Evolutionary Method for Designing and Learning Control Structure of a Wheelchair

Imen BEN OMRANE[1,2], Abderrazak CHATTI[1], Pierre BORNE[2]

[1] Institut National des Sciences Appliquées et de Technologie INSAT,
Centre Urbain Nord BP 676, Tunis, 1080, Tunisia,
imenbo7@yahoo.fr, pierre.borne@ec-lille.fr.

[2] Ecole Centrale de Lille ECLille,
Cité Scientifique Villeneuve-d'Ascq, Lille, 59650, France,
abderrazak_chatti@yahoo.fr

Abstract: This article describes an aspect of evolutionary robotics for trajectory tracking. We will combine genetic algorithms with neural networks for modelling and controlling a wheelchair for disabled people. The interest of the hybridization of Neural Networks (NN) with Evolutionary Algorithms (EA) in robotics is based on the observation that a local search by a gradient descent method is replaced by a global search performed by EA. The gradient descent methods are subject to variations in performance due to the initial position of the NN, which sometimes leads to a convergence towards local minima. In contrast, the proposed evolutionary methods provide a global research of both the structure and the weights of the neural net. The control structure used for robot trajectory tracking control is based on the Internal Model Control (IMC) which direct neural model was learned with our new EA.

Keywords: Evolutionary Robotics, trajectory tracking, evolutionary algorithms, Neural Networks, direct neural model, mobile robots, wheelchairs, Internal Model Control.

1. Introduction

The most common technique for training neural networks is by studying the variations of the gradient descent. This technique suffers from well known problems, essentially local minima. Hence, there is a need for more efficient and effective methods to determine network weights and structure of NN. These methods combine another biologically inspired technique, the technique of genetic algorithms with neural networks.

Developed by John Holland [3], a genetic algorithm is a biologically inspired search technique. In simple terms, the technique involves generating a random initial population of individuals, each of which represents a potential solution to a problem. Members of the population are then selected for reproduction based upon fitness function, and a new generation of potential solutions is generated. The process of evaluation, selection, and recombination is iterated until the population converges to an acceptable solution.

Several hybridization of genetic algorithm and neural network exist; the most common among them are the determination of network weights by the use of genetic algorithms [7], [11], [14] and the evolutionary design of the network architecture [1], [4], [l0], [12], [15].

Evolutionary methods have found applications that span the range of architectures for intelligent robotics. For example, evolutionary algorithms have been used to learn rule sets for rule-based autonomous agents, topologies and weights for neural nets for robotic control [8], [9], [14], [15] fuzzy logic control systems [19], and rules for behaviour-based robots [13], [2].

In this paper, the NN is learned with EA and is using Internal Model Control (IMC) structure in order to benefit from performances of each one of them to control a wheelchair for disabled people. Inverse and direct neural models of the wheelchair are elaborated and a trajectory tracking is realised.

We have chosen IMC strategy of control because it constitutes a powerful strategy of control for complex systems, thanks to its simplicity of implementation, its robustness toward the errors of modelling and its facility of adjustment. [20]

We make lateral control for position and direction of the wheelchair for disabled people from angle control u. The speed of the wheelchair will be considered low and constant. This is justified by the fact that the wheelchair is not subject to run at variable and high speed. Thus, dynamic model will not be considered.

Learning for modelling will be performed by two techniques:

- Standard: back propagation gradient.
- Evolutionary: hierarchical genetic algorithm.

The evolutionary method has simultaneously to determine the structure of NN and to learn it by minimizing the squared error

$$J = \frac{1}{2}\sum_{i=1}^{N}(S_d - S_r)^2$$

S_d is a desired output and S_r a network output.

We will compare these two methods of learning by simulating prediction error. Finally, we apply the IMC using the models already developed.

2. Simulation Model of the Wheelchair

For wheelchair model, we used the kinematic model with differential drive, which is valid in perfect adhesion conditions.

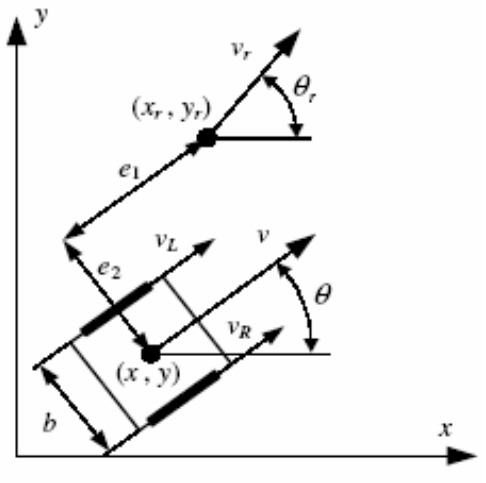


Figure 1. Wheelchair Model

Kinematics of a wheelchair with differential drive is defined by a Jacobian matrix J(θ) that transforms tangential and angular velocities expressed in the coordinates of the wheelchair base to its velocities expressed in global Cartesian coordinates (Figure 1):

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = J \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

The coordinate (x,y) gives the position of the wheelchair in the Cartesian space and θ angle position of the latter from the x-axis (Figure 1).

(xr,yr) is the reference position in cartesian space, θr is the reference robot orientation, and vr is the reference tangential velocities.

3. Evolutionary Algorithms Proposed to Simultaneously Determine Weights and Structure of a Neural Network for ROBOTIC

The natural evolution has created very complex biological systems adapted to many conditions. Its mechanisms are based on the principle of competition between individuals. The best adapted individuals survive and can create descendants who spread their genes.

Instead of using back propagation to train the networks over and over again, it seems to be a valid idea to have the evolutionary algorithm (EA) search for both structure and weights simultaneously.

EA have different classes which differ only on the implementation details of operators and the procedures for selection and population replacement. In our work, we used a class of EA, which is the hierarchical genetic algorithm (HGA). This choice based on that HGA is used for optimization of both the weights and structure of NN. The advantage of this approach is that genes of chromosome are classified into two categories (hierarchy). This is representing the order relations between:

- network layers,
- number of neurons (input, hidden, output)
- connection weights

Each chromosome consists of two types of genes:

- control genes (bits or integer) for the activation of neurons in the hidden layer,
- genes connections (real) for the determination of synaptic weights

3.1 Wheelchair neuronal model

The neural control structure that we apply, IMC, requires direct and inverse models of the wheelchair. Direct model was learned using standard back propagation gradient and evolutionary technique which is HGA. Then, we compare the performance of the various individuals to select the one with the nearest behaviour to the real system.

3.1.1 Direct neural model

Identify direct neural model which can reproduce outputs x, y, θ close to those of a wheelchair for the same control angle u.

Figure 2. Direct Model wheelchair

To design a model of the wheelchair, we follow the steps indicated below in making the necessaries choices;

- Choice of representation: Input Output representation
- Choose of noise assumption: we choose NARX model,
- Choice of model order: it is a system of order greater than one we choose for those two previous outputs and two previous inputs in the regressor.

a) Choice of network architecture

The system model is a multivariable, (MIMO) one. It contains one input and three outputs coupled. So we have selected a network composed of three sub-networks, each one related to a given output, and which takes into account the coupling between the three outputs. See Figure 3.

The network predictor performs three functions. The nonlinear regressor of each of these functions contains the past values of the control input and of three output variables as follows:

$x_r(k)=\varphi_1$ (x(k-1), x(k-2),y(k-1), y(k-2), θ(k-1), θ(k-2), u(k-1), u(k-2); C1)

$y_r(k)=\varphi_2$ (x(k-1), x (k-2), y(k-1), y(k-2), θ(k-1), θ(k-2), u(k-1), u(k-2); C2)

$\theta_r(k)=\varphi_3$ (x(k-1), x(k-2), y(k-1), y(k-2), θ(k-1), θ(k-2), u(k-1), u(k-2); C3)

Where:

- x_r, y_r, θ_r: neural outputs,
- x, y, θ: system outputs,
- φ_1, φ_2, φ_3: functions of network,
- C1, C2, C3: sub network parameters.

b) Back propagation learning

The value of learning rate is chosen μ = 0.1 and the maximum number of iterations is 500.

Validation of the model is found after learning by applying to the network inputs the test sequence and comparing outputs with the desired one.

We evaluate the performance of the model found by calculating the prediction error for each output which represents the generalization error of each sub-network.

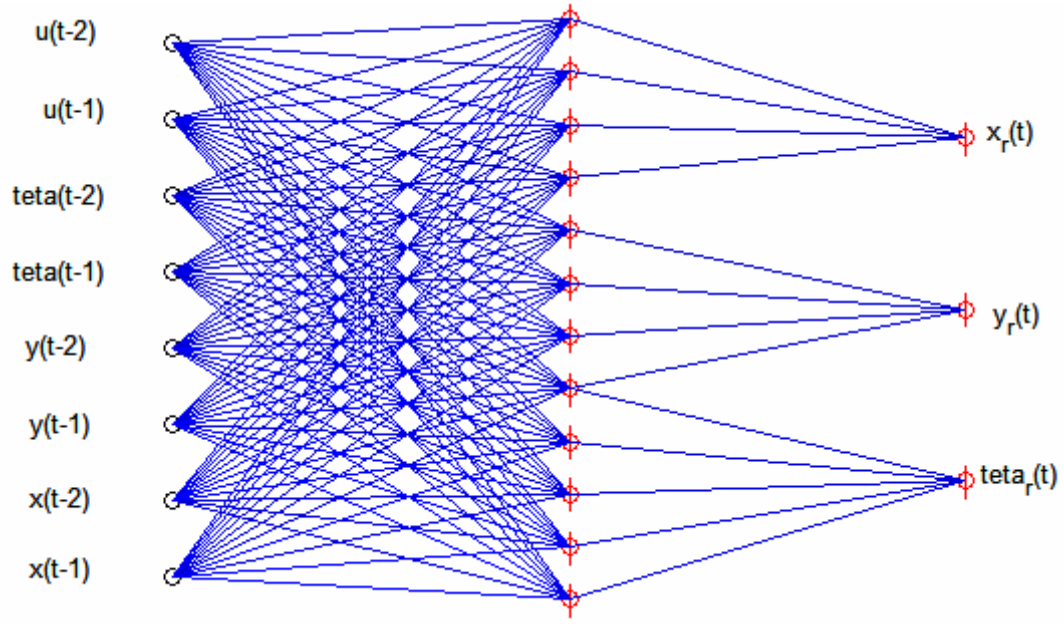


Figure 3. Neural Network Architecture

Figure 4 represents respectively the vectors of inputs and outputs and also shows the prediction error on each output.

c) Hierarchical genetic algorithm learning

We construct a HGA which generates various structures and selects a structure of a Multi

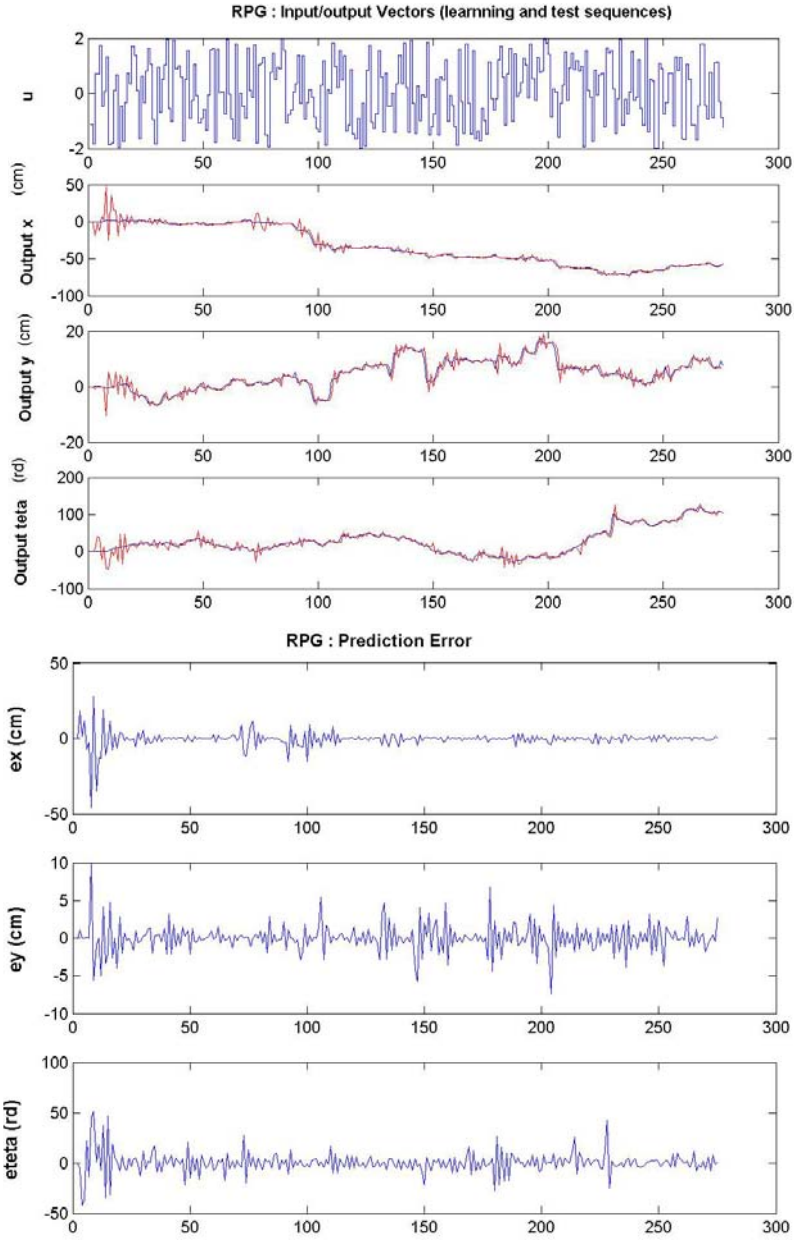


Figure 4. Back Propagation: inputs/outputs sequences and prediction error

We note that network outputs follow the evolution of the references, but they have several differences. Indeed, the prediction error goes up to 0,6 m for x and y, and 20 rd for θ.

The difference was large enough to prove the bad quality of the model. So we will try to train this direct model by using another method which is HGA.

Layer Perceptron (MLP) with one hidden layer. The MLP selected would be the best one that satisfies the criterion.

The number of neurons in the hidden layer and the different weights of the connections are given after evaluation of a cost function. This function to minimize corresponds to the following quadratic criterion:

$$J=\frac{1}{N}\sum_{i=1}^{N}\left(y_{di}-y_i\right)^2 \quad (3)$$

with:

- N: Number of examples,
- y_{di}: process Output,
- yi: neural network output of neurone i.

Our algorithm will go to several architectures of neural network and will train it.

a. Coding

Our chromosome is represented by a matrix. The first line is encoded in binary: "1" if the hidden layer neuron is considered and "0" if it is ignored. Remaining lines contain real numbers that represent the input and output weights of our NN.

When knowing the system inputs and outputs number, the matrix size is fixed. The direct neural model (DNM) that will be trained has 8 inputs and 3 outputs (Figure 5).

The maximum number of neurons (number of columns) that may contain the hidden layer is set by the user. We have considered a maximum of 15 hidden neurons.

b. Initial population

It is necessary to create and maintain sufficient genetic diversity in the population. For this reason, the initial population should be as heterogeneous as possible to prevent premature convergence. So, we have opted to randomly generate n individuals of the initial population.

But not having a search space far from the minimum desired and to minimize the search time, we decided to inject into the initial population a number of individual from backpropagation learning.

c. Selection.

We select groups of individuals based on their evaluation by the function J. Individuals having the lowest cost function will be selected.

d. Crossover and mutation

The genetic operators we create can change both the structure of the NN and the weights of the various existing connections. This produces different architectures taken into consideration during the learning.

	1	0	0	1	1	1	1	1	0	0	0	1	1	1	0
u(t-2)	W11e	W21e	W31e	W41e	W51e	W61e	W71e	W81e	W91e	W101e	W111e	W121e	W131e	W141e	W151e
u(t-1)	W12e	W22e	W32e	W42e	W52e	W62e	W72e	W82e	W92e	W102e	W112e	W122e	W132e	W142e	W152e
teta(t-1)	...	...													
teta(t-2)			...	...											
y(t-2)						...	...								
y(t-1)									...	...					
x(t-2)												...	...		
x(t-1)														...	...
xr(t)	W1s1	W1s2	W1s3	W1s4	W1s5	W1s6	W1s7	W1s8	W1s9	W1s10	W1s11	W1s12	W1s13	W1s14	W1s15
yr(t)	W2s1	W2s2	W2s3	W2s4	W2s5	W2s6	W2s7	W2s8	W2s9	W2s10	W2s11	W2s12	W2s13	W2s14	W2s15
tetar(t)	W3s1	W3s2	W3s3	W3s4	W3s5	W3s6	W3s7	W3s8	W3s9	W3s10	W3s11	W3s12	W3s13	W3s14	W3s15

Figure 5. Illustration of the chromosome coding

For creating the sub network, we will make some treatment: we consider the activated hidden neurons and we distribute it for outputs (x, y and θ)

We have started with an initial population of 100 individuals, each of which represents a structure of NN.

For each individual, we calculated the cost function J. Only the best 10% individuals pass to the next generation, 70% following individuals were created by crossing and the remaining 20% by mutation. Figure 6 illustrates the obtained results.

We have used 2 crossover types:

The first one was called INTRA. This one changes both; structure and weights of NN.

Example:

We choose randomly 2 parents and a crossover point to create 2 children. See Figure 6.

The children created present 2 different NN in the population.

The second one was called INTER. This crossover will only change weights of NN but not the structure of NN. The structure which is

presented by the first line of chromosome is still unchanged. See Figure 7.

For mutation, we choose to modify the structure only by changing one bit (which represents neuron) in the first line of chromosome (0 to 1 or 1 to 0). See Figure 8.

e. Results

We have run our HGA for 200 generations. So it creates 200*100=20000 individuals, which means 20000 structures of NN. The best solution is illustrated in Figure 9. The number of considered hidden neurons is 15 and the outputs connections of hidden layer were changed. S

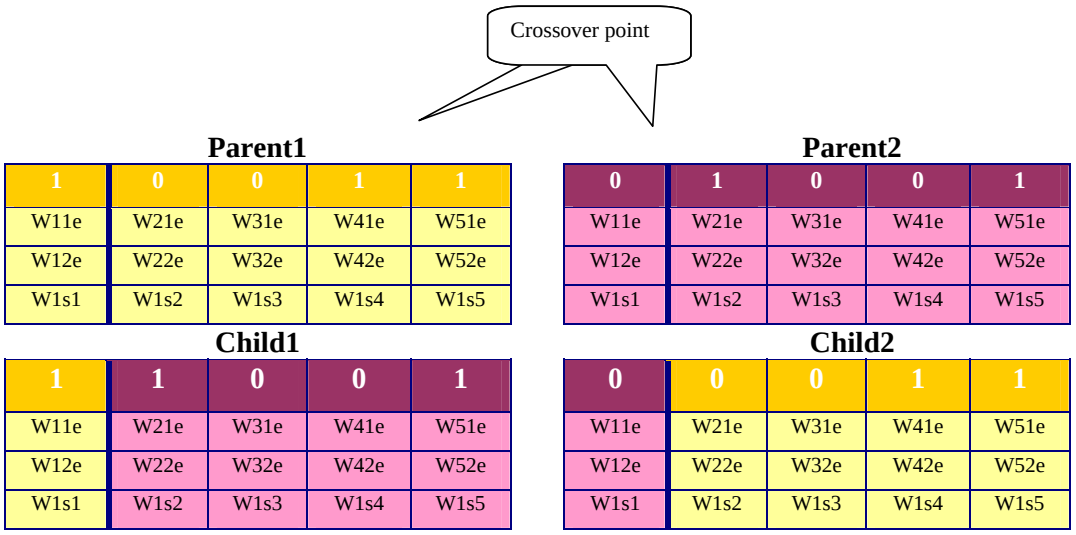


Figure 6. Illustration of "Crossover INTRA"

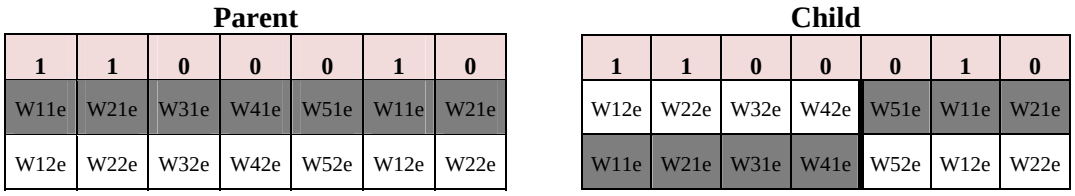


Figure 7. Illustration of "Crossover INTER"

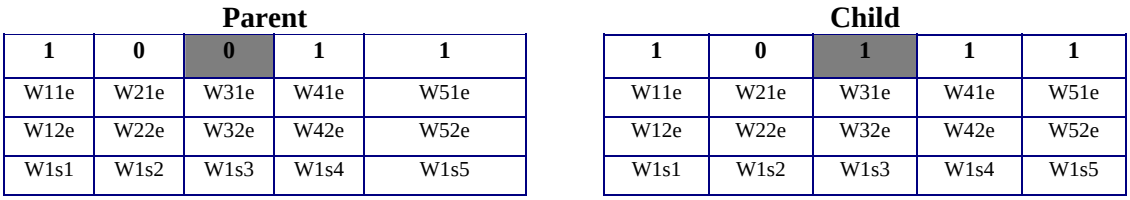


Figure 8. Illustration of "Mutation"

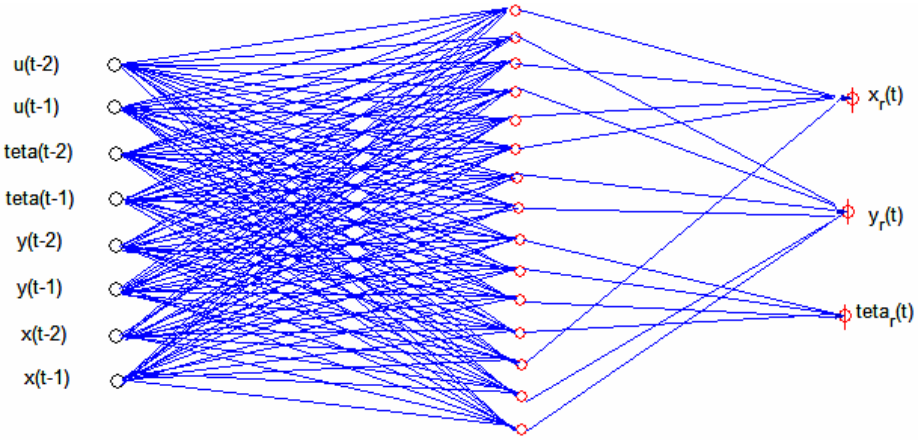


Figure 9. New Neural Network Architecture

The validation of the model is found after learning by applying the test sequence to the network input and comparing output with the desired one.

We evaluate the performance of the obtained model by calculating the prediction error for each output which represents the generalization error of each sub-network.

Figure 10 represents respectively the vectors of inputs and outputs, and also shows the prediction error on each output.

Comment: To learn our NN, HGA running takes several times as a consequence of size of the chromosome and size of population. But, by considering the considerable minimisation of the error compared to standard Back propagation learning, we can consider that the time is a false problem.

3.2 Inverse neural model

The aim is to approximate the inverse model of the wheelchair: the function giving the control

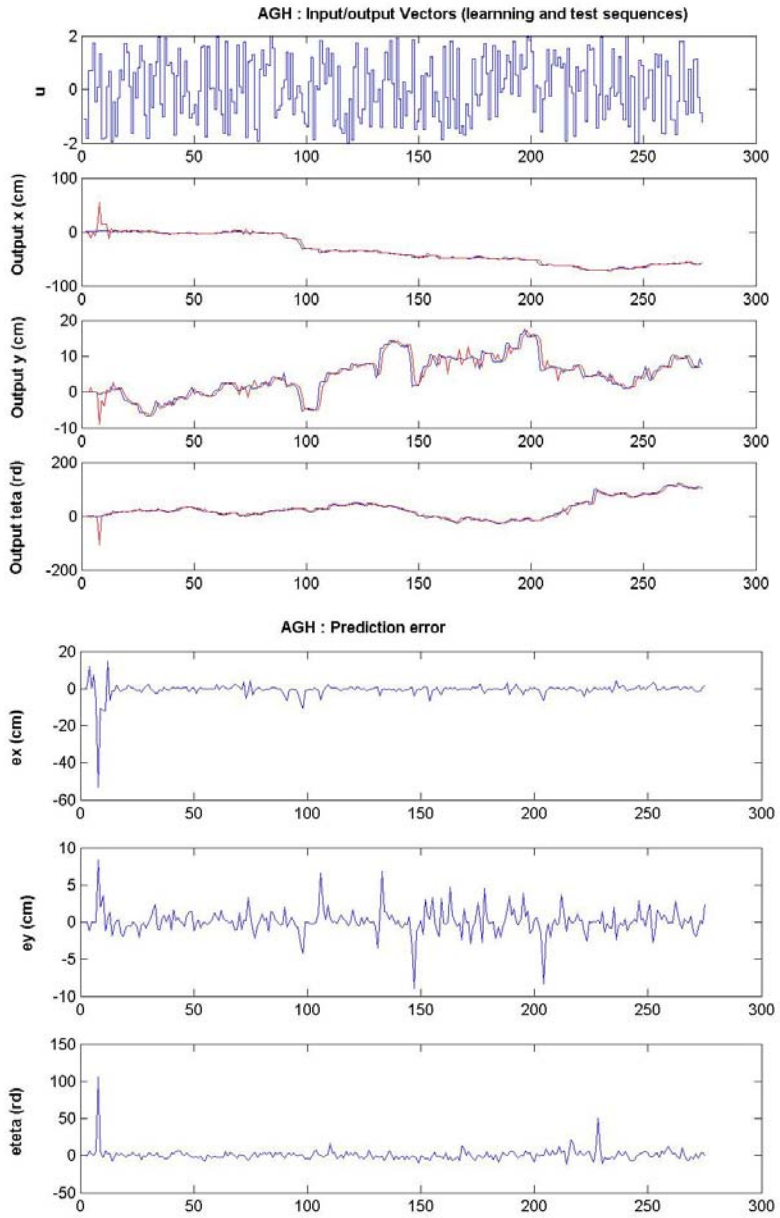


Figure 10. HGA: inputs/outputs sequences and prediction error

We note that the three outputs of the network follow the crowd of references. The neural direct model was very much improved; however, we note the presence of one or two gaps in the prediction error.

instruction from the desired displacement.

The principle of the model development is to provide for every step Δt the angle $d\theta$ (difference between reference orientation and

current orientation of the robot) and the distance r to be performed by the robot to reach the reference point of the trajectory, and to determine the control u to apply.

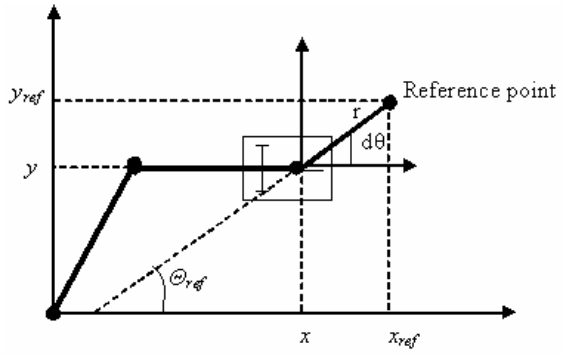


Figure 11. Moving the robot on the reference trajectory

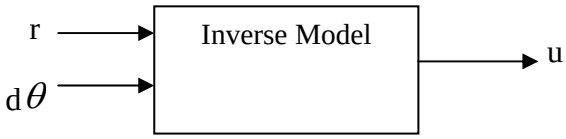


Figure 12. Inverse model

For the conception of the MNI, we will make the necessary choices including: for the regressor: [r (k) r (k-1) r (k-2) dθ (k) dθ (k-1) dθ (k-2) u(k-1) u (k-2)] and for the architecture, a multilayer MLP network.

Simulations

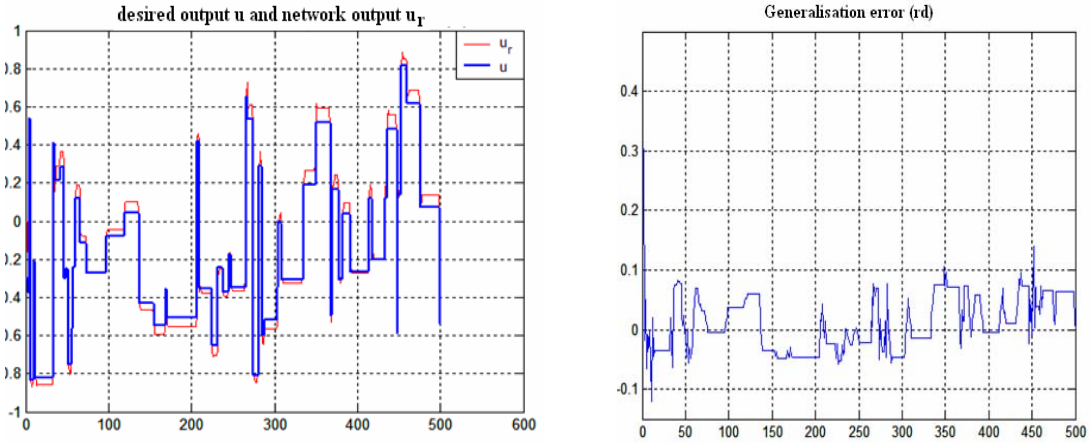
We have represented the results of simulation of the network output for the test sequence and the generalization error for a value of μ = 0.2 and a maximum number of iterations equal to 100. Figures 13a and 13b show respectively the simulation results and the variance of the error for a simple version of the gradient.

According to these figures, we note that the network output follows the desired output well with back propagation algorithm. It gives quite good results with an error variance of 0.033.

1.3 Internal model control for the wheelchair path tracking

Since 1982, the Internal Model Control constitutes a powerful strategy of control for complex systems, thanks to its simplicity of implementation, its robustness toward the errors of modeling and it facility of adjustments.

We apply, in what follows, the robot controller for the wheelchair trajectory tracking. We use for this the inverse and direct models generated previously. We use the direct model improved by HGA.



a) b)

Figure 13. Inverse Model Validation

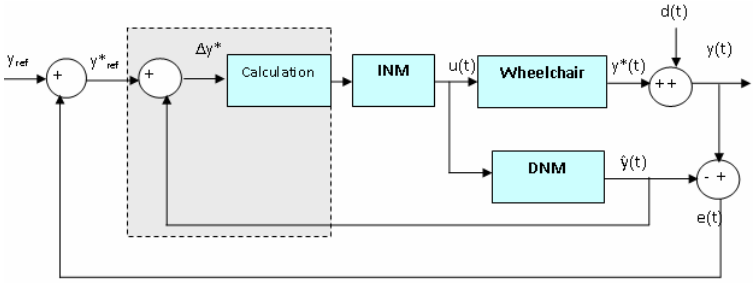


Figure 14. Structure for Internal Model Control

Internal Model Control is a structure that allows the error feedback to reflect the effect of disturbance and system mis-modeling.

So as not to clutter the diagram, we will present only one control loop on a single output y. See Figure 14.

The output y(t) is the output of the physical system, which is the sum of the ideal signal y*(t) and additive noise on its output d(t).

The dotted block calculates the entries from the INM reference to y^*_{ref} and the ideal output of the system represented by the output of DNM.

In our case, there is no noise output since it uses a simulation model, d(t) = 0, then we have:

e(t) = y(t) - y(t) = y*(t) - y(t).

The bloc Calculation of r and dθ made the following equation

$$r = \sqrt{\Delta x^2 + \Delta y^2}$$

Where $\Delta x = x_{ref} - x$ and $\Delta y = y_{ref} - y$

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

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

INM output is the control command u(t) which makes the movement of wheelchair.

Figure 15 shows the trajectory tracking for internal model control using the direct model established by HGA and the inverse one learned by backpropagation.

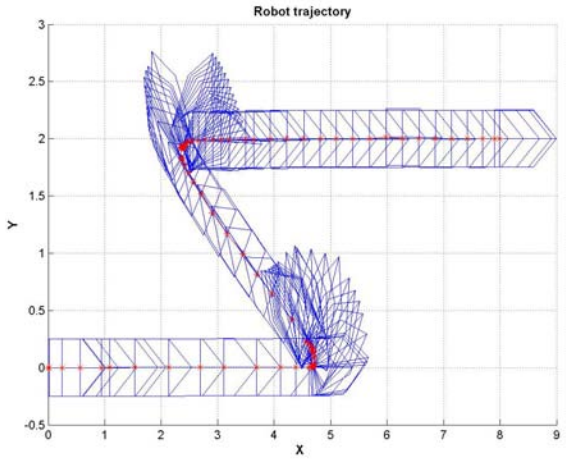


Figure 15. Tracking Z trajectory

A second simulation for a circular path has been made and shows that the robot reaches the trajectory and follows properly Figure 16.

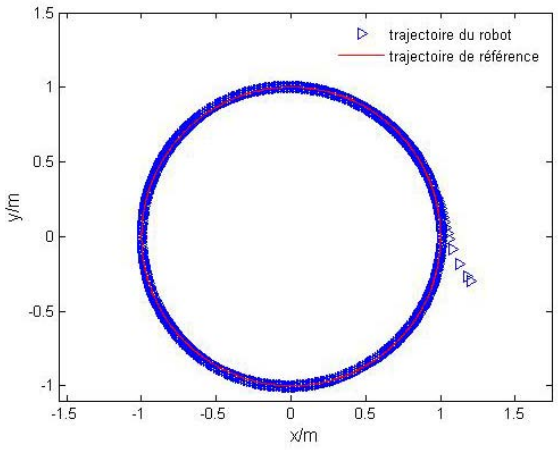


Figure 16. Tracking circular trajectory

We note that the trajectory followed by the wheelchair is perfect, showing that the wheelchair follows the form of the desired trajectories.

4. Conclusion

Usually, the modelling errors of direct model whose outputs are not ideal influence the outcome of the control. Indeed, for the direct model, we tried to improve learning and avoid local minimum. The evolutionary approach, which enables to improve both the structure and the weights of the network, allowed us to lead to very good results. As shown in Figure 4 and Figure 10, HGA decreased the prediction error, compared to that of backpropagation gradient. So a local minimum was avoided and the DNM improved.

So having established a “perfect” direct model with our approach HGA, we observe from Figure 15 that the trajectory tracking is very satisfying. It realizes a good tracking of the desired trajectory.

REFERENCES

1. ANGELINE, P. J., G. M. SAUNDERS, J. B. POLLACK, **An Evolutionary Algorithm that Constructs Recurrent Neural Networks**, IEEE Transactions on Neural Networks, Vol. 5, No. I, 1994.
2. IMEN, A., A. CHATTI, **Reactive Control Using Behavior Modelling of a Mobile Robot**, International Journal of Computers, Communications & Control, Vol. II, No. 3, 2007, pp 217-228.

3. HOLLAND, J., **Adaptation in Natural and Artificial Systems**. Ann Arbor: The University of Michigan Press, 1975.

4. ESPARCIA-ALCDZAR, A., K. SHARMAN, **Evolving Recurrent Neural Network Architectures by Genetic Programming**. In Genetic Programming 1997: Proceedings of the Second Annual Conference.

5. FATTOUH, A., Y. DADAM, D. PAHM, **Matlab Based 3D Dynamic Model of a Powered Wheelchair**, Innovative Production Machines and Systems Conference, 2008.

6. BONCI, A., S. LONGHI, A. MONTERIU, M. VACCARINI, **Navigation System for a Smart Wheelchair**, Journal of Zhejiang University SCIENCE, 2005.

7. KRISHNAN, R., V. B. CIESIELSKI, **2DELTA-GANN: A New Approach to Training Neural Networks Using Genetic Algorithms**. In Proceedings of the Fifth Australian Conference on Neural Networks, 1994.

8. MEYER, J. A., **Evolutionary Approaches to Neural Control in Mobile Robots, Systems, Man, and Cybernetics**, IEEE International Conference Vol. 3, 1998, pp. 2418-2423

9. GREFENSTETTE, J. J., **Evolutionary Algorithms in Robotics**. In International Symposium on Robotics and Manufacturing, New York, 1994.

10. MILLER, G. F., P. M. TODD, S. U. HEGDE, **Designing Neural Networks using Genetic Algorithms**. In Proceedings of the Third international Conference on Genetic Algorithms, 1989 (ICGA-89).

11. MONTANA, D. J., L. D. DAVIS, **Training Feedforward Networks using Genetic Algorithms**. In Proceedings of the international Joint Conference on Artificial intelligence, 1989 (IJCAI-89).

12. MORIARTY, D., R. MIIKKULAINEN, **Hierarchical Evolution of Neural Networks**. Technical Report A196-242, Department of Computer Sciences, The University of Texas, at Austin, Texas, USA, 1996.

13. PASSOLD, F., **Applying RBF Neural Nets for Position Control of an Inter/Scara Robot**, International Journal of Computers, Communications & Control, Vol. IV, No. 2, 2009, pp. 148-157.

14. NADI, A., S. S. TAYARANI-BATHAIE, R. SAFABAKHSH, **Evolution of Neural Network Architecture and Weights Using Mutation Based Genetic Algorithm**, Proceedings of the 14th International CSI Computer Conference (CSICC09),

15. BEN OMRANE, I., A. CHATTI, **Training a Neural Network Using Hierarchical Genetic Algorithm for Modeling and Controlling a Nonlinear System of Water Level Regulation**, Nonlinear Dynamics and Systems Theory, Vol. 10, N° 1, 2010, pp. 65-76.

16. OSORIO, R., J. A. ROMERO, M. PEÑA, I. LÓPEZ-JUÁREZ, **Intelligent Line Follower Mini-Robot System**, International Journal of Computers, Communications & Control Vol. I, No. 2, 2006, pp. 73-83.

17. CHATTI, A., I. AYARI, P. BORNE, M. BENREJEB, **On the Use of Neural Techniques for Path Following Control of a Car-like Mobile Robot**, Studies in Informatics and Control: Vol. 14, No. 4, 2005, pp. 221-234.

18. TANGOUR, F., P. BORNE, **Presentation of Some Metaheuristics for the Optimization of Complex Systems**, Studies in Informatics and Control : Vol. 17, No. 2, 2008, pp. 169-180.

19. ALASTY, A., H. N. PISHKENARI, S. H. MAHBOOBI, **Trajectory Tracking of a Mobile Robot Using Fuzzy Logic Tuned by Genetic Algorithm**, International Journal of Engineering Vol. 19, No. 1, November 2006, pp. 95-104.

20. BEL HADJ, S. A. NAOUI, D. JARBI, M. BENREJEB, **Neural Internal Model Control of a Mobile Robot**, Journal of Automation & Systems Engineering, Vol. 2, Issue 3, September 2008.