ANFIS Controller for Non-holonomic Robots

Ting Wang[1], Christophe Sabourin[2], Kurosh Madani[3]
1, 2, 3- Signals, Images, and Intelligent Systems Laboratory (LISSI / EA 3956), Universit´e Paris − Est, S´enart Institute of Technology, Avenue Pierre Point, 77127 Lieusaint, France.
Email[1]: wangting0310@yahoo.com.cn (Corresponding author)
Email[2]: sabourin@u-pec.fr
Email[3]: madani@u-pec.fr

Received: October 2010 Revised: January 2011 Accepted: May 2011

ABSTRACT:
In this paper, a control strategy for a non-holonomic robot based on an Adaptive Neural Fuzzy Inference System is proposed. The neuro-controller makes it possible for the robot to track a given reference trajectory. After a short introduction about Adaptive Neural Fuzzy Inference System, the control strategy which is used on our virtual non-holonomic robot is described. And finally, the simulations' results where the robot has to pass into a narrow path and also the first validation results concerning the implementation of the proposed concepts on a real robot is given.

KEYWORDS: ANFIS controller, nonholonomic, multi-robot .

1. INTRODUCTION

Research about the multi-robot systems have started in the late 1980s, for instance, the project CEBOT (Fukuda, 1998). Indeed, the multi-robot systems offers many advantages in comparison with systems using only one robot (Parker, 2008) (Cao, 1997):

- In first, cooperation between a group of several robots can carried out more complex tasks,
- Secondly, the use of several robots for a given task allows to increase robustness,
- And finally, the design and the use of several simple robots can be cheaper and more flexible.

Today, and in the future, many applications can benefit of advantages of multiple robot systems like, for instance, in the warehouse management, for the industrial assembling, in military applications, or for daily tasks, so on. However generally, the design of one control strategy for systems with several robots requires cooperation and coordination between all robots. This means that robots can communicate between them and self-organize in the group. With the new recent technologies like wireless communication, one robot can easily send information to another robot. Consequently, in the future works, the main challenge will focus on the design of control strategies allowing to a group of robots to self-organize with, if possible, emergent behaviors. In this context, the goal of our laboratory is to design control strategies for multi-robot systems. However, one major problem about the control of a multi-robot system is coordination and formation control, and namely, the design of control strategy making it possible for a wheeled robot to track a desired trajectory. And generally, the wheeled robots are nonholonomic robots increasing the difficulty to design the control strategy.

Most of the control approaches are based on asymptotic stabilization with the feedback controls. Different methods have been used to reduce or to transform the nonlinear kinematic equation into a linear approximation system. For instance, Samson (1995) transformed the nonlinear system into a chained system with the feedback control to solve the path-following problem. Several authors have addressed the problem of tracking admissible trajectory by applying dynamic feedback linearization techniques (Kolmanovsky, 1995), (D'Andrea-Novel, 1995), (De Luca, 1993),(Fliess, 1995).

In Morin (2003), the authors are certainly the first to address the problem of tracking arbitrary trajectories (i.e., not necessarily feasible for the controlled robot) based on the conception of transverse functions. And in Barfoot (2004) , the feedback control law inherits the strong robustness properties associated with stable linear systems, but it yields slow convergence. In this short overview about control strategies for nonholonomic robots, all approaches are based on a kinematic modeling and most of them have a slow convergence. The main drawback of this is the control strategy must be failed in some cases. An alternative solution to the kinematic modeling is to use neural networks.

In this paper, we propose a new approach to control nonholonomic robot based on Adaptive Neural Fuzzy

Inference System (ANFIS). This approach may be decomposed in two part: the first one allows to decompose an arbitrary path into several desired trajectories, and the second is composed of two neuro-controller, both position and orientation control, allowing to track these desired trajectories. In fact, ANFIS control don't depend on kinematic equations, and although, we present the control strategy for nonholomic robot, this concept may be used on another kind of wheeled robots.

The paper is organized in the following way. In the next section, we introduce the ANFIS Adaptive Neural Fuzzy Inference System. In the third section, we give the kinematic model of the nonholonomic robot, and we describe how we can control the wheeled robot with ANFIS. In section 4, we present the control strategy. Simulations' and validation's results have been shown in the fifth section. And finally, we give the simulations' results where the robot have to pass into a narrow path as well as the first validation results concerning the implementation of the proposed concepts on real robot. At last, we get some brief conclusions.

2. ANFIS NEURAL NETWORK

The main advantage of a Fuzzy Inference Systems is that it allows to deal some systems, where it is difficult to design control strategy based on mathematical modeling such as nonholonomic systems, because they are a non-linear systems. However, the main disadvantage of fuzzy system is that it needs a knowledge of an expert and needs a long time to get the accurate membership functions. Neural network, or more generally adaptive systems based on learning process (i.e. Q-learning, genetic algorithm, so on), can make up for this disadvantage and improve the basic fuzzy system. For instance ANFIS, which is based on both neural networks and fuzzy inference systems, is a class of adaptive fuzzy inference system. In this section, we remember briefly the ANFIS architecture initially proposed by Jang (1995). Assume that a control system with m inputs $x_1, x_2, \ldots, x_m$ and one output y, the n linguistic rules R_i can be expressed as:

$$\boldsymbol{If}\ x_1\ \boldsymbol{is}\ A_{i1}\ \boldsymbol{and}\ x_2\ \boldsymbol{is}\ A_{i2} \ldots\ldots\ldots \boldsymbol{and}\ x_m\ \boldsymbol{is}\ A_{im} \quad (1)$$
$$\boldsymbol{Then}\ y\ \boldsymbol{Is}\ w_i \quad i = 1, \ldots\ldots, n$$

where i is the index of the rule, A_{ij} is a fuzzy set for i-th rule and j-th input and w_i is a real number that represents a consequent part. In the present case, the membership function is defined as a gaussian function:

$$\mu_{ij} = \exp \frac{-\left(x_j - a_{ij}\right)^2}{2b_{ij}^2} . \quad (2)$$

The output of this neural network is given by the following equation:

$$y = \frac{\sum_{i=1}^{n} u_i\, w_i}{\sum_{i=1}^{n} u_i} \quad (3)$$

where u_i is given by:

$$u_i = \mu_{i1}\, \mu_{i2} \ldots\ldots\ldots\ldots \mu_{im} \quad (4)$$

Now, we define z the set of all parameters to adapt in the neural network:

$$z = a_{11}, \ldots, a_{nm}, b_{11}, \ldots, b_{nm}, w_1, \ldots, n_n \quad (5)$$

And V(z) the function to minimize :

$$V(z) = \frac{1}{2}(y(t) - y^d(t))^2 \quad (6)$$

where $y(t)$ is the output of the neural network and $y^d(t)$ is the desired output. In this case, Godjevac (1995) shown it was possible to use an iterative procedure to update parameters in order to minimize the function $V(z)$. The three kinds of parameter a_{ij}, b_{ij} and w_i may be updated by Eqs. 7,8 and 9 respectively.

$$a_{ij}(t+1) = a_{ij}(t) - \Gamma_a \times$$
$$\frac{u_i}{\sum_{k=1}^{n} u_k}(y - y^d)(\omega_i - y)\ \frac{\left(x_j - a_{ij}(t)\right)}{b_{ij}(t)^2} \quad (7)$$

$$b_{ij}(t+1) = b_{ij}(t) - \Gamma_b \times$$
$$\frac{u_i}{\sum_{k=1}^{n} u_k}(y - y^d)(\omega_i - y)\frac{\left(x_j - a_{ij}(t)\right)^2}{b_{ij}(t)^3} \quad (8)$$

$$\omega_i(t+1) = \omega_i(t) - \Gamma_\omega \frac{u_i}{\sum_{k=1}^{n} u_k}(y - y^d) \quad (9)$$

where a_{ij}, b_{ij} and ω_i are the parameters of the adaptation of the learning algorithm. Γ_a, Γ_b and Γ_ω are the predefined constants.

3. CONTROL NONHOLONOMIC ROBOT

Generally, the control of wheeled robots consists in doing a follow of reference path and supposes to measure both the position and orientation with respect to a fixed frame. Let us consider a given trajectory C in the reference frame, and a point P attached to the robot chassis, at the mid-distance of the wheels, as illustrated on figure 1. The state of the robot can be described by a triplet as $P(x, y, \theta)$, in which x and y are the coordinates of the robot in the reference frame. θ is the angle from *X-axis* to the robot's motion direction.

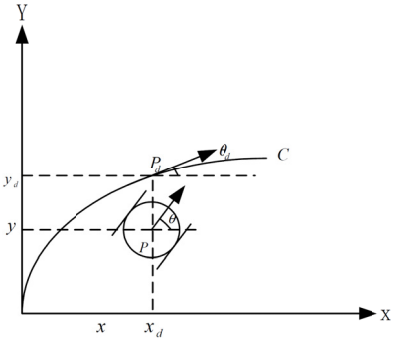

Fig. 1. Robot's coordinates described by a triplet as $\boldsymbol{P(x, y, \theta)}$.

The kinematic modeling of this wheeled robot (i.e. unicycle-type mobile robot) may be represented by Eqs. 10 and 11 (Pascal, 2008):

$$\begin{cases} V_x = V\cos\ \theta \\ V_y = V\sin\ \theta \\ \dot{\theta} = \Omega \end{cases} \quad (10)$$

$$\begin{cases} V = \dfrac{r}{2}(\Omega^{rigth} + \Omega^{left}) \\ \Omega = \dfrac{r}{2l}(\Omega^{rigth} - \Omega^{left}) \end{cases} \quad (11)$$

Where V_x and V_y represent respectively the instantaneous horizontal and vertical velocities of the point P located at mid-distance of the actuated wheels.

V represents the intensity of the longitudinal velocity and Ω the angular velocity of the robot. Ω^{left} and Ω^{rigth} are the angular velocity of the left and right wheels, respectively. r is the radius of the wheels and l is the distance between the two wheels.

For an unicycle-type mobile robot, the goal of the control strategy is to compute the velocities of each wheel in order to the robot follows the desired path. The given trajectory can be expressed as a function of time $P^d(x^d(t), y^d(t), \theta^d(t))$, with the $\theta^d(t)$ represents of the trajectory's curvature at each step time t. But, in the case of non-holonomic robots, where the kinematic model is represented by Eqs. 10 and 11, this control is not a trivial problem.

In this paper, we propose a new approach based on neural networks. The goal of these neural networks are to control the velocity of each wheel in order to minimize both error between position and desired position $(x - x_d, y - y_d)$, and orientation and desired orientation $(\theta_d - \theta_d)$.

3.1. Orientation control

The orientation control allows to the robot to rotate on itself in following the target angle. Consequently, the ANFIS needs one input x^θ which is the difference between of the angle between the robot's direction θ and the desired angle θ^d (see Eq. 12), and one output, which is an angular velocity.

$$x_\theta(t) = \theta(t) - \theta^d(t) \quad (12)$$

$$y_\theta(t) = \frac{\sum_{i=1}^{n} u_i^\theta\, w_i^\theta}{\sum_{i=1}^{n} u_i^\theta} \quad (13)$$

The relation between $y_\theta(t)$ and $\Delta\Omega$ (the difference between the right $\Omega_\theta^{\text{rigth}}$ and left $\Omega_\theta^{\text{left}}$ angular velocity) is given by the following equation:

$$\Delta\Omega(t) = \Omega_\theta^{rigth}(t) - \Omega_\theta^{left}(t) = y_\theta(t) \quad (14)$$

A each step time, the parameters w_i^θ are updated in order to minimize the following equation:

$$V_\theta(t) = (\theta(t) - \theta^d(t))^2 \quad (15)$$

3.2. Position control

The position control allows to the robot to follow the target point $(x_d(t), y_d(t))$ on a desired path. In this case, the neural network needs two inputs x_{px} and x_{py} which are given by Eqs. 16 and 17, respectively:

$$x_{px}(t) = x(t) - x^d(t) \quad (16)$$

$$x_{py}(t) = y(t) - y^d(t) \quad (17)$$

Where $x(t)$ and $y(t)$ correspond to the coordinates of the robot, and $x_d(t)$ and $y_d(t)$ correspond to the desired coordinates of the robot. The neural network have only one output $y_p(t)$:

$$y_p(t) = \frac{\sum_{i=1}^{n} u_i^p\, w_i^p}{\sum_{i=1}^{n} u_i^p} \quad (18)$$

And the relation between $y_p(t)$ and the right $\Omega_{\text{p}}^{\text{rigth}}$ and left $\Omega_{\text{p}}^{\text{left}}$ angular velocity is given by the following equation:

$$\Omega_p^{rigth} = \Omega_p^{left} = y_p(t) \quad (19)$$

4. CONTROL STRATEGY

In order to explain clearly the proposed approach, we present a practical example where the robot must move from an initial position to goal position by passing a narrow path (figure 2). This approach may be decomposed in two part: the first one allows to decompose the path into several desired trajectories (section 4.1), and the second is composed of two neuro-controller, both position and orientation control, allowing to track these desired trajectories (section 4.2).

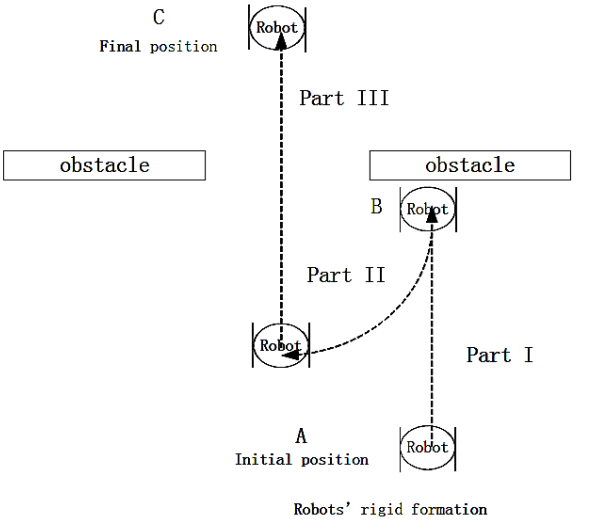

Fig. 2. Description of the path of the robot from point A to point C.

4.1. Desired trajectory

The figure 2 shows the trajectory of the robot from the initial position to the final position. The proposed example may be decomposed in three parts: firstly, the robot moves from the point A toward the obstacles.

Secondly, the robot follows a circle trajectory, and finally, the robot goes towards the final position. During these three parts, the desired trajectory $P^d(x^d(t), y^d(t), \theta^d(t))$, are computed as follow:

- During the first part, the robot moves from initial position A to the obstacle with position's control only. In this part, robot follows the vertical line $x^d = 0.3$ without the orientation control (see Eq. 20).

$$\mathrm{P^d(t)} = \begin{cases} \mathrm{x^d(t) = 0.3} \\ \mathrm{y^d(t) = y^d(t-1) + \Delta y} \\ \mathrm{\theta^d = 0} \end{cases} \quad (20)$$

$x^d(t)$ and $y^d(t)$ represent the coordinates of the robot according to the reference frame. $\theta^d = -180°$ is the orientation of the robot. Δy is chosen according to both length L and duration T of the path.

- During the second part, firstly, the robot turns around itself from $\theta^d = -180°$ to $\theta^d = 0°$ by using the orientation control, and secondly, the robot uses trajectory control to follow a circular arc (see Eq. 2).

$$P^d(t) = \begin{cases} x^d(t) = 0.3 * cos(\theta(t)) \\ y^d(t) = -0.3 * sin(\theta(t)) \\ \theta^d(t) = \theta^d(t-1) + \Delta\theta \end{cases} \quad (21)$$

Finally, the robot turns around itself from $\theta^d = 90°$ to $\theta^d = 0°$.

- During the final part, the robot follows a vertical line ($x^d = 0.0$) and goes in a narrow path to arrive at the final position C.

4.2. Trajectory control

If we combine orientation's control with position's control, we get trajectory's control which can make robot to follow a desired trajectory. In this case, the angular velocity of two wheels (Ω^{rigth} and Ω^{left}) are given by Eq. 3. $\Omega_{\mathrm{p}}^{\mathrm{rigth}}$ and $\Omega_{\mathrm{p}}^{\mathrm{left}}$ are given by the ANFIS position control, and $\Delta\Omega$ is given by ANFIS orientation control.

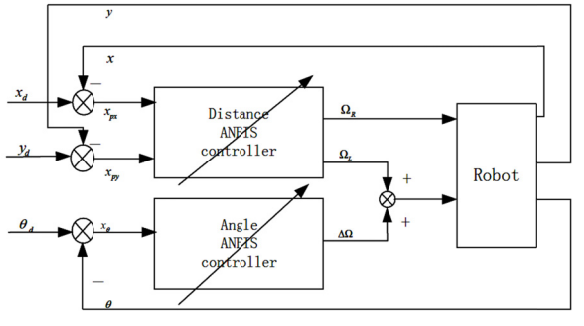

Fig. 3. Two neuro-controllers are used for both position's control and orientation's control.

Inputs e_x, e_y, and e_θ are the differences between the real position of the robot given by $P = (x, y, \theta)$, and the desired position $P^d = (x^d, y^d, \theta^d)$.

$$\begin{matrix} \Omega^{\mathrm{rigth}} = \Omega_{\mathrm{p}}^{\mathrm{rigth}} \\ \Omega^{\mathrm{left}} = \Omega_{\mathrm{p}}^{\mathrm{left}} + \Delta\Omega \end{matrix} . \quad (22)$$

5. RESULTS

In this section, first we present results of simulation for the problem described in the section 5, and secondly the first results obtained on the real robot kheperaIII.

5.1. Simulation

Simulation have been performed by using software Webots [1] with the virtual robot KheperaIII. The controller have been designed with the software Matlab [2]. Figures 4 and 5 show the trajectory and orientation of the robot during the simulation, respectively. On both figures, the red line represents the desired trajectories and the blue dot line the real position of the robot. On the figure 5, the axis t represents the time step. Figures 6, 7, 8 and 9 show a snapshot of this simulation.

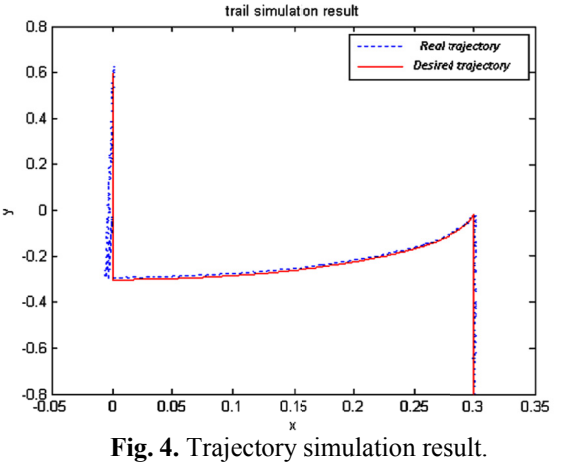

Fig. 4. Trajectory simulation result.

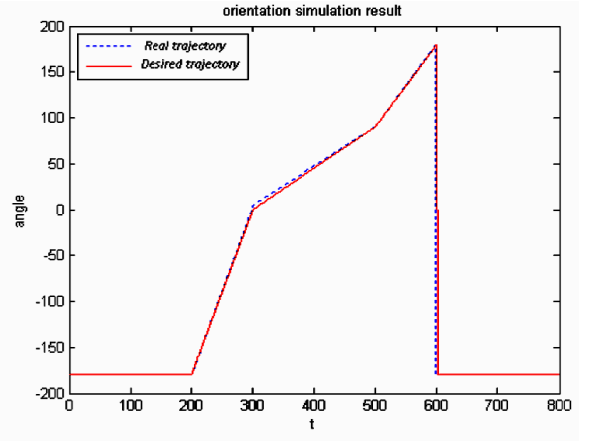

Fig. 5. Orientation simulation result.

[1] www.cyberbotics.com
[2] www.mathworks.com

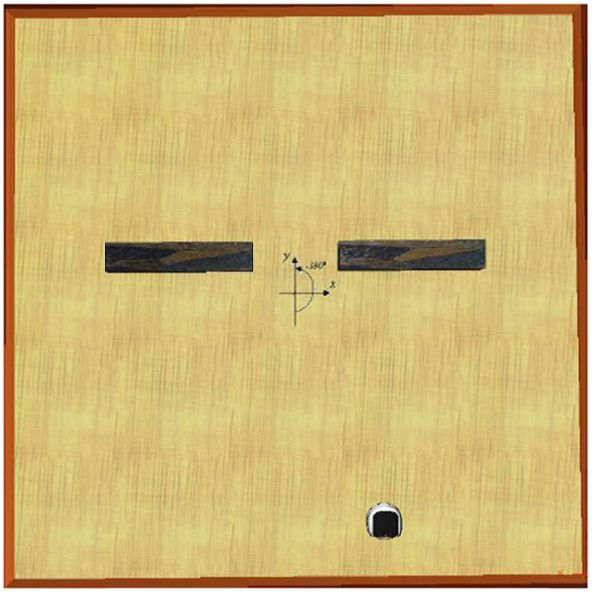

Fig. 6.. Position of the robot in the initial position $P_1^d = (0.3, -0.8, -180°)$.

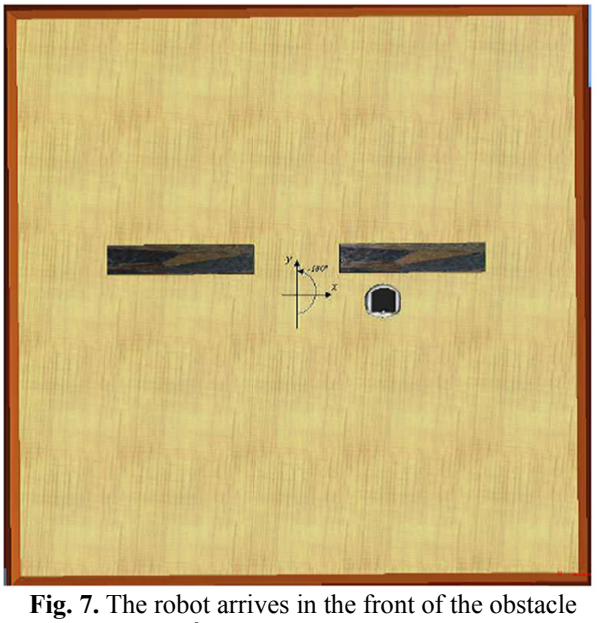

Fig. 7. The robot arrives in the front of the obstacle $P_2^d = (0.3, 0.0, -180°)$.

The path of the robot can be interpreted as follow:

- From $t = 0$ (figure 6) to $t = 200$ (figure 7), the robot follows a vertical line and moves from the point $(x = 0.3, y = -0.8)$ to point $(x = 0.3, y = 0)$. The desired angle is equal to $-180°$ ($\theta^d = -180°$).
- At $t = 200$ (figure 7), the robot turns on itself during 100 step time. During this stage, the robot stays at the point $(x = 0.3, y = 0)$ but, turns from $\theta = -180°$ to $\theta = 0°$
- From $t = 300$ to $t = 500$ (figure 8), the robot follows a desired circular trajectory (see section 4.1) and moves progressively from the point $(x = 0.3, y = 0, \theta = 0°)$ to $(x = 0, y = -0.3, \theta = 90°)$
- At $t = 200$, the robot turns on itself during 100 step time. During this stage, the robot stays at the point $(x = 0, y = -0.3)$ but turns from $\theta = 90°$ to $\theta = 180° = 0°$
- Finally, from $t = 600$ to $t = 800$ (figure 9), the robot follows a vertical line and moves to goal position $(x = 0, y = 0.6)$

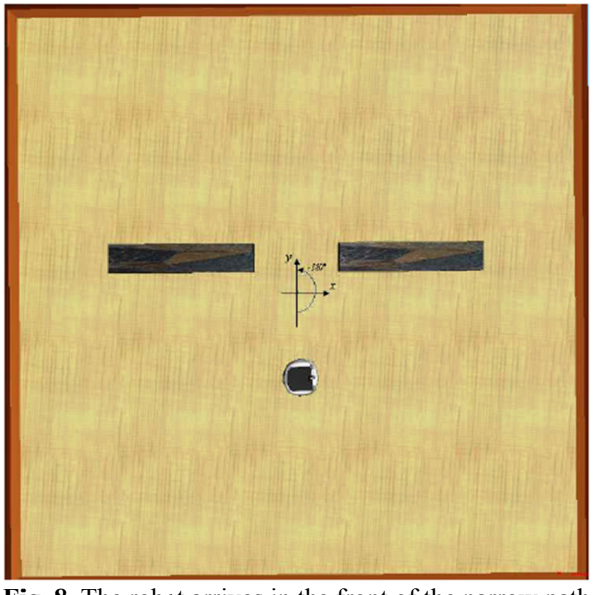

Fig. 8. The robot arrives in the front of the narrow path $P_3^d = (0.0, -3.0, 90°)$.

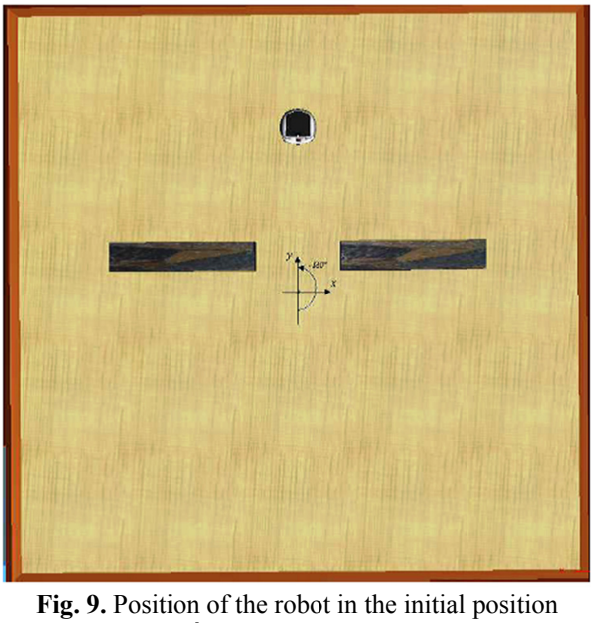

Fig. 9. Position of the robot in the initial position $P_4^d = (0.0, 0.6, -180°)$.

5.2. Experimental validation

The used robot is the robot kheperaIII with the korebotLE (see http://www.k−team.com/).

The robot Khepera III is equipped with two motors associated with incremental encoders, 9 infrared sensors and five ultrasonic sensors . A dsPIC 30F5011 microprocessor allows to manage all devices of the robot through a I2C communication. In addition, this robot offers the possibility to connect a KoreBot board allowing to increase the computational abilities. The main component of the KoreBot board is an Intel PXA255 XScale processor running at 400 MHz with 60 MB RAM and 32 MB flash memory. And, when the KoreBot board is mounted on the Khepera III robot, the dsPIC microcontroller running the communication protocol switches to the I2C slave mode. To control the robot, we used the "Khepera III Toolbox". This is a collection of scripts, programs and code modules for the Khepera III robot (see http://en.wikibooks.org/wiki/Khepera_III_Toolbox) allowing to design program to control the robot.

The previously described cognitive controllers, which are based on ANFIS have been designed with c language and implemented on the korebot. Both orientation (e.g. rotation) and position of the robot are computed by using an odometry process.

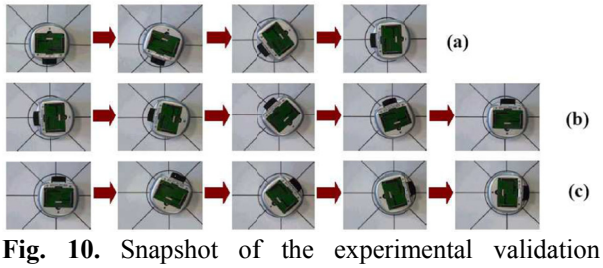

Fig. 10. Snapshot of the experimental validation concerning the real kheperaIII robot's orientation's control: rotation of $-90°$ (a), $-180°$ (b) and $-270°$ (c).

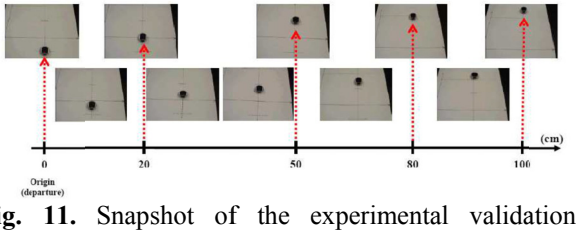

Fig. 11. Snapshot of the experimental validation concerning the real kheperaIII robot's position's control. From left to right: robot moves away 20, 50, 80 and 100 centimeters from its starting position.

The figures 10 and 11 show the first results relative to the experimental validation. Figure 10 shows the ANFIS based controller's validity as well as its performance on controlling the real kheperaIII robot's orientation. In this experimental validation, the robot is supposed to perform three successive rotations according to the following protocol: first, starting from its initial orientation (shown by the first picture of the figure 10-a), the robot performs a rotation of $-90°$ (e.g. "turning-right" operation) maintaining its position. Then, starting from its new orientation, the robot repeats two times the above-mentioned operation (e.g. turns-right) attaining successively the $-180°$ and $-270°$ (see figures 10-b and 10-c). Figure 11 shows the experimental validation relative to the ANFIS based controller's performance on controlling the real kheperaIII robot's position. In this second experimental validation, the robot is supposed to move respecting a straight line (e.g. without changing its initial orientation shown in the leftist picture of the figure) attaining four successive new positions: 20cm, 50cm, 80cm and 100cm from its starting position, respectively.

6. CONCLUSION

In this paper, we have proposed a control strategy for nonholonomic robot based on Adaptive Neural Fuzzy Inference System. This neuro-controller makes it possible the robot track a given reference trajectories. We have presented results relative to the control of a robot aiming to avoid an obstacle. We also presented the experimental validation results relative to the implementation of the proposed cognitive controller using a real kheperaIII robot. The obtained results show the viability of the proposed machine-learning based approach in controlling as well the robot's position as its orientation. The first interest of our approach is that the kinematics modeling is not needed to control the robot. Consequently, it is possible to extend our control strategy for another kind of robot as cart-like model, for example. The second interest is given by the possibility to design multi-level control: path planing, trajectory computing, and robot's controller.

Further works will focused, on this one hand, the design of the multi-level control strategy to the control of a robot's formation, and on the other hand, the experimental validation on the real robots kheperaIII.

7.ACKNOWLEDGMENTS

Authors express whish thank Mr. Fabien Gautero, student in Electrical Engineering and Computer Sciences department, for his active participation and accomplished efforts in implementation and experimental validation of the present work. Also, this work was supported by a doctoral fellowship from the China Scholarship Council (CSC). Authors wish to express their gratitude to CSC.

REFERENCES

[1] Fukuda,T.; Nakagawa,S.; Kawauchi,Y. and Buss, M. (1998). **"Self organizing robots based on cell**

structures – CEBOT". *Proc IEEE Int. Workshop Intell. Robot. Syst*., pp. 145–150.

[2] Parker, L. E. (2008). **"Multiple Mobile Robot Systems"**. *Springer Handbook of Robotics*, pp: 921-941.

[3] Cao, Y.U.; Fukunaga, A.S.; Kahng, A.B. and Meng, F. (1997) **"Cooperative mobile robotics: Antecedents and directions"**, *Autonomous Robots, Vol 4*, pp 1-23.

[4] Samson. C. (1995). **"Control of chained systems application to the path following and time varying point-stabilization"**. *IEEE Transactions on Automatic Control, vol 40.1*, pp.64-77.

[5] Kolmanovsky, I. and McClamroch N.H. (1995). **"Developments in nonholonomic control problems"**. *IEEE Control Systems Magazine, Vol 15*, pp.20-36.

[6] D'Andrea-Novel B.; Campion, G. and Bastin, G.(1995). **"Control of nonholonomic wheeled mobile robots by state feedback linearization"**. *The International Journal of Robotics Research, Vol. 14*, No. 6, 543-559.

[7] De Luca, A. and Di Benedetto, M.D. (1993). **"Control of noholonomic systems via dynamic compensation"**. *Workshop on system structure and control N2, vol. 29, no 6*, pp. 593-608.

[8] Fliess, M.; Levine, J. and Martin, P. (1995). **"Flatness and defect of non-linear systems: introductory theory and examples"**. *International Journal of Control, Vol. 61, No. 6*, pp. 1327-1361.

[9] Morin, P. and Samon, C. (2003). **"Practical stabilization of drift-less systems on Lie groups: the transverse function approach"**. *IEEE Transactions on Automatic Control, vol. 48, no9*, pp. 1496-1508.

[10] Barfoot, T.D. and Clark, C.M. (2004). **"Motion Planning for Formations of Mobile Robots"**. *Robotics and Autonomous Systems, Vol 46.2*, pp.65-78.

[11] Jang, J.-S.R and Sun, C.T. (1995). **" Neuro-fuzzy modeling and control"** *Proceedings of the IEEE*, pp. 378-406.

[12] J. Godjevac (1995). **"A Learning Procedure for a Fuzzy System: Application to Obstacle Avoidance"**. *In Proceedings of the International Symposium on Fuzzy Logic*, pp. 142-148

[13] Pascal, M. and Claude, S. (2008). **"Motion Control of Wheeled Mobile Robots"** *Handbook of Robotics, S. Bruno, K.S Oussama (Eds), Springer-Verlag Berlin Heidelberg*, pp.799-825.