\beta_{max}$ are the values of the angle and velocity saturations, identified at respectively $0.5 rd$ and $0.2\ rd/s$ by the neural network representing the dynamics of the vehicle ;
- $\Delta T=0.04\ s$ is the sampling period ;
- N is the actuator dead-time, experimentally identified as being ~ 4 ($4\ \Delta T = 160\ ms$) ;
- ϕ_{NN} is the non-linear function implemented by the neural network representing t
he kinematics of the vehicle.
- the state variable v is governed by a relationship involving the control inputs Θ (throttle) and B (brakes) characterized by the unknown function f_u (its knowledge is not necessary for the lateral control of the vehicle).

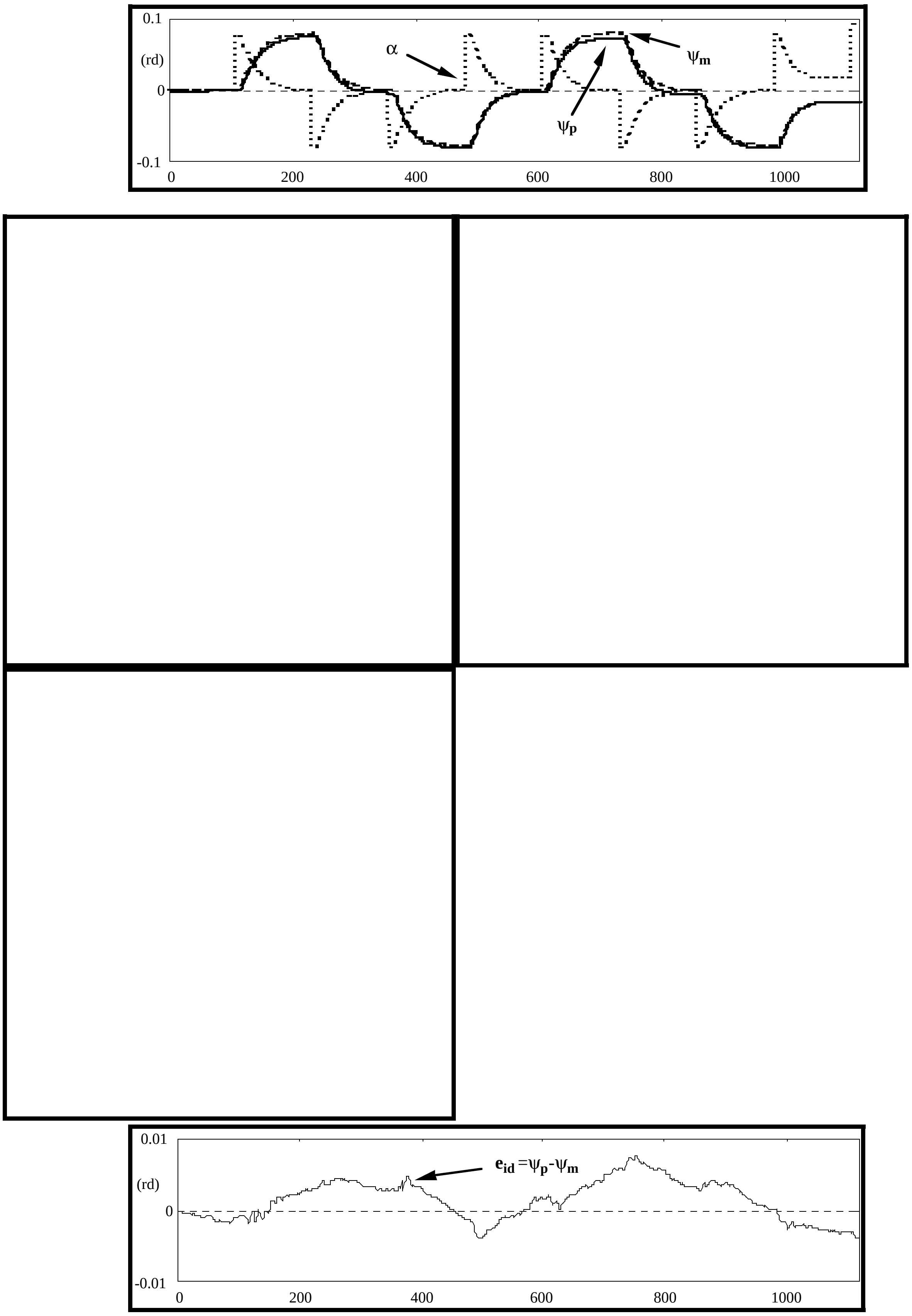


FIGURE 5

Comparison between actual vehicle heading ψ_p and recursive neural predictor output ψ_m over 1120 time steps (*45 s*).

Figure 5 shows a long-term prediction of the heading using the overall model, which turns out to be fairly accurate. The performance obtained justifies the simplifying assumptions made about the dynamics of the vehicle, at least under these operating conditions (even and dry ground), and allows us to use the model as a simulation model for control purposes.

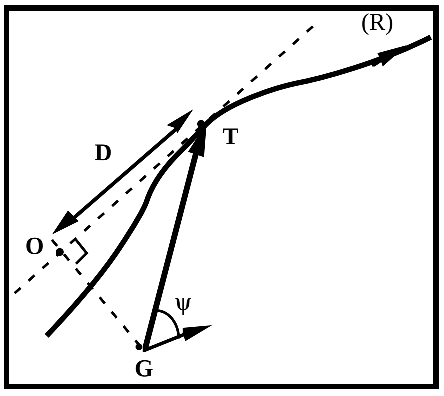
3.2 The path following problem

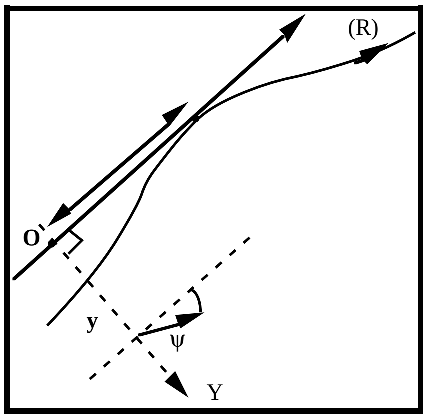
The problem is to steer the vehicle along a known reference path or trajectory *(R)* defined in the plane. We express the path following problem in terms of controlling the vehicle orientation only, given its advancement velocity (see [SAM92]). Our aim is to derive a feedback control law for the steering command so as to (i) have the vehicle rally and follow the path and (ii) have the orientation of the vehicle tangent to the path when the vehicle follows the path. The first objective corresponds to a positional requirement, the second one to an orientational requirement.

We present two different approaches to solve this problem. In both cases, we choose to define a moving target-point *T* on *(R)* at a distance *D* function of vehicle speed, much as a human driver would look out further along the path, the higher the vehicle speed :

a) In the first approach, we specify only an orientational requirement merging the two objectives (orientation and position). During the training phase, a neural controller learns to bring the heading ψ of the vehicle back to zero ; during the operating phase (figure 6a), the origin ψ=0 is defined by vector $\overrightarrow{GT}$ so that ψ represents the orientation error. This approach, termed " heading-based " approach, is described in part 3.2.1.

b) In the second approach, requirements in both position and orientation are specified. During the training phase, a neural controller learns to drive the vehicle to the posture $\xi = [y,\psi] = [0,0]$; during the operating phase (figure 6b), the line *y=0* is defined as the tangent to *(R)* at the target position, the orientation $\psi=0$ being given by the positive *X* axis. As in a), ψ represents the orientation error, and y the transversal error. Part 3.2.2 is devoted to this approach, termed " posture-based " approach.





a) Heading-based approach. **b) Posture-based approach.**

FIGURE 6

Target-point and relative coordinates definition during the operating phase.

(T = target-point ; O = orthogonal projection on the tangent to (R) through P ; G = guide-point of the vehicle)

3.2.1 The heading-based approach

a) Training phase and simulation results

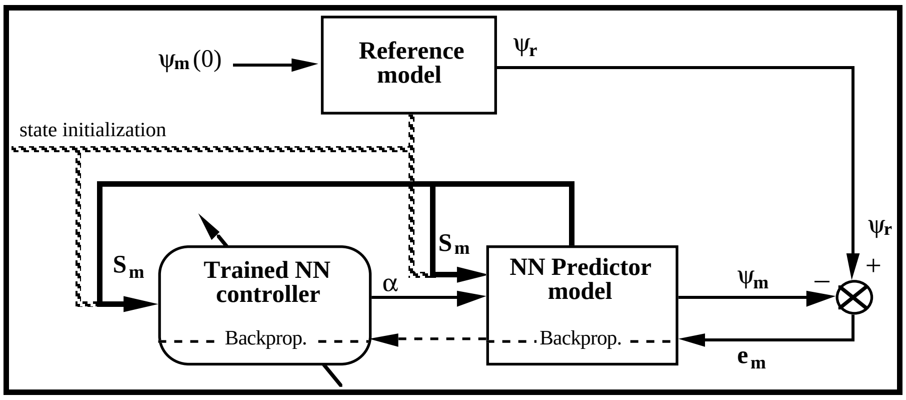
The neural controller task is to make the model output ψ_m as close as possible to the output of a reference model ψ_r which defines the desired way of bringing the heading to zero. Feedback is thus required only from the partial state :

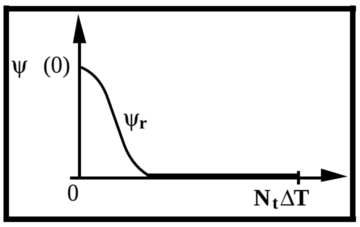
$$S_m(k) = [\psi_m(k), \beta_m(k), v_m(k)]$$

The cost-function to minimize over a trajectory of time-length N_t involves only the heading ψ_m :

$$J = \frac{1}{2} \sum_{k=N_t-N_c+1}^{N_t} (\psi_r(k) - \psi_m(k))^2 = \frac{1}{2} \sum_{k=N_t-N_c+1}^{N_t} e_m^2(k)$$

The reference output ψ_r is defined as the output of the linearized model ($\psi_r(k+1) = \psi_r(k) + v(k)\ (\Delta T/L)\ \alpha_r(k-N)$) controlled by a minimum-time controller with inequality constraints on α_r and $\Delta\alpha_r$ (corresponding to the angle and velocity saturations β_{max} and $\Delta\beta_{max}$ of the actuator). The parameters of the reference controller are computed for various initial conditions, with constant speed on each trajectory, in which case the computation is very simple. Since the reference model specifies only the desired output (and not the whole state), the training has to be performed with an UD algorithm. We choose N_c=N_t, N_t being sufficiently long for the reference model to reach ψ=0 (figure 7b). The initial conditions for the vehicle model and reference model are taken in the range $\psi_r(0)=\psi_m(0)[\,[-\pi/2;\ +\pi/2]$ *rd* and $v[[0;10]$ *m/s.*



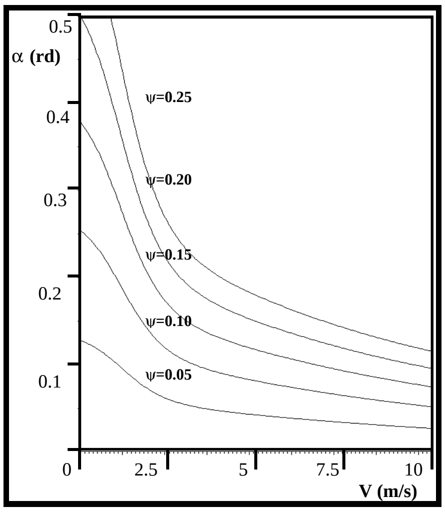


a) Controller training architecture using an UD algorithm. **b) Reference heading.**

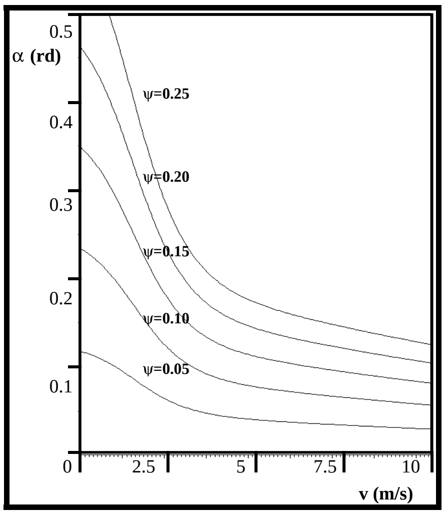
FIGURE 7

Heading-based approach : training of the steering controller with minimum-time control reference model.

To avoid the drawbacks of a bang-bang type controller, we used reference models with a lower constraint on $\Delta\alpha_r$ than the real value of $\Delta\beta_{max}$ ($\Delta\beta_{max}$ = *0.20 rd/s*). The best trade-off beetwen performance and driving comfort was determined experimentally on REMI, leading to the choice of $\Delta\alpha_{max}$ = *0.175 rd/s*. Figure 8 shows the output α of the neural controller as a function of v and ψ , for these two values of $\Delta\alpha_{max}$.



a) $\Delta\alpha_{max}$ ***= 0.20 rd/s (real*** $\Delta\beta_{max}$ ***value)***



b) $\Delta\alpha_{max}$ ***= 0.175 rd/s***

FIGURE 8

Output of the neural network controller trained with two different reference models.

The neural controller is a MLP (2,3,3,1), with inputs ψ, v and output α (feeding β to the network did not improve the performance).

b) Operating phase and experimental results

The distance D to the target-point is chosen to depend linearly on vehicle velocity :

$$D = d_0 + F_v\,.\,v\,.$$

The values of the parameters d_0 and F_v were determined by computer simulations.

The controller has been used successfully on various trajectories. Figure 10a shows the experimental path following performance obtained along the trajectory of figure 9. The transversal error e_t (distance to the closest point of the reference trajectory) is kept very small (< *60 cm*, the curvature reaching *0.1* m^{-1}) with speeds up to *25 km/h*. The heading error e_ψ (defined with respect to the tangent to (R) at the closest point of the reference trajectory) is kept smaller than *0.1 rd*. Figure 10b shows the behaviour of the vehicle for the trajectory servoing problem, the vehicle being initialized far away from the reference trajectory.

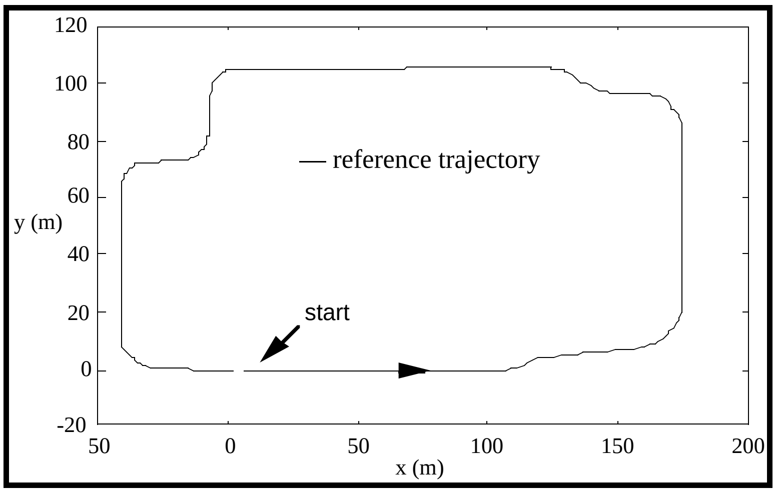
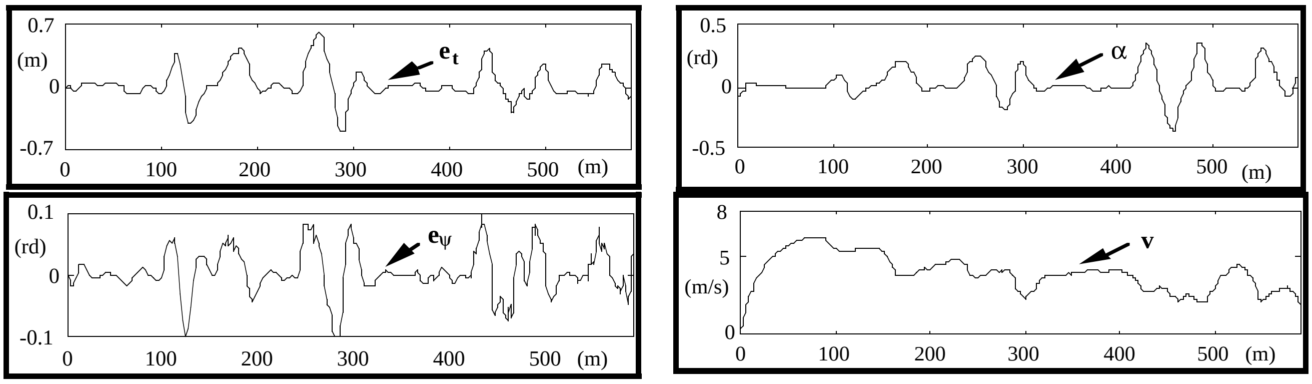
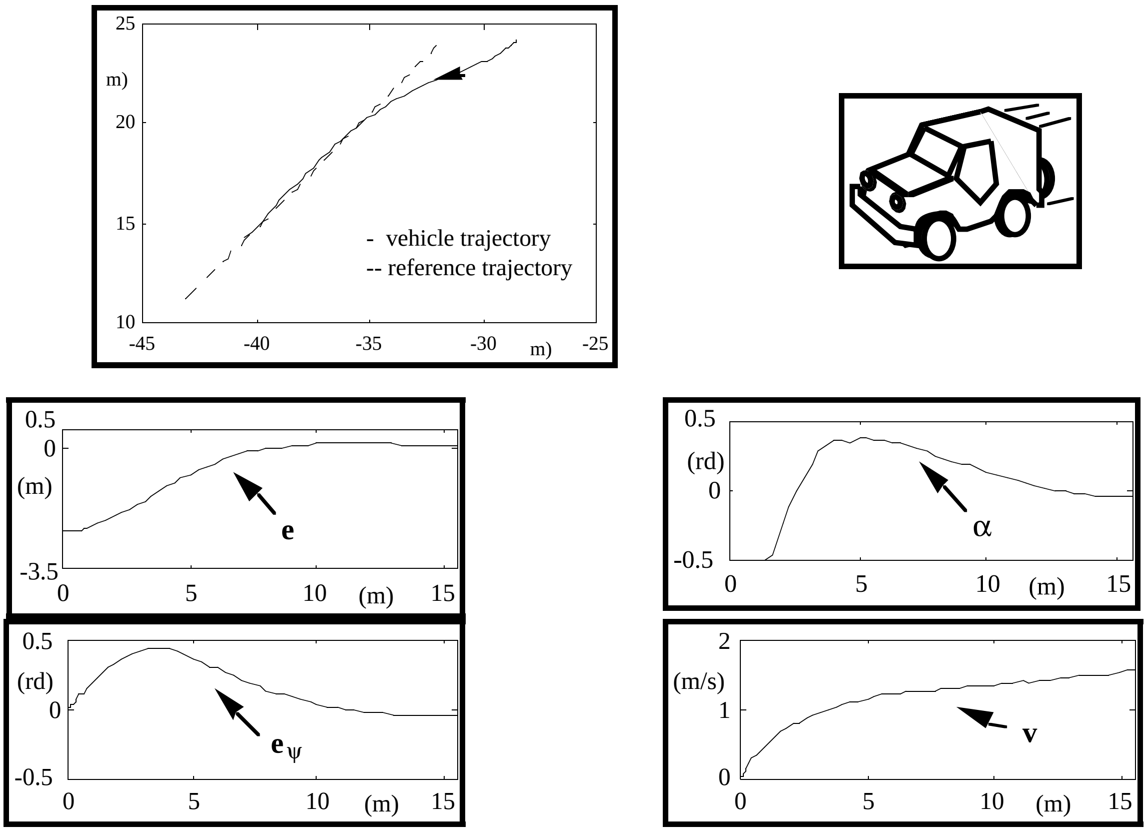


FIGURE 9
Map of the reference trajectory (around the SAGEM research center, with total length ~ 600 m) used for the path following and trajectory servoing experiments of figures 10 and 13.



a) Path following : experimental results.



b) Servoing the reference trajectory with an initial transversal error of *2,5 m* : experimental results.

FIGURE 10
Experimental results on the vehicle REMI using the heading-based approach
(e_t = transversal error, e_ψ = heading error, α = steering command (neural controller output), v = vehicle velocity).

3.2.2 The posture-based approach

a) Training phase and simulation results

The neural controller task is now to bring the model output ξ_m as close as possible to the *posture* ξ_r defined by a reference model. Feedback is now required from the state :

$$S_m(k) = [y_m(k), \psi_m(k), \beta_m(k), v_m(k)]$$

The cost-function to minimize over a trajectory of time-length N_t involves the posture, thus ψ_m and y_m :

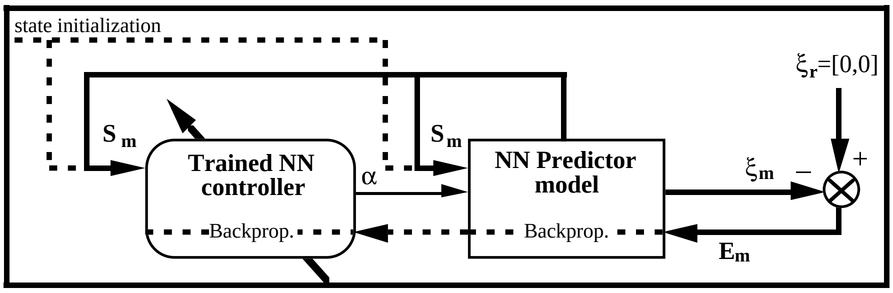
$$J = \frac{1}{2} \sum_{k=N_t-N_c+1}^{N_t} (y_r(k)-y_m(k))^2 + F_\psi\,.\,(\psi_r(k)-\psi_m(k))^2 = \frac{1}{2} \sum_{k=N_t-N_c+1}^{N_t} \text{»}E_m(k)\text{»}^2$$

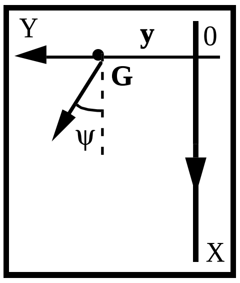
where F_ψ is a weighting factor which has to be determined.

In this case, the computation of a minimum-time controlled reference model is quite difficult ; without taking the velocity saturation of the actuator into account, it is already a non-trivial task and requires some tedious algebra (see [RIN93]). Another choice could be to fix a distance depending on vehicle speed within which the robot should be brought back on the reference trajectory, and to define the path (polynomial (e. g. [SHI90]), splines…) satisfying the boundary conditions and the constraints on curvature and curvature variation. We found it more tractable to use a simple " identity " reference model :

$$\xi_r(t) = [0, 0]$$

and to let the cost-function, hence the weighting factor F_ψ, determine entirely the desired behaviour.



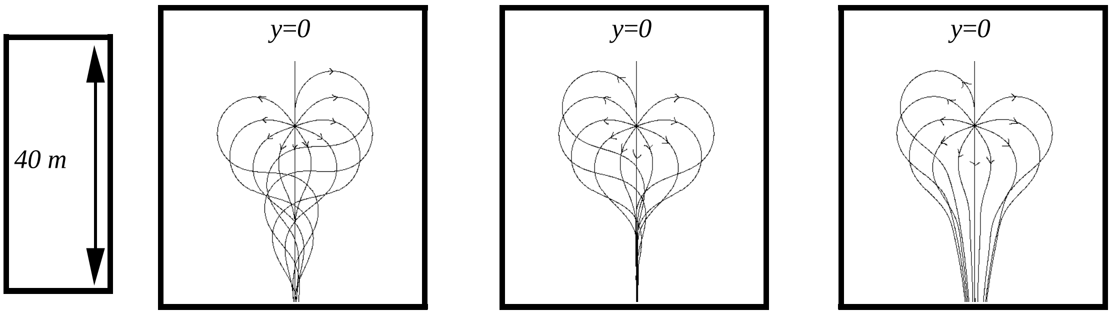


a) Controller training architecture using an UD algorithm. **b) Reference posture.**

FIGURE 11

Posture-based approach : training of the steering controller with identity reference model.

We performed the training using an UD algorithm with $N_c=N_t$, N_t being chosen sufficiently long for the model to stabilize on the *y=0* trajectory. The state of the model was randomly initialized in the range $\psi(0)[[-\pi;+\pi]$ *rd*, *y(0)[[0;10] m*, and with constant speeds *v[[0;10] m/s*. As expected, the choice of the weighting factor in the cost-function proves to be decisive. Figure 12 shows simulation results obtained with three different values for F_ψ , which led us to the choice of $F_\psi =10$.



a) F_ψ=1 **b) F_ψ=10** **c) F_ψ=50**

FIGURE 12

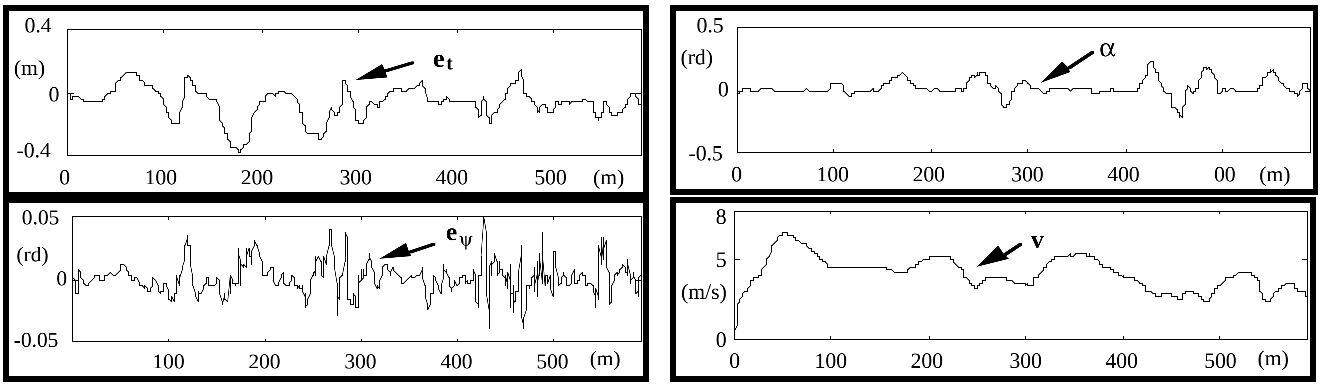
Simulation of the model behaviour depending on the weighting factor F_ψ in the cost-function used for training.

The neural controller is a MLP (3,3,3,1), with inputs *y*, ψ, *v* and output α (as in the heading-based approach, β is not used).

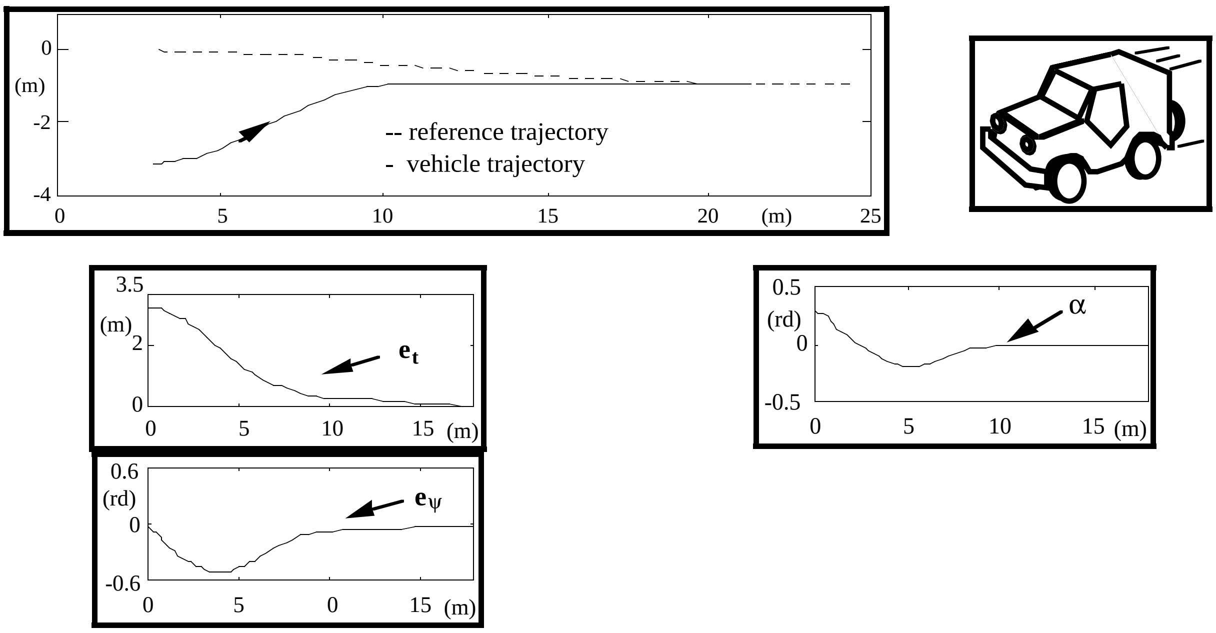
b) Operating phase and experimental results

During the operating phase, the distance *D* to the target-point was also chosen to depend linearly on vehicle speed, and the appropriate values of d_0 and F_v for this case were also determined by computer simulations.

Figure 13a shows the experimental path following performance of the controller on the same trajectory (the trajectory shown in figure 9). With an equivalent speed profile, both the transversal error (e_t<35 *cm*) and the heading error (e_ψ<0,05 *rd*) are smaller than in the heading-based approach. Figure 13b shows the behaviour for the trajectory servoing task : the trajectory is reached without overshoot.



a) Path following : experimental results.



b) Servoing the reference trajectory with an initial transversal error of 3 m : experimental results.

FIGURE 13
Experimental results on the vehicle REMI using the posture-based approach
(e_t = transversal error, e_ψ = heading error, α = steering command (neural controller output), v = vehicle velocity).

4. DISCUSSION

Both approaches, especially at high speed, lead to better results than the classical approach used before, consisting in LQ control (with fixed gains), based on the linearized model, without taking the actuator dynamics into account. The posture-based approach leads to better behaviour for the trajectory servoing task. This is an important property, which plays a role when a relocalization procedure is activated for example. The choice of the target-point is also extremely important : we compared our strategy to the commonly used geometric approach where the target-point is simply the closest point on the trajectory, and where a *feed-forward* curvature term is added to the command output by the feedback controller [RIN93][VDM93]. Our approach proved to be much more robust with respect to curvature variations, and does not necessitate the estimation of the curvature, which is often a problem (be it by image processing [JUR93], or with a trajectory defined by a series of setpoints - which is our case).

We would further like to stress the following advantages of the neural control scheme developed in part 2, of both theoretical and practical nature, as they are illustrated by our application :`

- The use of a *reference model* allows roughly two types of control policies : (i) In the case where a reference model can be defined, the right choice of the learning algorithm and of its parameters (N_c, N_t) makes it possible to assign *explicitly* its dynamics to the closed-loop system ; this was shown with the heading-based approach, where an optimal controller was learned by the neural network ; not only do we obtain optimal solutions in the sense of a chosen criterion (and to a given accuracy) but, in addition to their optimality, these solutions take a *closed-form expression*, as pointed out in [RIN93]. (ii) In the case where it is difficult to formulate the desired dynamics

explicitly, the optimal behaviour is defined *implicitly* by the cost-function, much as in optimal control ; this was illustrated with the posture-based approach.
- Any *non-linear model of the process* can be used, be it a neural network, or a physical model, provided its jacobian can be evaluated and used for the computation of the gradient. Known non-linearities such as saturations introduced by the actuators can be simply incorporated in the model using saturation functions instead of standard sigmoids. The algorithms can then be applied regardless of the complexity of the model (and of the controller).
- The neural controllers are of small size (at most ten neurons), making their implementation easy. There is *no need for special purpose hardware* to operate in real time, even at high frequency control rates (up to *12.5 Hz* for the heading-based approach, *5 Hz* for the posture-based). Neural networks are well suited to real time operation, in contrast to traditional, iterative methods of solving optimal control problems.

5. CONCLUSION

The neural control scheme outlined in this paper applies successfully to non-linear systems such as wheeled mobile robots with actuator limitations. As a demonstration, the design of a neural controller for the trajectory following and servoing problem using two different approaches has been presented, already showing the promising performance and flexibility of the neural control framework. Future studies will deal with the extension of this framework to more difficult operating conditions (higher speed, rough terrain) requiring adaptive identification and control schemes. This work is also being currently extended to the velocity control using brakes, throttle and gear.

Acknowledgements

The authors are grateful to Philippe Lemoine and Stéphane Sallé (SAGEM) for numerous discussions and their help with the experiments, and wish to thank Michel de Cremiers (SAGEM) for initiating and efficiently supporting the project.

6. REFERENCES

[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.
[JUR93] Jurie F., Rives P., Gallice J. & Brame J. L. (1993) "High speed vehicle guidance based on vision", 1st IFAC International Workshop on Intelligent Autonomous Vehicles, pp. 205-210.
[LAN79] Landau Y. D. (1979) *Adaptive Control : the Model Reference Approch*, Marcel Dekker.
[NAR90] Narendra K. S. & Parthasarathy K. (1990) "Identification and control of dynamical systems using neural networks", IEEE Trans. on Neural Networks vol.1 No.1, pp. 4-27.
[NAR91] Narendra K. S. & Parthasarathy K. (1991) " Gradient methods for the optimization of dynamical systems containing neural networks ", IEEE Trans. on Neural Networks **2**, 252-262.
[NER92a] Nerrand O. (1992) "Réseaux de neurones pour le filtrage adaptatif, l'identification et la commande de processus", Thèse de doctorat de l'Université Paris VI.
[NER92b] Nerrand O., Roussel-Ragot P., Personnaz L., Dreyfus G. & Marcos S. (1992) " Neural networks and non-linear adaptive filtering : unifying concepts and new algorithms ", Neural Computation **5**, 165-199.
[NER93a] Nerrand O., Personnaz L. & Dreyfus G. (1993) " Non-linear recursive identification and control by neural networks : a general framework ", European Control Conference 1993.
[NER93b] Nerrand O., Roussel-Ragot P., Personnaz L. & Dreyfus G. (1993) " Training recurrent neural networks : why and how ? An illustration in process modeling ", IEEE Trans. on Neural Networks, in press.
[RIN93] Rintanen K. T. (1993) " Curvature-optimal path planning and servoing for autonomous vehicles : a neural net implementation ", 1st IFAC International Workshop on Intelligent Autonomous Vehicles, pp. 475-480.
[SAM92] SAMSON C. (1992) " Path following and time-varying feedback stabilisation of a wheeled mobile robot ", Proc. Conf. ICARCV'92, Singapore, September 1992.
[SHI90] Shin D. H. & Singh S. (1990) " Vehicle and path models for autonomous navigation ", in *Vision and navigation : the Carnegie Mellon Navlab*, Thorpe C. E. ed., Kluwer Academic Publishers, Boston, pp. 283-307.
[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.

[VDM93] Van der Molen G. M. (1993) " Modelling and control of a wheeled mobile robot ", 1st IFAC International Workshop on Intelligent Autonomous Vehicles, pp. 289-294.