A Reactive Obstacle Avoidance Method Using Hierarchical Neural Network

Ma¨ıssa BOUJELBEN, Chokri REKIK and Nabil DERBEL
Control & Energy Managment Laboratory (CEM-Lab)
University of Sfax, Sfax Engineering School, BP W,1173-3038 Sfax, Tunisia.
e-mail: boujelben.maissa@gmail.com , chokri.rekik@enis.rnu.tn, nabil.derbel@ieee.org

***Abstract*—The aim of this paper is to control a mobile robot from an initial position to a desired one in an autonomous way. To ensure this objective, we propose a new approach based on neural networks. This approach adopts the principle of the potential field method: the robot is attracted to the target and pushed from obstacles. The control procedure is described as follows. A simple neural network is used to guide the robot to the target position when the environment is without obstacles. Since there are obstacles, a hierarchical neural network is proposed to guarantee obstacle avoidance. This constitute a solution for processing sensors information. Simulation results illustrate the validity and the efficiency of the suggested approach.**

***Index Terms*—Mobile robots, Hierarchical neural networks, Obstacle avoidance.**

I. Introduction

The interest of robotics researchers for improving the autonomy degree of their robotic system is growing increasingly. In fact, robots are designed to operate in different conditions (dangerous or hostile environment), without human intervention. In addition, many applications in the robotics field, require autonomous navigation such as service robots, supervision or exploration. In the literature, navigation is a very important issues for the successful use of an autonomous mobile robot [1]-[4]. When the environment is without obstacles, the problem becomes less complex to handle. Whereas, as the environments becomes complex, the resolution of this problem needs much more treatments to allow the robot to move safely. To ensure this autonomy, the use of control methods related to the artificial intelligence is a very required task. Indeed, intelligence offers the ability to reason, to analyze situations and to make decisions. Among these methods, we find for example genetic algorithm, fuzzy logic and neural networks. Several works have used neural networks to solve navigation problems. In fact, neural networks are considered as an effective control method due to their nonlinear approximation and learning abilities [5]-[7]. Two neural networks are proposed by Janglova´ [8] to construct a collision-free path for the robot motion. The first neural network is used to determine the free space and the second one is used to find a safe direction for the next robot step while avoiding obstacles. Fierro et al. [9] developed a controller based on artificial neural network by combining the feedback velocity control technique and torque controller. Mitic et al. [10] presented a solution for the visual control of a nonholonomic mobile robot using neural networks.

In this paper, we propose to use neural networks to control a mobile robot from an initial position to a desired one without colliding obstacles. To guide the robot to the target position, a simple neural network is used. For obstacles detection, we have to use the eight sensors of Khepera II robot [11] in order to locate obstacles from several directions. In this case, if we use only one neural network, we will have a high number of neural network input variables. So that, developing database samples become a very hard task. As a solution, we propose to use a hierarchical neural network. The idea is to decompose the neural network, that refer to the whole system, into several subnets of a smaller size (intended to model a set of input variables) and to relate them in a hierarchical way [12]. According to Mavrovouniotis [13], hierarchical neural networks are more accurate, faster to train and easier to analyze, compared to traditional neural networks.

This paper is organized as follows. In section II, we have presented some generalities about neural networks. Here, a description of the used neural network as well as the expressions of the weights adjustment are given. Then, we have introduced the problem formulation and we have detailed the interest of using the hierarchical structure. The proposed control algorithms allowing to generate the robot motion are given in section IV. Section V presents some simulation results which demonstrate the efficiency of the proposed algorithms.

II. Generalities about neural network

The first neural network, called the perceptron, was invented by Rosenblatt [14]. It was conceived for pattern recognition in a human visual system. However, the perceptron is a linear classifier which respond only with true or false for each input vector. In addition, only the weights of the decision layer can be modified. Therefore, several problems can not be solved using the perceptron [15]. The appearance of the Multi Layer Perceptron (MLP) helps to overcome the limitations of the perceptron. Figure 1 represents a MLP which is a feed forward neural network containing the following layers:

- The input layer: this layer is composed of neurons in which the input vector is applied.
- The output layer: the outputs of the neurons of this layer constitute the network outputs.

ISSN:2356-5608

- The hidden layer: layers between input and output layer are called hidden layer. We can find one or more hidden layers. In this work, we have adopted neural networks with one hidden layer. This reduces the computational complexity.

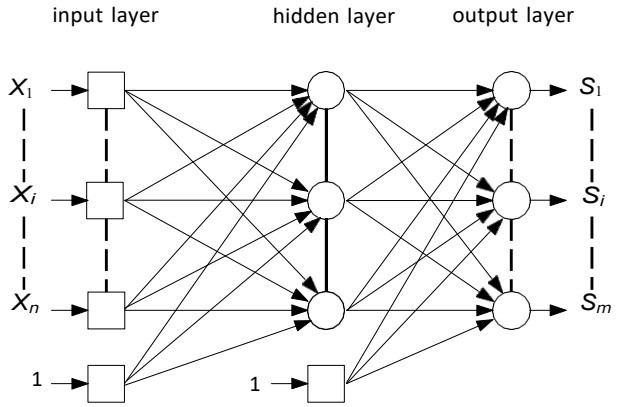

Fig. 1. Neural network topology

A MLP is well suited to approximate solutions for complex problems such as nonlinear function approximation, pattern recognition, classification problems, etc. This success is due to its high ability for learning. Indeed, learning is the process of finding weight values that minimize an error function. Back-propagation algorithm [16] is the most sophisticated learning algorithm for multi-layered networks. This algorithm offers the possibility of adapting all layers as it is an approximation of the gradient descent method. Firstly, a weighted summation is performed, at each layer, according to the previous layer signals. Then, the result is transmitted to the next layer until obtaining the current output signal. Finally, error signals are passed backward to adjust weights iteratively.

The error function, presented by equation(1), is defined as the sum of squared errors between the desired output and the network output.

$$E_p = \frac{1}{2}\|S^p - Y^p\|^2 = \frac{1}{2}\sum_{i=1}^{m}(s_i^p - y_i^p)^2 \quad (1)$$

with $Y=(y_1, ..., y_m)$ is the desired output value, $S=(s_1, ..., s_m)$ is the network output, p is the training example and m is the number of output neurons.

The error over the set of training data can be measured according to equation (2). This is to evaluate the training quality.

$$E = \frac{1}{M}\sum_{p=1}^{M} E_p = \frac{1}{2M}\sum_{p=1}^{M}\|S^p - Y^p\|^2 \quad (2)$$

with M is the samples number.

The weights adjustment is carried out according to equation (3). In this equation, we note by w_{uv} the weight between layers u and v. The first term of this equation is responsible to adapt weights proportionally to the error. The second one aims to remove oscillations arising from the choice of a high learning rate value. Indeed, a very low value of the learning rate may increase the convergence time. So, it is necessary to choose adequately the value of this parameter because it will have a big impact on the convergence of the algorithm.

$$\Delta w_{uv}(t) = -\varepsilon\frac{\partial E_p(w)}{\partial w_{uv}(t)} + \alpha\Delta w_{uv}(t-1) \quad (3)$$

where t is the current iterations and ε is the learning rate.

After some computations, equation (3) can be rewritten in two ways:

First case: when the neuron v belongs to the output layer, so:

$$\Delta w_{uv}(t) = -\varepsilon o_u(s_v^p - y_v^p)f'(I_v) + \alpha\Delta w_{uv}(t-1) \quad (4)$$

Second case: when the neuron v belongs to the hidden layer:

$$\Delta w_{uv}(t) = -\varepsilon o_u w_{vk}(s_k^p - y_k^p)f'(I_k)f'(I_v) + \alpha\Delta w_{uv}(t-1) \quad (5)$$

Here, I_v and O_v are used to express the input and the output of the neuron v respectively, k is the index of neurons that are connected to the neuron v and owned to the output layer and f is the sigmoid activation function.

III. Problem Formulation

Khepera II robot is a nonholonomic mobile robot having two independent driving wheels. It is controlled by acting on the speeds of these two wheels. Figure 2 represents the schematic model of this robot. The kinematic model of

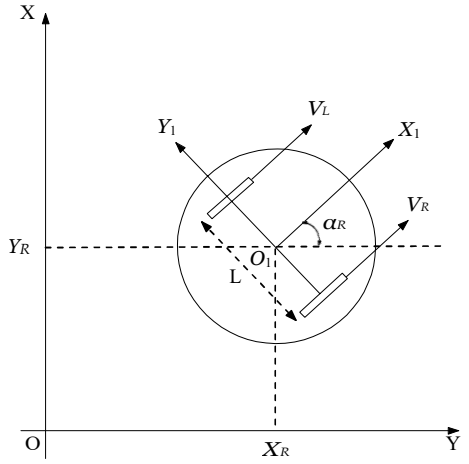

Fig. 2. Schematic model of the mobile robot Khepera II.

Khepera II robot is defined as follows:

$$\begin{cases} \frac{dX_R}{dt} = \frac{V_R + V_L}{2}\cos\alpha_R \\ \frac{dY_R}{dt} = \frac{V_R + V_L}{2}\sin\alpha_R \\ \frac{d\alpha_R}{dt} = \frac{V_R - V_L}{L} \end{cases} \quad (6)$$

where X_R and Y_R are the robot coordinates, α_R is the angle between the robot direction and the X-axis, V_R and V_L are

ISSN:2356-5608

respectively the robot right and left wheel velocities and L is the distance between two wheels.

The discretization of system (6), using Euler method, gives:

$$\begin{cases} X_{R,k+1} = X_{R,k} + T\dfrac{V_{R,k} + V_{L,k}}{2}\cos\alpha_{R,k} \\ Y_{R,k+1} = Y_{R,k} + T\dfrac{V_{R,k} + V_{L,k}}{2}\sin\alpha_{R,k} \\ \alpha_{R,k+1} = \alpha_{R,k} + T\dfrac{V_{R,k} - V_{L,k}}{L} \end{cases} \quad (7)$$

with T is the sampling time and k is the iteration step.

Khepera II robot is equipped with eight infrared sensors, distributed on its contour as shown in figure 3. These sensors allow to locate obstacles from several directions. Each sensor measures the distance d_i separating the sensor S_i and the obstacle, and the angle ϕ_i between the sensor orientation and the obstacle. Therefore, if we try to use only one neural network for the robot control, that take into account all sensor measurements, we will have a problem of dimensionality. Indeed, the neural network will have sixteen inputs, which are the distances d_i and the angles ϕ_i for $i = 1, .., 8$, and two outputs, which are the two wheel velocities V_R and V_L. Preparing database samples for this neural network is a very hard task. To simplify the analysis and the modeling of this neural network, we propose to design a hierarchical structure. In fact, hierarchical neural networks are composed of a number of loosely-coupled subnets, arranged in layers [13]. Each subnet is intended to model a particular aspect of input variables. For that, possible relationships between input variables should be determined based on system knowledge and desired behavior. A subnet which belongs to the lower layer of the hierarchical structure is related to a particular set of input variables. whereas, Subnets of intermediate layers receive their inputs from the previous layer and send their output to the next layer.

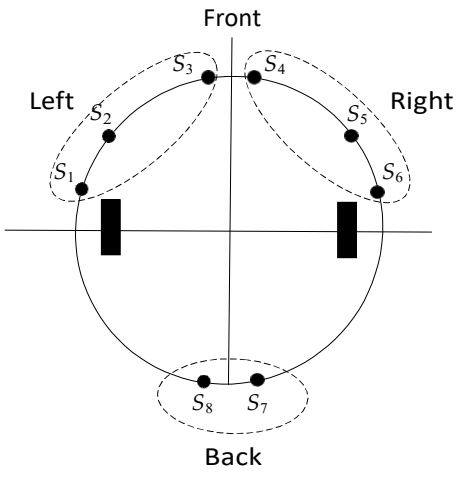

Fig. 3. Position of eight sensors of Khepara II robot.

The proposed hierarchical structure is presented in figure 4. The first layer of this structure is composed of eight neural networks which are related to the sensor measurements. The inputs of a network i are the distance d_i and the angle ϕ_i, for $i = 1, .., 8$. While, the outputs are the two wheel velocities V_{Ri} and V_{Li} responsible for obstacles avoidance and an index called I_0^i $(i = 1, .., 8)$ which gives information about obstacle position with respect to the robot. For this index, we have used three numerical values while preparing database samples, which are 0, 1 and 2. The value 0 indicates that the obstacle is far away from the robot, the value 1 indicates that the obstacle is relatively near to the robot, whereas the value 2 indicates that the obstacle is close to the robot [17]. Then, as it is shown on figure 3, we grouped sensor according to their positions and orientation in the robot. With this grouping, three neural networks are obtained, in the second layer of the hierarchical structure, enabling to locate obstacles on the three robot sides (left, right and back sides). Finally, the last neural network collects data provided by the previous layer about obstacle position, and performs two indices I_{tR} and I_{tL} allowing to find the target direction knowing obstacles positions. For these indices, we have also used three numerical values. But, in this case, the value 0 shows that the obstacle is close to the robot; The robot should then modify its direction. The value 1 is used when the obstacle is relatively near to the robot; the robot can advance with a low speed and modify its direction. The value 2 indicates that there is no obstacle; So, the robot can advance safely.

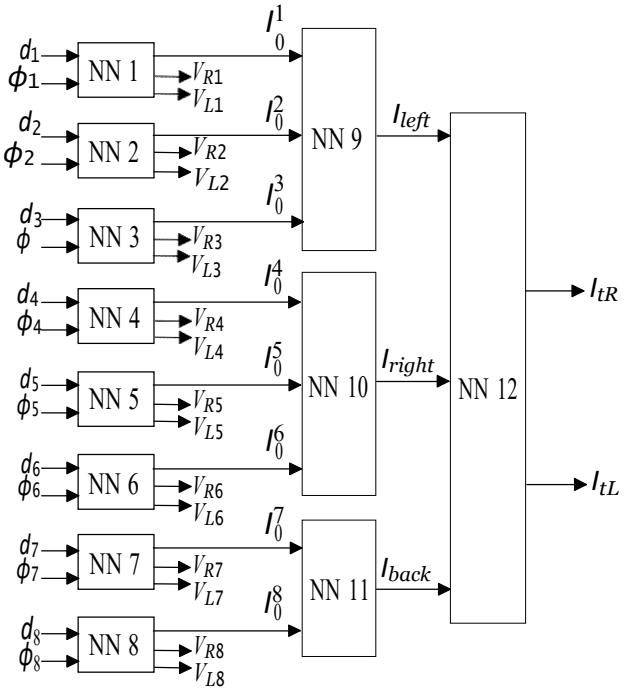

Fig. 4. The hierarchical neural network control structure.

ISSN:2356-5608

IV. CONTROL ALGORITHMS

Backpropagation algorithm consists of two phases: training phase and generalization phase.

A. The backpropagation training algorithm

The backpropagation training algorithm is an iterative gradient algorithm [18] designed to search for optimal weight values that minimize the squared error between the desired output and the current network output. This algorithm can be stated as follows::

Step 1 : Initialize the network weights w_{uv}.
Step 2 : Present the training example p (the input data $X=(X_1, ..., X_n)$ and the desired output Y).
Step 3 : Compute the network output S.
Step 4 : Find the change of weights Δw_{uv} according to equations (4) and (5).
Step 5 : Update weights as:
$w_{uv}(t+1) = \Delta w_{uv}(t) + w_{uv}(t)$
Step 6 : Repeat steps 2 to 4 until the error E_p is below a prespecified threshold.

For the robot control, two situations should be distinguished. The first situation presents the case of an environment without obstacles. In this case, a neural network having two inputs|two outputs (2I|2O) is trained. This network is responsible for controlling the robot towards the target. Therefore, the network inputs are the distance d_t between the robot position and the target, and the angle ϕ_t between the robot orientation and the target. The outputs of this network are the left and right wheels velocities V_{Rt} and V_{Lt} allowing the robot motion to the target position. The second situation deal with obstacles avoidance problem. To solve this problem, we propose to use the proposed hierarchical structure (figure 4). For that, the four neural networks, constituting this structure, should be trained. These networks are the following:

1) a 2I|3O neural network (used in the first layer of the hierarchical structure). The inputs are the distance d_i and the angle ϕ_i provided by each sensor ($i = 1, .., 8$). While, the outputs are the left and right wheels velocities (V_{Ri} and V_{Li}) and the index I^i_0 used to analyze the obstacle position compared to the robot.
2) a 3I|1O neural network (used in the second layer of the hierarchical structure). This network uses three indices I^i_0 as inputs and provides the index I_{left} (for $i = 1, 2, 3$) or I_{right} (for $i = 4, 5, 6$), ie. depending on sensors position in the robot.
3) a 2I|1O neural network (used also in the second layer of the hierarchical structure). This network is related to the information supplied by the back positioned sensors. So, it has as inputs the indices $I^i_0 (i = 7, 8)$ and as output the index I_{back}.
4) a 3I|2O neural network (used in the third layer of the hierarchical structure). The indices I_{left}, I_{right} and I_{back} are considered the network inputs. They give information about obstacles location in the environment. The network outputs are, in this case, the indices I_{tR} and I_{tL} to find the target direction knowing obstacles position.

B. Generalization

The generalization phase is conducted to demonstrate the ability of a neural network to respond adequately on patterns that are not trained during learning. In this phase, given an input vector X, a forward propagation of data is performed (using the optimal weight values).

In this work, the actions ensuring the robot motion are produced in this phase. So, the robot control algorithm can be summarized as follows:

1) Specify the start and goal configurations.
2) Compute the distances d_t and d_i and the angles ϕ_t and ϕ_i, $i = 1, .., 8$.
3) Activate the neural network related to the target to find V_{Rt} and V_{Lt}.
4) - $\forall\ i = 1, .., 8$, if $d_i > 50$ mm (sensors scope), then the robot right and left wheel velocities responsible for controlling the robot are given by:

$$\begin{aligned} V_R &= V_{Rt} \\ V_L &= V_{Lt} \end{aligned} \tag{8}$$

- Else, execute neural networks (using the optimal weight values) according to the proposed hierarchical structure, and compute V_R and V_L as follows:

$$\begin{aligned} V_R &= I_{tR} V_{tR} + \sum_{i=1}^{n} I^i_0 V_{Li} \\ V_L &= I_{tL} V_{tL} + \sum_{i=1}^{n} I^i_0 V_{Ri} \end{aligned} \tag{9}$$

where n is the sensor number.

5) Update the robot position and orientation according to system (7).
6) Return to step 2 until the robot reaches the target position.

V. SIMULATION RESULTS

Firstly, it is noted that the different neural networks are off-line trained. Then, given the robot start position, its initial orientation and the target position, the robot should takes on-line decisions in order to reach this target.

This section is devoted to evaluate the performance of the proposed approach. For that, and without loss of generality, we suppose that the robot starts from the position $(X_{R_0}, Y_{R_0}) = (0, 0)$ and should reach a target located at the position $(X_t, Y_t) = (300, 300)$. Figure 5 presents some simulation results for different environments. This figure shows that, the robot responds well to its environment changes. In fact, regardless obstacles positions, the robot is able to reach its goal safely, which prove the reactivity of the suggested approach. In addition, the generated trajectories are reasonably smooth.

ISSN:2356-5608

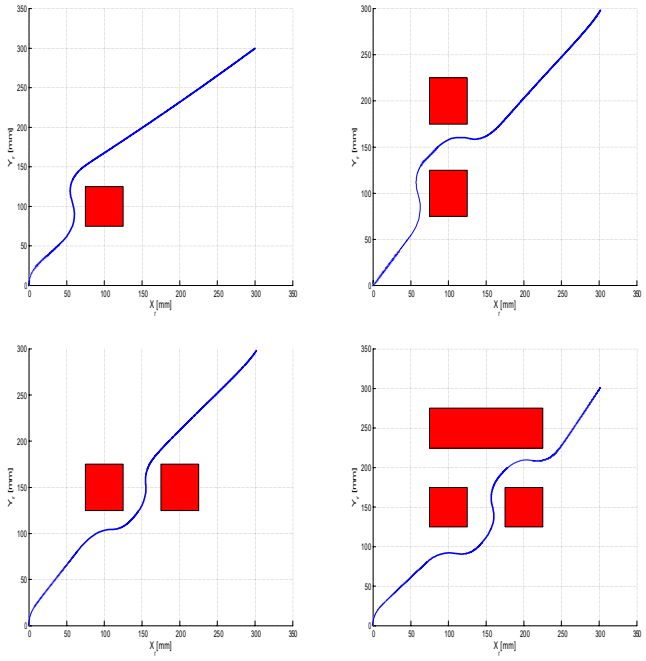

Fig. 5. Simulation results for different configurations of obstacles.

Comparing this method with the one which uses hierarchical fuzzy systems [12], we find that both methods are reactive and enable a safe and smooth navigation, which is the major issue of this work. Nevertheless, neural networks training phase requires an important computation time.

VI. CONCLUSION

This paper deal with autonomous navigation problem. The objective is to guide a mobile robot from an initial position to a desired one without colliding obstacles. For that, we present a control approach based on hierarchical neural networks. The idea of this approach is to decompose the neural network, that refer to the whole system, into several subnets of a smaller sizes (which are intended to model a particular set of input variables). These subnets are related in a hierarchical way. This is to overcome the problem of large number of input variables. Simulation results demonstrate the reactivity of the proposed control algorithm. The obtained trajectories are safe and smooth. However, the development of neural network algorithms, especially for learning, requires an important computation time.

REFERENCES

[1] W. L. Xu, S. K. Tso,"Sensor-based fuzzy reactive navigation of a mobile robot through local target switching", IEEE Trans. on Systems, Man, and Cybernetics, 29(3):451–459, (1999).

[2] A. Zhu and S.X. Yang, "A fuzzy logic approach to reactive navigation of behavior-based mobile robots", IEEE International Conference on Robotics and Automation, 5:5045–5050, (2004).

[3] M. M. Joshi, M. A. Zaveri, "Reactive Navigation of Autonomous Mobile Robot Using NeuroFuzzy System", Int. J. of Robotics and Automation, 2(3):128–145 , (2011).

[4] M. Boujelben, C. Rekik and N. Derbel, "Hierarchical fuzzy controller to avoid mobile obstacle for a mobile robot", Int. Multi-Conf. on Systems, Signals & Devices (SSD), Hammamet, Tunisia, (2013).

[5] C.-W. Chen, "Stability analysis and robustness design of nonlinear systems: an NN-based approach", Appl. Soft Comput., 11(2):2735–2742, (2011).

[6] C.-F. Hsu, C.-M. Lin, R.-G. Yeh, "Supervisory adaptive dynamic RBF-based neural-fuzzy control system design for unknown nonlinear systems", Appl. Softw. Comput., 13(4):1620–1626, (2013).

[7] J. J. Rubio, "Modified optimal control with a back propagation network for robotic arms", IET Control Theory and Appl., 6(14):2216–2225, (2012).

[8] D. Janglová, "Neural Networks in Mobile Robot Motion", Int. Journal of Advanced Robotic Systems, 1(1):15–22, (2004).

[9] R. Fierro, F.L. Lewis, "Control of a nonholonomic mobile robot using neural networks", IEEE Trans. on Neural Networks, 9(4) : 589–600, (1998).

[10] M. Mitić, Z. Miljković, "Neural network learning from demonstration and epipolar geometry for visual control of a nonholonomic mobile robot", Soft Computing, (2013).

[11] K-Team S. A., Khepera II User Manual, Switzerland, (2002).

[12] M. Boujelben, C. Rekik and N. Derbel, "Hierarchical fuzzy controller for a nonholonomic mobile robot", 20th Mediterranean Conf. on Control & Automation (MED), Barcelona, Spain, (2012).

[13] M. L. Mavrovouniotis, "Hierarchical neural networks", National Science Foundation, Program (NSFD CD 8803012), Industry and the University. TR 91–4, (1991).

[14] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain", Psychological Review, 65:386–408, (1958).

[15] M. Minsky and S. Papert, "Perceptrons: an introduction to computational geometry", MIT Press, (1969).

[16] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning Internal Representations by Error Propagation", Parallel Distributed Processing: Explorations in the Microstructures of cognition, 1:318–362, (1986).

[17] M. Boujelben, C. Rekik and N. Derbel, "A multi-agent architecture with hierarchical fuzzy controller for a mobile robot", Int. J. of Robotics and Automation, 30(3), (2015).

[18] R.P. Lipmann, "An introduction to computing with neural nets", IEEE Acoust. on Speech, Signal Processing (ASSP Magazine), 688–695, (1987).

ISSN:2356-5608