On the Use of Neural Techniques for Path Following Control of a Car-like Mobile Robot

Abderrazak Chatti[1]

Imen Ayari[2]

Pierre Borne[2]

Mohamed Benrejeb[1]

LARA, Ecole Nationale d'Ingénieurs de Tunis,[1]
BP. 37, Le Belvédère ,1002 Tunis,
TUNISIE
abderrazak.chatti@insat.rnu.tn
mohamed.benrejeb@enit.rnu.tn

LAGIS, Ecole Centrale de Lille,[2]
BP. 48, F 59651 Villeneuve d'Ascq Cedex,
FRANCE
ayari_i@voila.fr
p.borne@ec-lille.fr

Abstract: A neural approach for the control of a car-like mobile robot, proposed in this paper, is based on the use of an inverse neural model adapted to the path following problem. In order to assure path following with obstacle avoidance, a neuro-fuzzy approach is also developed for a sensor referenced control.

Keywords: Mobile robots, non holonomic systems, path following, neural or neuro-fuzzy modeling and control, inverse model, internal model control.

1. Introduction

On one side, the conception of autonomous mobile robots able to move in a given environment without any collision risk constitutes at the present moment the aim of many research works. On the other side, techniques based on the use of Artificial Neural Networks (ANN) are nowadays of a great interest in control and robotic domains. The fastness of treatment and their capacity of approximating complex non-linear functions motivate their use for mobile robot control [1,2,3].

Hence, is proposed in this framework firstly a neural network modeling and a control approach in order to solve path following problem for a car-like mobile robot. Then, it is presented a sensor-equipped robot control for path following with obstacle avoidance, using a neuro-fuzzy technique.

2. Problem Statement

The robot under consideration is a car-like one with non holonomic property that restricts its mobility in the sideways direction and with limitation of the steering angle [2,4]. To realize the robot control by changing the steering angle, the velocity is considered constant. For this aim, inverse and direct neural models of the proposed robot are elaborated and a path following is realized for different trajectories using direct inverse neural control and internal model one. Results are compared with feedback linearising control structure ones [4]. The case of obstacle avoidance of the robot equipped with a virtual sensor in the front direction [5,6,7], is then considered by the use of a specific neuro-fuzzy approach is used.

3. Kinematic Model of a Car-like Mobile Robot

The kinematic model of a four wheeled robot can be considered as a tricycle with a motion front wheel and drive back wheels, figure 1 [2], with :

R :	reference point,
x_R, y_R :	coordinates of the point R, middle of the back axle taken as a reference point of the robot,
θ:	heading angle of the robot ,
ϕ:	steering angle of the robot,
V_R :	reference point velocity,
L:	distance between axes .

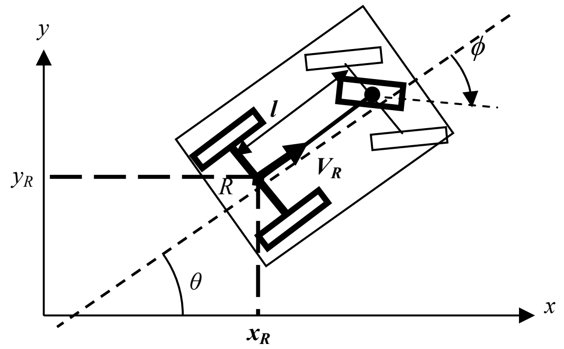


Figure 1: Geometric Model of a Car-like Mobile Robot

The inputs of this system are ϕ and V_R ; the outputs are (x_R,y_R) and θ.
In perfect adhesion conditions (movement without sliding), this kinematic model can be described by the following equations:

$$\begin{aligned}\dot{x}_R &= V_R \cos\theta \\ \dot{y}_R &= V_R \sin\theta \\ \dot{\theta} &= \frac{V_R}{l}\tan\varphi\end{aligned} \qquad (1)$$

The two first equations of the system (1) lead to the following equation:

$$-\dot{x}_R \sin\theta + \dot{y}_R \cos\theta = 0 \qquad (2)$$

which constitutes a non holonomic constraint, also called a constraint of non wheels sliding.

4. System Decomposition and Discretization Proposed for the Studied Car-like Robot

In order to control the car-like robot direction, a discrete time simulated model is elaborated; it represents the lateral dynamic of a vehicle by modelling separately the driving actuator and the vehicle kinematic, figure 2.

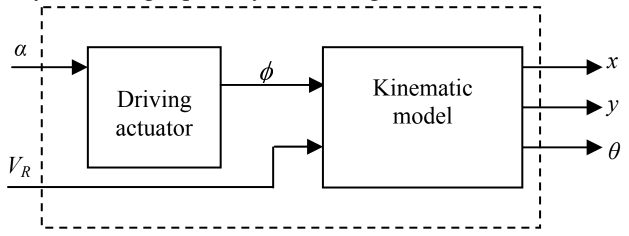


Figure 2: Structure of the Model of the Car-like Robot

The driving actuator is represented by a DC machine with limitation of the angle amplitude α: $\alpha \leq \alpha_{max}$=1.2 rd =67°, and the rotation velocity of the driving actuator: $d\alpha/dt \leq (d\alpha/dt)_{max}$ =0.5 rd/s, to express the wheels mechanical constraints existing in a real vehicle. The input of this block is the control angle α of the driving actuator and the output is the steering angle ϕ.

The inputs of the vehicle kinematic block are the steering angle ϕ and the velocity supposed constant in our work, V_R =0.5m/s. Outputs are the position (x,y) and the heading θ.
By discretization of the system (1) using Euler method [3], it comes:

$$\begin{aligned}x(k) &= x(k-1)+TV_R\cos(\theta(k-1)) \\ y(k) &= y(k-1)+TV_R\sin(\theta(k-1)) \\ \theta(k) &= \theta(k-1)+T\frac{V_R}{l}\sin(\theta(k-1))\end{aligned} \qquad (3)$$

with l= 1m and the discretization step T=0.05s.

5. Neural Networks Modelling of the Car-like Mobile Robot

The robot control is realized in this work using two neural modelling based structures: the control by the inverse model and the internal model one. Both proposed control strategies use a closed loop structure that allows good reference tracking and promising performances when dealing with measurement disturbances in a real vehicle. These structures need the elaboration of direct and inverse neural models of the robot.

5.1. On the direct model training

The Direct Neural Model (DNM) of the studied robot, figure 3, is obtained by exciting the robot model and the neural network with the same input vector and comparing the outputs according to the figure 4.

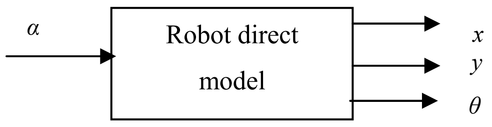


Figure 3: Robot Direct Model

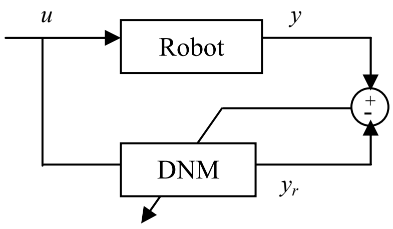


Figure 4: Direct Model Training

Then, the DNM training consists of the minimization of the following criterion *J*:

$$J = \frac{1}{2}\sum_{i=1}^{N}(y^i - y_r^i)^2 \qquad (4)$$

with :

N : number of training examples,

y^i : robot output,

y^i_r: DNM output.

Two types of direct modeling can be considered: a black box model and a semi physical one[2,3].

The Black box modeling needs to make the following choices of:

–the type of representation: input-output or state representation,
–the model order,
–the NARX (Non linear AutoRegressive with eXogen inputs) model assuming the existence of a state noise, or the NOE model (Non linear Output Error) assuming the existence of an output error, or the NARMAX (Non linear AutoRegressive with Mean Average and eXogen inputs) model assuming the existence of a state and an output noise.

If no information is given about the process, can be tried all possible representations and hypothesis, and used increasing model orders until obtaining a satisfactory model [3].
The robot black box model consists in a feedforward network with a single hidden layer.
The semi physical or grey box model can be considered as a compromise between a well known model and a black box one, taking in consideration available information about the process described by algebraic or differential equations. This model can use parameterized functions where parameters are determined by training [3].

The semi physical model of the studied robot is inspired from the one used by I.Rivals [2] to model the lateral behavior of a real vehicle. It exploits the information about the kinematic model of the robot composed of a black box model and a semi physical one.

The black box model of the driving is considered as a first order non linear system with a predictor in the form:

$$\Phi(k)=f(\Phi(k-1),\ \alpha(k-1),\ W) \tag{5}$$

where f is a non linear function and W is the weights matrix.

The corresponding neural network model is composed of 2 hidden neurons with sigmoidal activation functions and an output linear neuron. The training algorithm used is the backpropagation algorithm with simple gradient.

The semi physical model of the dynamical robot behaviour describes the kinematic model (3). In fact, supposing a perfect adhesion of the system, the two first equations are always verified, while the third one relating the heading θ to the steering angle ϕ is not surely verified for a real robot and depends on the direction mechanism. This equation is consequently approximated by a non linear neuron [2].

By approximating the function 'tan' by a sigmoidal neuron, the neural predictor of the last equation of the system (3) becomes:

$$\theta(k)=\theta(k-1)+T\frac{V_R}{l}\ w_2\ g(\varphi(k-1),w_1) \tag{6}$$

where g is the function realized by the network, w_1 and w_2 are weights to determine.

Table 1: Mean Square Error of Trained Models

MSE	Black-box model	Semi-physical model
x	0.9040	0.0307
y	0.7640	0.0315
θ	0.4548	0.0181

Both types of models were simulated; the summary of the results is shown in table 1 in the form of Mean Square Error (MSE) for an easier comparison. The two columns show the prediction error respectively for the black box model and the semi physical one for all outputs. It is clear that the error for the semi physical model is better; this is justified by the use of a well known kinematic model. Thus, the latter is used for the internal model control design.

5.2 Neural Inverse Model Characterization

The Inverse Neural Model (INM) structure to elaborate is so that this model gives, at every step Δt, the control angle α corresponding to the realized movement (distance r and angle variation $d\theta$ representing the difference between previous θ and actual θ), figure 5. The movement is computed according to the robot reference.

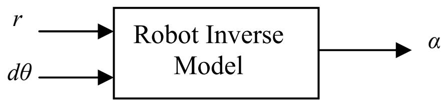


Figure 5: Robot inverse Model Structure

For the design of the robot INM, a normalized multi-layer perceptron with a hidden layer composed of sigmoidal hidden neurons and with a linear output neuron.

The elaborated network realizes the function h as follows:

$$\alpha(k)=h[r(k),\ r(k-1),\ r(k-2),\ d\theta(k),\ d\theta(k-1),\ d\theta(k-2),\quad \alpha(k-1),\ \alpha(k-2)\ ] \tag{7}$$

Inputs r and $d\theta$ are normalized respectively in [0, 1] and [-1,1] intervals and the output α in [-1,1] interval.

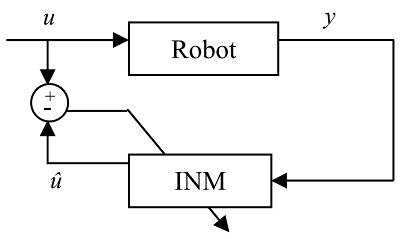


Figure 6: Inverse Neural Model Training Structure

The backprobagation training algorithm minimizing the following criterion J :

$$J = \frac{1}{2}\sum_{i=1}^{N}(u_i - \hat{u}_i)^2 \tag{8}$$

is used in two versions, the simple gradient version and the gradient with a momentum term, figure 6, with :

u_i : robot input,

$\hat{u}_I$: INM output.

The simple gradient version consists in modifying the weights w according to the following formula:

$$w(k) = w(k-1) + \Delta w(k) \tag{9}$$

with :

$$\Delta w(k) = \mu \frac{\partial J}{\partial w} \tag{10}$$

where μ is the training step.

The gradient with a momentum consists in adding a momentum term λ to the weights modifying formula:

$$\Delta w(k) = \mu \frac{\partial J}{\partial w} + \lambda\, \Delta w(k-1) \tag{11}$$

The last gives better results in simulation than the one with simple gradient since it takes into account the last weights modification. The figure 7 presents the results of test sequence and generalized error for μ = 0.2 and λ=0.2.

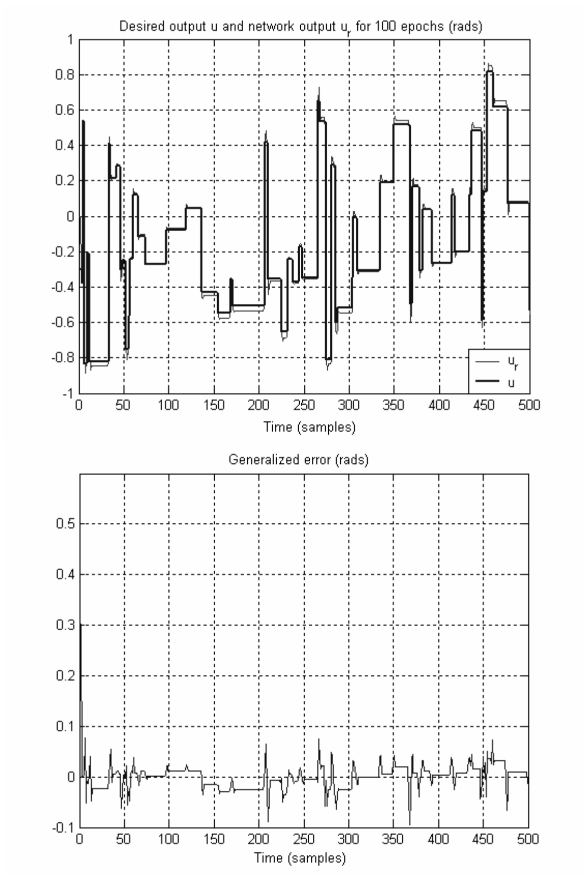


Figure 7: Network Output and Prediction Error

The MSE of the trained model is 0.0323; the model is then considered satisfactory since it generates a small generalized error.

6. Neural Control for Path Following Problem

In this section, the robot path following using the elaborated inverse model as a controller, is simulated for different reference trajectories with different curves. The results are compared to those of an internal model control exploiting the generated inverse and direct models of the robot.

6.1 Controller Using the Inverse Model for Path Following Problem

To guide the studied robot, the produced Inverse Neural Model (INM) is used, as shown in the figure 8.
At each sample, the movement, represented by the distance *r* and angle variation *dθ*, is given to the controller represented by the INM to let the robot reach the next reference point, figure 9. The inverse model elaborated has as reference inputs *r* and *dθ*, and should generate the appropriate command *α* allowing executing the right movement. The realized structure is then a closed loop one.

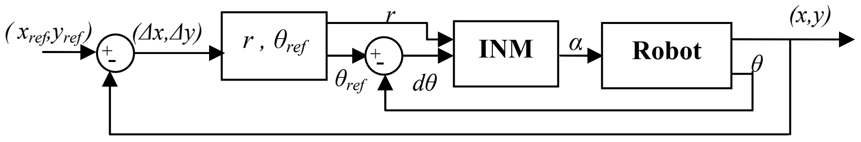


Figure 8: Controller Using the Inverse Model of the Robot

The movement parameters are computed according to the robot reference as below:

$$r = \sqrt{(\Delta x)^2 + (\Delta y)^2} \tag{12}$$

$$d\theta = \theta_{ref} - \theta \tag{13}$$

with :

$$\Delta x = x_{ref} - x$$

$$\Delta y = y_{ref} - y$$

$$\theta_{ref} = a\tan\left(\frac{\Delta y}{\Delta x}\right)$$

In order to test the designed controller, different reference paths are chosen to provide several direction changes and types of curves (continuous and discontinuous).

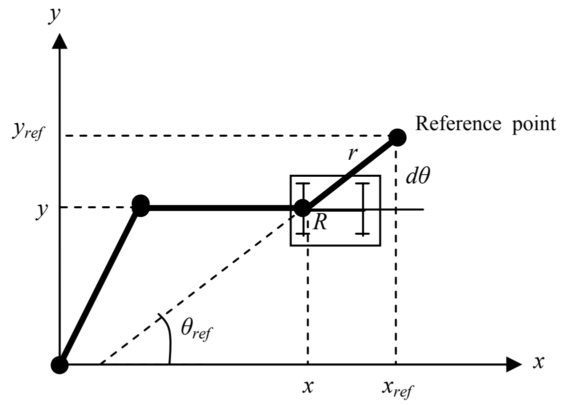


Figure 9: Robot Movement in the Reference Trajectory

Sinusoidal trajectory reference case (continuous curvature)

The reference trajectory is a sinusoidal-like curve presenting a continuous curvature and well-matched with the simulated robot steering ray. The figure 10 shows the robot trajectory. The path following is satisfactory, a small tracking error exists because of the little inverse model mismatch.

Discontinuous trapezoidal trajectory

The application of a trapezoidal trajectory composed of several segments (discontinuous curve which does not respect the kinematic constraints of the robot) leads to the robot trajectory of the figure 11. The path following is good, the robot tends to overlapping the trajectory by few centimeters at the first turning; this is due to the curvature discontinuity which leads to an abrupt change of *Φ* at segments end.

Discontinuous N shaped trajectory case

The application of an N shaped reference trajectory composed of segments and presenting two 90° attenuated angles (not respecting the kinematic constraints of the robot and steering angle limitation) leads to the results of the figure 12. The path following, in this case, is quite satisfactory but less good than previous cases; the tracking error exists at the 90° angles of the trajectory, this is due to the sudden direction change and especially to the limitation of the angle amplitude constraint : α in [-67°, 67°] interval.

6.2 Neural Internal Model Control of the Robot

In this part, the Internal Model Control (IMC) is applied in order to compare the results with those of the direct inverse one. The same INM and the DNM previously generated are used.

This structure allows having an off set free response by canceling the noise added to the process output. The figure 13 gives the adopted control structure.

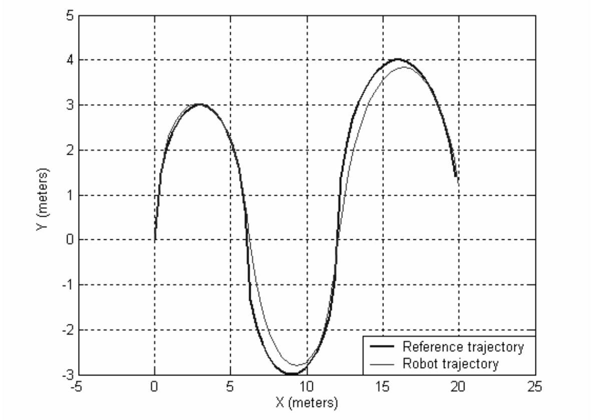
For any output, y for example, it corresponds to the real physical system output, which is, in fact the sum of the ideal signal y^* and the noise p added to the output.

The framed block computes the INM inputs from the reference y^*_{ref} and the ideal system output y, represented by the output of the DNM with the assumption that y^* isn't accessible.

No disturbances at the output are considered here; then :

$$p = 0$$
$$e = y - \hat{y} = y^* - \hat{y} \tag{14}$$

For a perfect DNM, we get $\hat{y}=y^*$ and $e=0$, the structure of the control is the same of the previous one.



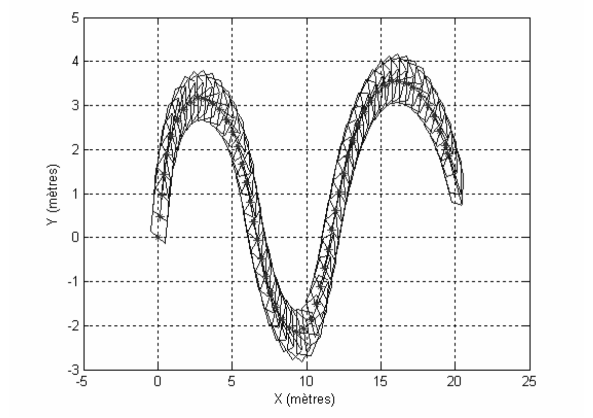


Figure 10: Robot Response for Sinusoidal Reference Trajectory

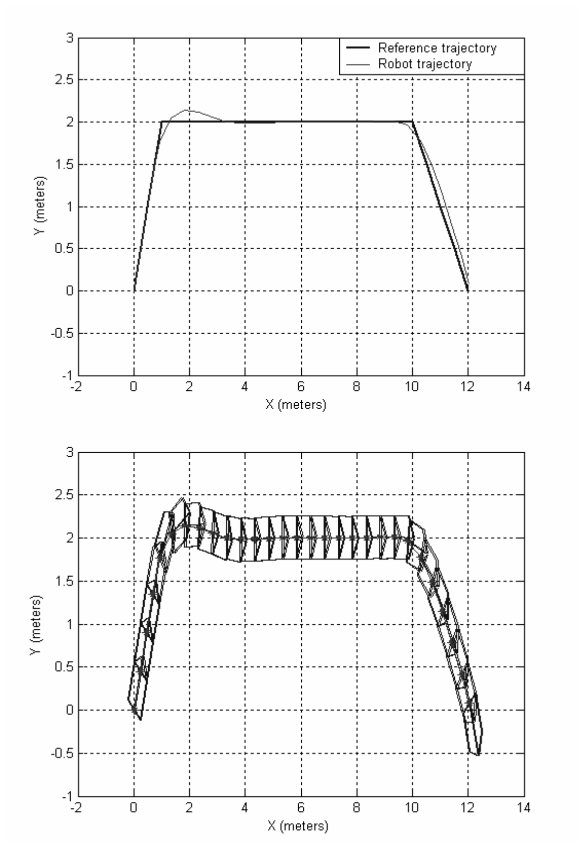


Figure 11: Robot Response for Trapezoidal Reference Trajectory

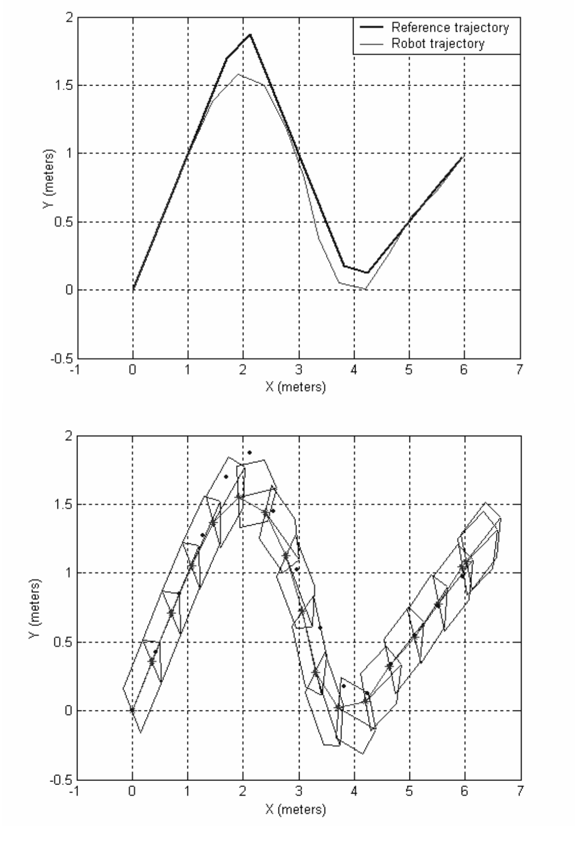


Figure 12: Robot Response for N Shaped Reference Trajectory

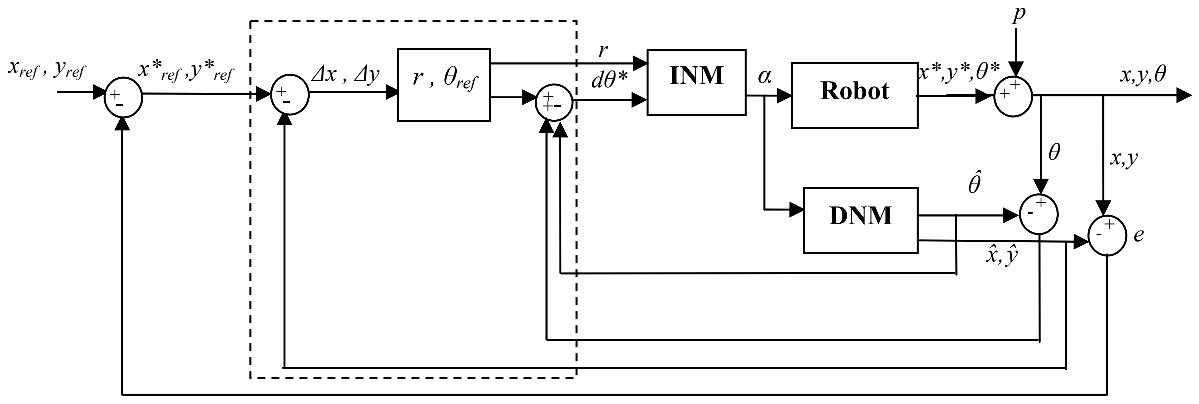


Figure 13: Proposed Internal Model Control Structure

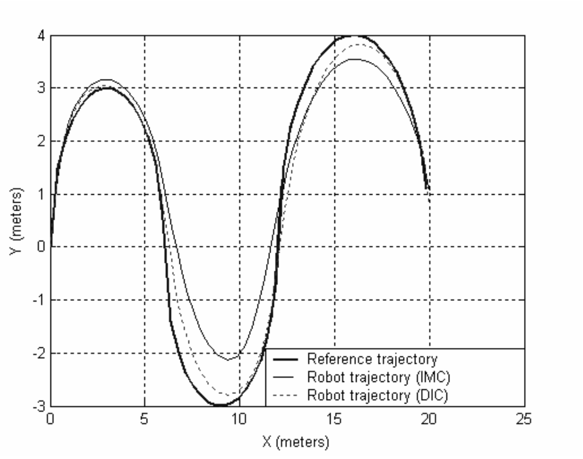


Figure 14: Robot Response for Sinusoidal Reference Trajectory (DIC and IMC)

According to the used control structure analysis, the same response as the direct inverse control has to be found, but the figure 14 shows that the result is little less satisfactory than the previous one, this is due to the small prediction error of the DNM.

6.3 Comparing the Neural Control with a Feedback Linearising Control

A feedback linearising control performances [4] is here compared to the open loop control ones.

Feedback linearising control

The main idea of this type of control, proposed by A. Kamaga and A. Rachid [4] for a mobile robot control for path tracking, is to make the pursuit of each line composing the reference trajectory by imposing a steering angle command calculated in this way:

$$\varphi_i = atan(l[k_1 y_i + k_2 \tan\psi_i] \cos^3 \psi_i)$$

(15)

with

ϕi : command angle allowing the robot to join the line *i*,

ψi : orientation difference between the line *i-1* and the line *i*,

yi : *y* coordinate of the robot reference point in the mark composed of the line *i* and its perpendicular on the first of the line *i*,

k1,k2: coefficients to adjust to apply the appropriate command.

ψ and ϕ are in the]-90°,90°[interval.

The new command angle ϕ_{i+1} allowing the robot to reach the line *i+1* is applied, for each line, at a particular distance d_i at the end of the line *i*. This distance *d*, called security distance, is computed as the following:

$$d_i = \frac{k_2}{k_1 \cos \psi_{i+1}} \tag{16}$$

Simulation results

The same reference trajectories as previously are chosen for the simulation in order to compare the results.

The following values: *k1*=4 and *k2*=4, giving the best results, are chosen for simulation of the control algorithm. The figure 15 show the robot trajectory for the sinusoidal-like reference trajectory and the N shaped one.

For the application of this control structure, only the kinematic model is used without any driving actuator, the command is applied to the steering angle *Φ*.

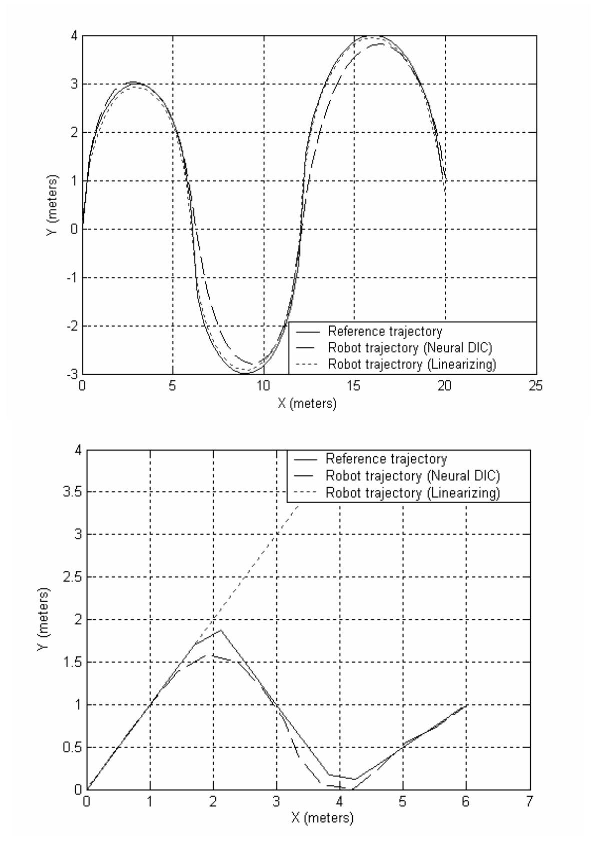


Figure 15: Robot Trajectories

According to the figure 15, the path following using feedback linearization gives good results for the first reference curve but catastrophic ones for the second; this is due to the attenuated 90° angle of the trajectory. The neural controller gives satisfactory results for both cases. This shows the robustness of the neural control, adding the fact that it doesn't need any analytic model of the robot that is the most time imperfect and difficult to find.

7. Sensor Referenced Control for Path Following with Obstacle Avoidance

7.1 Basic Idea

The main idea of this part is based on the use of a neuro-fuzzy approach to realize a path following with obstacle avoidance. The robot have to follow the reference trajectory; when encountering an obstacle, it should avoid it and join this reference trajectory as soon as possible [1,5,7,8].

In order to take the obstacle existence into account, the robot model is endowed with a virtual sensor, placed in the front of the robot, which computes the distance to the obstacle, figure 16.
The information issued from the sensor represents 3 distances: a distance in front of the robot *d* and 2 laterals ones (left *d_L* and right *d_R*) limiting the detection zone of the sensor defined by:

$d_{max}=d\_L_{max}=d\_R_{max}$ (17)

and by the width of the detection sector β, figure 16.

If the minimal distance d to the obstacle in front of the robot is superior to d_{max} the sensor gives d_{max} , figure 16. Otherwise, it gives the measured frontal distance d and the lateral ones d_L and d_R, figure 17.

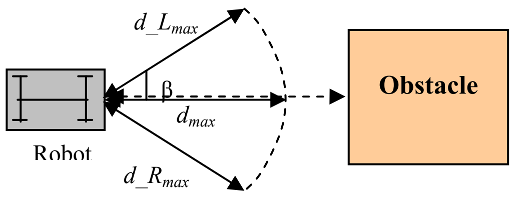


Figure 16: Far Obstacle Case

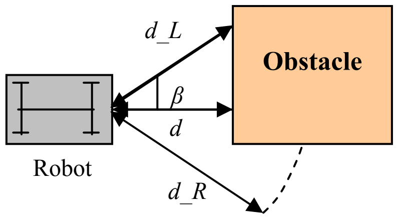


Figure 17: Close Obstacle Case

The obstacle avoidance is made as the following:

–if $d < d_{max}$ the obstacle existence is confirmed,
–if $d\_L > d\_R$ then the robot goes right to avoid obstacle,
–otherwise, it goes left.

The path following is realized by the control of the robot heading, the program computes at each time dθ to apply in order to join the reference trajectory.

7.2 Neuro-fuzzy Adopted Control

The chosen neuro-fuzzy approach for the control combines the advantages of both neural and fuzzy ones.

A multi-layer network whose structure describes the adopted fuzzy system is presented here.

Input/Output variables

The network inputs are $d\theta$ and the distances issued from the sensor: d and $d\_LR= (d\_L- d\_R)$.
The output is the steering angle ϕ.

Fuzzy sets

The fuzzy sets describing the output variables represent the first hidden layer.
The activation function of each neuron corresponds to the membership function in the fuzzy system.
The variable $d\theta$ has 7 fuzzy sets: {NegB, Neg, NegS, Nul, PosB, Pos, PosS}
The variable d has 3 fuzzy sets: {Near, Middle, Far}
The variable d_LR has 2 fuzzy sets: {Pos, Nul, Neg}

Fuzzy inference system

The inference system used is a Sugeno of zero order, is represented by the next layers which calculate the truth degree of premises in each rule.
The rules are defined as following examples :

–example of path following rules (weight =1)

IF ($d\theta$ is NegB) AND (d is far) THEN ϕ =-1 ; (18)
IF ($d\theta$ is Neg) AND (d is far) THEN ϕ =-0.75 ; (19)

–example of avoiding obstacle rules (weight=2)

IF(*d* is near) AND (*d_LR* is Pos) THEN ϕ =1 ; (20)

IF (*d* is near) AND (*d_GD* is Nul) THEN ϕ =-1 ; (21)

7.3 Neuro-fuzzy System Structure

The figure 18 shows the neuro-fuzzy architecture for the rules (18) and (20) [1].

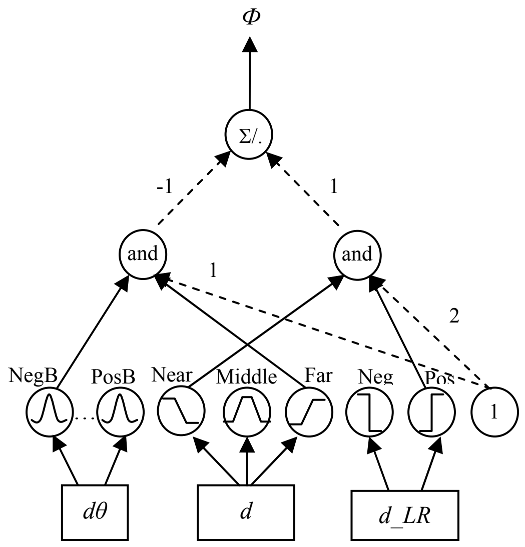


Figure 18: Neuro-fuzzy Network Architecture

The obstacle avoidance rules have priority than the path following ones, that's why they have a superior weight.The output is calculated with the average sum method, witch corresponds to a neuron with a normalized sum activation function.

7.4. Simulation Results

The figure 19 presents the simulation results for a trapezoidal reference trajectory following, intercepted by a circular obstacle.

For this aim is used the previously defined architecture without any parameters training.

We effectively note the obstacle avoiding behaviour of the robot, and the quite satisfactory response. However, when increasing the vehicle velocity, the robot behaviour changes and the controller performance can be lost.

A solution to improve controller performances can consist in making the parameters training of the elaborated fuzzy system. In fact, the training process could affect the output of each rule and the rules weights, these parameters are in dashed line in figure 18. We could also extend the training process to the membership functions of the input variables. This needs, in the case of real robot, to execute several obstacle avoiding situations at different velocity values to serve as a base of training examples.

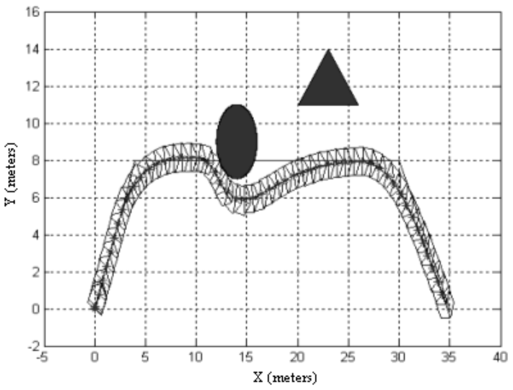


Figure 19: Robot Trajectory with Obstacle Avoidance

8. Conclusion

The neural controller using the direct inverse structure has given satisfactory results when applied to the simulated car-like robot model. The use of such controller for a real robot is promising since this work takes into account the non linearities and the mechanical constraints existing in a real vehicle.

For the sensor referenced control, the results are also satisfactory; however, a training of some of the neuro-fuzzy network parameters could improve the controller performances and guarantee its robustness.

REFERENCES

1. E. GAUTHIER, **Utilisation des réseaux de neurones artificiels pour la commande d'un véhicule autonome**, Thèse de Doctorat, Inst. Nat. Polytechnique de Grenoble, 1999.

2. I. RIVALS, **Modélisation et commande de processus par réseaux de neurones; application au pilotage d'un véhicule autonome**, Thèse de Doctorat, Université de Paris 6, 1995.

3. G. DREYFUS, J. M. MARTINEZ, M. SAMUELIDES, M. B. GORDON, F.BADRAN, S. THIRIA et L. HERAULT, **Réseaux de neurones, méthodologie et applications**, Paris, Editions Eyrolles, 2002.

4. A. KAMAGA and A. RACHID, **A simple Path Tracking Controller for Car-Like Mobile Robots**, 1997.

5. A. SAFFIOTTI, **The uses of fuzzy logic in autonomous robot navigation**, Soft Computing, 1(4):180-197, 1997.

6. P. Y. GLORENNEC, **Un réseau neuro-flou évolutif**, In Neuro-Nimes, pp 301-314, Nimes, 1991.

7. C. LAUGIER, T. FRAICHARD, P. GARNIER, and I. PAROMTCHIK, **Sensor-based control architecture for a car-like vehicle**, In Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems, Vancouver, 1998.

8. A. LABBI and E. GAUTHIER, **Combining fuzzy knowledge and data for neurofuzzy modeling**, Journal of Intelligent Systems, 7(1-2):145-163, 1997.

9. J. P. LAUMOND, **La robotique mobile**, Collection of Articles and Researshs, Paris, Hermès Sciences Publications, 2001.

10. M. SAMPEI, **Arbitrary Path Tracking Control of Articulated Vehicles Using Nonlinear Control Theory**, IEEE Transaction, on Control Systems Technology, Vol. 3, pp 125-131, 1995.

11. Y. KANAYAMA, Y. KIMURA, F. MYAZAKI and T. NOGUCHI, **A Stable Tracking Control for an Autonomous Mobile Robot**, IEEE International Workshop on Intelligent Robots and Systems, Osaka, pp 1236-1241, 1991.

12. S. BEL HADJ ALI, **Sur la commande par modèle interne de processus dynamiques**, Thèse de Doctorat, Ecole Nationale d'Ingénieurs de Tunis, Tunis, 2003.

13. A. YAICH, **Sur la modélisation et la commande des systèmes en incluant les concepts des réseaux de neurones et de la logique floue**, Thèse de Doctorat, Ecole Nationale d'Ingénieurs de Tunis, Tunis, 2002.

14. K. BEN SAAD, **Modélisation et commande d'un moteur pas à pas tubulaire à réluctance variable et à quatre phases, Approches conventionnelles, par logique floue et par réseaux de neurones artificiels**, Thèse de Doctorat, Ecole Nationale d'Ingénieurs de Tunis, Tunis, 2005.

15. A. ABDELKRIM, **Contribution à la modélisation du processus d'écriture à la main par approches relevant du calcul évolutif**, Thèse de Doctorat, Ecole Nationale d'Ingénieurs de Tunis, Tunis, 2005.

16. 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, Vol. 2, pp 1236-241, Osaka, 1991.

17. P. REIGNIER, **Pilotage réactif d'un robot mobile : étude du lien entre la perception et l'action**, Thèse de doctorat, Inst. Nat. Polytechnique de Grenoble, 1994.

18. I. RIVALS, L. PERSONNAZ, G. DREYFUS and D. CANAS, **Real-time control of an autonomous vehicle: A neural network approach to the path following problem**, In Neuro-Nimes, pp 219-229, Nimes, 1994.