ISSN 2007-9737

Integration of an Inverse Optimal Neural Controller with Reinforced-SLAM for Path Panning and Mapping in Dynamic Environments

Alma Y. Alanis, Nancy Arana-Daniel, Carlos Lopez-Franco, Edgar Guevara-Reyes

Universidad de Guadalajara, CUCEI, Zapopan, Jalisco, Mexico

{almayalanis, nancyaranad, clzfranco}@gmail.com, guevara_1@hotmail.com

Abstract. This work presents an artificial intelligence approach to solve the problem of finding a path and creating a map in unknown environments using Reinforcement Learning (RL) and Simultaneous Localization and Mapping (SLAM) for a differential mobile robot along with an optimal control system. We propose the integration of these approaches (two of the most widely used ones) for the implementation of robot navigation systems with an efficient method of control composed by a neural identifier and an inverse optimal control in order to obtain a robust and autonomous system of navigation in unknown and dynamic environments.

Keywords. Optimal neural control, reinforced-SLAM, path panning, mapping, dynamic environments.

1 Introduction

In order to achieve autonomous navigation, a robotic system must be able to interact with the environment, recognize, and reconstruct it, and choose and execute an appropriate action using a low level control system to accomplish its goal [1, 2, 3]. However, for a robot to be truly independent and able to cope with real environments, it has to solve many subtasks [4] such as 1) to map the environment and know where it itself is located in this map, this is the Simultaneous Localization and Mapping (SLAM) problem, 2) to plan paths and react to unexpected changes in the environment (the local and global path planning problem) and to have an efficient control algorithm, 3) to follow the paths planned (the control problem).

For autonomous robot navigation, an extensive class of controllers has been proposed for mobile robots [3–10]. Most of these references present only simulation results and the controllers are implemented in continuous time. A common problem when applying the standard control theory is that the required parameters are often either unknown at times or are subject to a change during operation. For example, the inertia of a robot as seen at the drive motor has many components which might include the rotational inertia of the motor rotor, the inertia of gears and shafts, the rotational inertia of its tires, the robot's empty weight, and its payload. Worse yet, there are elements between these components such as bearings, shafts, and belts which may have spring constants and friction loads [11].

Additionally, it is required to have an efficient method that integrates the algorithms which solve the above problems into a robust and real time system. This last issue is the one that this work solves with the use of intelligent algorithms that belong to the state of the art of robot navigation, control, and mapping for a kind of robots known as electrically nonholonomic mobile robots (Figure 1). The kind of robots used within this work is a common class of very popular robots with many useful applications in different fields: industrial, military, medical, search and rescue, and even educational fields. The state space model of the robot shown in Figure 1 can be expressed as follows [3, 12, 13, 14]:

$$\begin{aligned} \dot{x}_1 &= J\left(x_1\right)x_2, \\ \dot{x}_2 &= M^{-1}\left(-C\left(\dot{x}_1\right)x_2 - Dx_2 - \tau_d + NK_t x_3\right), \\ \dot{x}_3 &= L_a^{-1}\left(u - R_a x_3 - NK_E x_2\right), \end{aligned} \tag{1}$$

where each subsystem is defined as

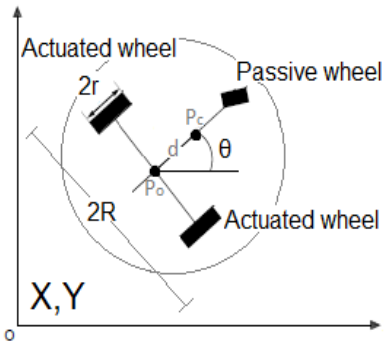

Fig. 1. Nonholonomic mobile robot (or car-like robot) with two actuated wheels

$$x_1 = \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \end{bmatrix} = \begin{bmatrix} X \\ Y \\ \theta \end{bmatrix},$$
$$x_2 = \begin{bmatrix} x_{21} \\ x_{22} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \quad (2)$$
$$x_3 = \begin{bmatrix} x_{31} \\ x_{32} \end{bmatrix} = \begin{bmatrix} i_{a_1} \\ i_{a_2} \end{bmatrix}$$

in which $x_{11} = X$, $x_{12} = Y$ are the coordinates of P_0; $x_{13} = \theta$ is the heading angle of the mobile robot; $x_{21} = v_1$, $x_{22} = v_2$ represent the angular velocities of the right and left wheels; and $x_{31} = i_{a_1}$, $x_{32} = i_{a_2}$ represent motor currents of the right and left wheels; and

$$J(x_1) = 0.5r \begin{bmatrix} \cos(x_{13}) & \cos(x_{13}) \\ \sin(x_{13}) & \sin(x_{13}) \\ R^{-1} & R^{-1} \end{bmatrix},$$
$$M = \begin{bmatrix} m_{13} & m_{12} \\ m_{12} & m_{11} \end{bmatrix}, \quad (3)$$
$$C(k) = 0.5rR^{-1}r^2 m_c d \begin{bmatrix} 0 & \dot{x}_{13} \\ -\dot{x}_{13} & 0 \end{bmatrix},$$
$$D = \begin{bmatrix} d_{11} & 0 \\ 0 & d_{22} \end{bmatrix},$$

$$m_{11} = 0.25R^{-2}r^2\left(mR^2 + I\right) + I_w,$$
$$m_{12} = 0.25R^{-2}r^2\left(mR^2 + I\right),$$
$$m = m_c + 2m_w,$$
$$I = m_c d^2 + 2m_w R^2 + I_c + 2I_m, \quad (4)$$
$$\tau = \begin{bmatrix} \tau_1 & \tau_2 \end{bmatrix}^T,$$
$$\tau_d = \begin{bmatrix} \tau_{d_1} & \tau_{d_2} \end{bmatrix}^T,$$

where R is half of the width of the mobile robot and r is the radius of the wheel, d is the distance from the center of mass P_c of the mobile robot to the middle point P_0 between the right and left driving wheels, m_c and m_w are the masses of the body and the wheel with a motor, respectively, I_c, I_w, and I_m are the moments of inertia of the body about the vertical axis through P_c, the wheel with a motor about the wheel axis, and the wheel with a motor about the wheel diameter, respectively. The positive terms d_{ii}, $i = 1, 2$, are the damping coefficients, $\tau \in \Re^2$ is the control torque applied to the wheels of the robot, $\tau_d \in \Re^2$ is a vector of disturbances including unmodeled dynamics. $K_T = diag\begin{bmatrix} k_{t_1} & k_{t_2} \end{bmatrix}$ is the motor torque constant, $i_a = \begin{bmatrix} i_{a_1} & i_{a_2} \end{bmatrix}^T$ is the motor current vector, $u \in \Re^2$ is the input voltage, $R_a = diag\begin{bmatrix} r_{a_1} & r_{a_2} \end{bmatrix}$ is the resistance, $L_a = diag\begin{bmatrix} l_{a_1} & l_{a_2} \end{bmatrix}$ is the inductance, $K_E = diag\begin{bmatrix} k_{e_1} & k_{e_2} \end{bmatrix}$ is the back electromotive force coefficient, and $N = diag\begin{bmatrix} n_1 & n_2 \end{bmatrix}$ is the gear ratio. Here, $diag[\bullet]$ denotes the diagonal matrix. The model is discretized using the Euler Methodology.

The paper is organized as follows. Section 2 presents the method that integrates the SLAM algorithm with a Reinforcement Learning (RL) algorithm in order to obtain a system which simultaneously maps the environment, localizes the robot, and learns a navigation policy which allows the robot to plan paths (global and local paths) which is needed to deal with dynamic

environments. In Section 3, a Neural Control is developed using a Recurrent High Order Neural Network to identify the robot model; also it shows an inverse optimal controller designed with the Lyapunov Control Function to guide the robot to follow the path planned with the navigation policy learned with the RL in the map constructed with SLAM. Section 4 shows the integration of the described system (Planning-Identifier-Controller), applied on a mobile robot in real time through wireless communication.

2 Reinforcement Learning-SLAM

In this section we present the subsystem dedicated to solve the tasks of path planning and mapping. The system described [15] integrates RL with a SLAM algorithm; it implements RL to learn to define a relation between situations and actions to maximize a numerical reward generated by the response of the environment. RL begins with a complete system that involves the environment and a definite goal [16]. The task consists in a series of actions that the robot has to perform to achieve its goal; then the mission of the learner is to find the action rules (policies) to optimally achieve a certain goal through its interaction with the environment. In this case the robot uses the RL algorithm known as Q-Learning [17]. The optimal Q value is defined as the sum of rewards obtained by performing an action on a state and following the optimal policy [17, 18]:

$$\begin{aligned} Q(s_k, a_k) &= E\left(R_k \left| s_k\right.\right) = E\left(\sum_{n=0}^{\infty} \gamma^k r_{k+n+1}\right) \\ &= Q(s_{k-1}, a_{k-1}) + \ldots \\ \ldots &+ \alpha^n \left[r_k + \gamma \max_{\alpha} Q(s_k, a_k) - Q(s_{k-1}, a_{k-1})\right] \end{aligned} \tag{5}$$

where from a state s_k, the action a_k selected from the set A is performed by an agent; as a result the agent receives a reward with an expected value $R(s_k, a_k)$, and the current state changes to the following state $s(k+1)$ according to the probability transition function $P\left(s(k+1)\left|s_k, a_k\right.\right)$. The α^n parameter stands for the learning rate that determines how much importance the system gives to the reward r_k obtained at time k by taking an action at state s_k, γ is the forgetting factor used to weight the importance that the system gives to long term rewards against immediate rewards. So a Q-value tells us how good an action is given a certain state.

Q-Learning algorithm was implemented in this work to obtain an intelligent exploration agent with capabilities to learn and to deal with dynamic environments while it is mapping and locating itself in its environments. The classic SLAM deals with environments which are considered by definition, static over time, so in dynamic environments, a SLAM algorithm must somehow manage moving objects. It can detect and ignore them; it can track them as moving landmarks, but it must not add a moving object to the map and assume it is stationary. The conventional SLAM solution is highly redundant. As noted in [19], landmarks can be removed from the map without loss of consistency, and it is often possible to remove large numbers of landmarks with little change in convergence rate. This property has been exploited to maintain a contemporaneous map by removing landmarks that have become obsolete due to changes in the environment. To explicitly manage moving objects, [20, 21] implemented an auxiliary identification routine and then removed the dynamic information from a data scan before sending it to their SLAM algorithm. Conversely, in [22] moving objects are added to their estimated state and models are provided for tracking both stationary and dynamic targets. Simultaneous estimation of moving and stationary landmarks is very costly due to the added predictive model. For this reason, our implemented solution involves a stationary fast-SLAM update combined with a simple RL module to deal with moving objects.

The agent continues selecting and executing actions (learned with RL), creating a path of states visited until it arrives to the desired position. When the training (RL process) is complete, the system uses SLAM during the exploration phase to make a reconstruction of the environment. When the system faces a dynamic environment in which it is hard to locate itself through SLAM, it uses the experience previously obtained with RL to guide the robot to a goal with the correct action, and the new environment information which is obtained by

navigating with RL is added to the map constructed with the SLAM algorithm [1, 2]. The SLAM problem requires a probability distribution function which is calculated for any time k. This probability distribution describes the posterior density of the location of both the robot and the characteristics of the map at a time k [23, 24]:

$$P\left(x_k, m \middle| Z_{0:k}, U_{0:k}, x_0\right) = \frac{P\left(z_k \middle| x_k, m\right) P\left(x_k, m \middle| Z_{0:k-1}, U_{0:k}, x_0\right)}{P\left(z_k \middle| Z_{0:k}, U_{0:k}\right)} \quad (6)$$

where x_k is the robot position at the time k, m is the map of the environment, $Z_{0:k-1}$ and $U_{0:k-1}$ are the sequence of observations and the sequence of control actions, respectively, from time 0 to time $k-1$, x_0 is the initial position of the robot. Eq. 6 is known as the Bayes filter, and it is a popular form of inference mapping due to its recursive formulation, allowing additional observations to be incorporated into the posterior density efficiently.

A lot of techniques have been proposed so far to implement the solution to the SLAM problem; the main difference between the different approaches is the representation of the environment and the representation of uncertainty when a technique builds the map and estimates the position of the robot [25]. Two of the most important solutions that have gained great acceptance are the Extended Kalman Filter (EKF) SLAM and the fast-SLAM [18], of which the second approach will be used because of the advantages provided by it [26] such as the number of features that can de handled (near 50,000 in simulated environments) and the logarithmic execution time (against quadratic time of EKF-SLAM). The reader can found in [24] a profound explanation of the algorithms mentioned.

With the obtained information, the system is capable of getting a path and constructing a map of the environment; however, to move the agent mobile robot, a control system is needed. Such system must give the correct action as the robot moves through the path designed by the RL.

3 Controller

To deal with the problem of following the path planned in the map created in Section 2, the implemented low control system uses a neural network approach that is presented in this section. When the neural network control approaches are presented, it is generally understood that a neural network is responsible for calculating the control action, but this can be divided in two groups: direct control and indirect one. In the first method, the control is performed by the neural network, in the second method indirect control is always based on models, and the objective is to use a neural network to identify the system model [27, 28].

The plant information is obtained by running the application in order to acquire a lot of data to describe the system behavior. This process consists in obtaining the parameters that best arrange the association between inputs and outputs. The goal of this stage is focused on giving the system a known input and observing how the system output behaves [27].

3.1 Neural Control

For control tasks, we use the high order extension of the Hopfield model called RHONN, which has a greater interaction between neurons; it is very flexible and allows the incorporation of a priori information about the structure of systems to the neural model. Now consider a MIMO system (multiple inputs, multiple outputs) [13]

$$x_{(k+1)} = F\left(x_{(k)}, u_{(k)}\right). \quad (7)$$

To identify the system we use a RHONN [16, 17] defined as

$$\hat{x}_{i(k+1)} = w_{i(k)}^T z_i\left(\hat{x}_{i(k)}, u_{(k)}\right), \quad (8)$$

where $\hat{x}_i$ is the state of the i-th neuron with $\left(i = 1, 2, \cdots, n\right)$, w_i is the online adapted weight vector, and $z_i\left(\hat{x}_{i(k)}, u_{(k)}\right)$ is given by

$$z_i\left(\hat{x}_{i(k)},u_{(k)}\right)=\begin{bmatrix} z_{i_1} \\ z_{i_2} \\ \vdots \\ z_{i_L} \end{bmatrix}=\begin{bmatrix} \prod_{j\in I_1}\xi_{ij}^{d_{ij}^{(1)}} \\ \prod_{j\in I_2}\xi_{ij}^{d_{ij}^{(2)}} \\ \vdots \\ \prod_{j\in I_{L_i}}\xi_{ij}^{d_{ij}^{(L_i)}} \end{bmatrix}, \quad (9)$$

L_i is the number of high order connections, $I_1, I_2, \cdots, I_{L_i}$ is a collection of non-ordered subsets, $1,2,3,\cdots,n+m$, n is the state dimension, and m is the number of external inputs, with $d_{ij}(\bullet)$ being non-negative integers, and ξ_i defined as

$$\xi_i=\begin{bmatrix} \xi_{i1} \\ \vdots \\ \xi_{in} \\ \xi_{in+1} \\ \vdots \\ \xi_{in+m} \end{bmatrix}=\begin{bmatrix} S\left(x_{1(k)}\right) \\ \vdots \\ S\left(x_{n(k)}\right) \\ u_{1(k)} \\ \vdots \\ u_{m(k)} \end{bmatrix}, \quad (10)$$

in which $u=\begin{bmatrix} u_1 & u_2 & \cdots & u_m \end{bmatrix}^T$ is the input vector to the neural network and $S(\bullet)$ is defined as

$$S(\zeta)=\frac{1}{1+e^{-\beta\zeta}},\quad \beta>0, \quad (11)$$

where ζ is a real value variable.

Consider the problem to approximate the general discrete time nonlinear system (7) by the following RHONN representation [17]:

$$\hat{x}_{i(k+1)}=w_{i(k)}^T z_i\left(x_{i(k)},u_{(k)}\right)+e_{z_i}. \quad (12)$$

In this case x_i represent the state model, e_{z_i} is a bounded approximation error which can be reduced by increasing the value of the adjustable weights. Assume that an ideal weight vector w_i^* exists, the ideal weight vector w_i is an artificial quantity required for analytical purpose, such that $\left\|e_{z_i}\right\|$ can be minimized on $\Omega_{z_i}\subset\Re^{L_i}$; w_i is an estimation defined as

$$\tilde{w}_{i(k)}=w_{i(k)}-w_i^*, \quad (13)$$

w_i is used for the stability analysis and w_i^* is a constant:

$$\tilde{w}_{i(k+1)}-\tilde{w}_{i(k)}=w_{i(k+1)}-w_{i(k)}. \quad (14)$$

Training of neural networks with EKF for both static networks and recurrent networks has proven reliable and practical for many applications. For the training of a neural network based on EKF, the weights become the state to be estimated. The objective of training is to find the optimal weight values that minimize the prediction errors. EKF is described as [18, 23]

$$\begin{aligned} K_{i(k)}&=P_{i(k)}H_{i(k)}M_{i(k)}^{-1}, \\ w_{i(k+1)}&=w_{i(k)}+\eta_i K_{i(k)}e_{i(k)}, \\ P_{i(k+1)}&=P_{i(k)}-K_{i(k)}H_{i(k)}^T P_{i(k)}+Q_{i(k)}, \end{aligned} \quad (15)$$

with

$$\begin{aligned} M_{i(k)}&=\left[R_{i(k)}+H_{i(k)}^T P_{i(k)}H_{i(k)}\right], \\ e_{i(k)}&=x_{i(k)}-\hat{x}_{i(k)}, \end{aligned} \quad (16)$$

where $K_{i(k)}\in\Re^{L_i\times m}$ is the Kalman gain matrix, $w_{i(k)}\in\Re^{L_i}$ is the weight (state) vector, $P_{i(k)}\in\Re^{L_i\times L_i}$ is the prediction error associated covariance matrix, $Q_{i(k)}\in\Re^{L_i\times L_i}$ is the state noise associated covariance matrix, $R_{i(k)}\in\Re^{m\times m}$ is the measurement noise associated covariance matrix, $H_{i(k)}\in\Re^{L_i\times m}$ is a matrix for which each entry (H_{ij}) is the derivative of one of the neural network outputs ($\hat{x}_i$) with respect to one neural network weight (w_{ij})

$$H_{ij} = \left[\frac{\partial \hat{x}_{i(k)}}{\partial w_{ij(k)}} \right], \tag{17}$$

L_i is the total number of weights of the neural network, x_i is the i-th plant state component, $\hat{x}_i$ is the i-th neural state component, η_i is a design parameter. Usually, P_i, Q_i, R_i are initialized as diagonal matrices with entries $P_i(0), Q_i(0), R_i(0)$, besides they are bounded [23, 24].

3.2 Controller

The main purpose of an optimal control is to obtain a control signal that causes the process to satisfy some physical restrictions [27, 28]. Then for the inverse optimal controller, the Lyapunov Control Function (LCF) is designed in order to satisfy the passivity condition. This states that a passive system can be stabilized by making a negative feedback from the output $u_k = -\alpha y_{(k)}$, with $\alpha > 0$

$$V\left(x_{(k)}, x_{ref(k)}\right) = \frac{1}{2}\left(x_{(k)} - x_{ref(k)}\right)^T K^T P K \left(x_{(k)} - x_{ref(k)}\right), \tag{18}$$

where $x_{ref(k)}$ is the desired path and K is a gain matrix further introduced to modify the rate of convergence of the tracking error [29, 30]:

$$\begin{aligned} x_{(k+1)} &= f\left(x_{(k)}\right) + g\left(x_{(k)}\right)u_{(k)} + d_{(k)}, \\ y_{(k)} &= h\left(x_{(k)}\right) + j\left(x_{(k)}\right)u_{(k)}. \end{aligned} \tag{19}$$

This solution is applied on the neural identifier developed in Section 3 to obtain a discrete-time neural model for the electrically driven nonholonomic mobile robot with $n = 5$ trained with the EKF as follows:

$$\begin{aligned} \hat{x}_{1(k+1)} &= w_{11(k)} S\left(x_{11(k)}\right) + w_{12(k)} S\left(x_{12(k)}\right), \\ \hat{x}_{2(k+1)} &= w_{21(k)} S\left(x_{11(k)}\right) + w_{22(k)} S\left(x_{12(k)}\right), \\ \hat{x}_{31(k+1)} &= w_{31(k)} S\left(x_{11(k)}\right) + w_{32(k)} S\left(x_{12(k)}\right), \\ \hat{x}_{4(k+1)} &= w_{41(k)} S\left(x_{11(k)}\right) + w_{42(k)} S\left(x_{12(k)}\right), \\ &+ w_{43(k)} S\left(x_{21(k)}\right) + w_{44(k)} S\left(x_{31(k)}\right) + G_1 u_{11(k)}, \\ \hat{x}_{5(k+1)} &= w_{51(k)} S\left(x_{11(k)}\right) + w_{52(k)} S\left(x_{12(k)}\right), \\ &+ w_{53(k)} S\left(x_{22(k)}\right) + w_{54(k)} S\left(x_{32(k)}\right) + G_2 u_{12(k)}. \end{aligned} \tag{20}$$

In order to facilitate the development of equations, we rewrite the RHONN as

$$\hat{x}_a = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \end{bmatrix}, \quad \hat{x}_b = \begin{bmatrix} \hat{x}_4 \\ \hat{x}_5 \end{bmatrix}, \quad \hat{x}_c = \begin{bmatrix} \hat{x}_6 \\ \hat{x}_7 \end{bmatrix}. \tag{21}$$

The goal is to force $\hat{x}_b$ to follow the desired reference signal $\hat{x}_{a,ref} = \begin{bmatrix} X & Y & \theta \end{bmatrix}^T$, this is achieved with the designed control. In addition, we force $\hat{x}_c$ to follow the previous control law $u_{2(k)} = x_c = w_b^{-1}\left(-w_{2(k)} z_{(k)} + u_{1(k)}\right)$; therefore, $\hat{x}_c = u_{2(k)}$ is the reference signal for the control law $u_{3(k)}$ leaving control laws as

$$\begin{aligned} h_i\left(\hat{x}_{i(k)}, x_{iref(k+1)}\right) &= g_i^T\left(\hat{x}_{i(k)}\right)\bar{P}\left(f\left(\hat{x}_{i(k)}\right) - x_{iref(k+1)}\right), \\ J_i\left(\hat{x}_{i(k)}\right) &= \frac{1}{2} g_i^T\left(\hat{x}_{i(k)}\right)\bar{P} g_i\left(\hat{x}_{i(k)}\right), \\ u_{i(k)} &= -\left[I_m + J_i\left(\hat{x}_{i(k)}\right)\right]^{-1} h_i\left(\hat{x}_{i(k)}, x_{iref(k+1)}\right). \end{aligned} \tag{22}$$

Finally, the pseudo-code of the algorithm which performs the integration between the planning (RL-SLAM)-identifier-controller is shown in Algorithm 1.

Algorithm.1 Planning-Identifier-Controller

1: Establish initial state and goal.
2: Obtain initial desired route Q_k.
3: Initialize SLAM map $P_0 = 0$, initial pose.
4: Get observations z_0.
5: Add new features.
6: **while** current state (x_k) ≠ Goal **do**
7: Update state map with z_k, x_k.
8: Update Q_k based on the observed.
9: Search obstacles.
10: **if** obstacle **then**
11: Look for a new route in Q_k and update map.
12: **if** No path found in Q_k **then**
13: It is not possible to find a route.
14: **else**
15: Upgrade Path.
16: **end if**
17: **end if**
18: Neural identification.
19: Perform next control action u_k.
20: Get odometry.
21: Get observations z_0.
22: Perform prediction step of SLAM.
23: Perform measurement update of SLAM.
24: Add new features.
25: **end while**

4 Experimental Results

The proposed schemes are carried out in a Matlab® and Simulink® (Matlab and Simulink are registered trademarks of the MathWorks, Inc.) environment, applied on a differential robot model with sensors of vision and movement and sensor readings corrupted by noise caused by wireless transmission. The anatomy, various components, and body axes of the Quanser (Quanser is a registered trademark of Quanser Inc.) are shown in Figure 2.

The Quanser Qbot is an innovative autonomous ground robot system [31]. The vehicle is comprised of an iRobot Create robotic platform, an array of optional infrared and sonar sensors, and a Logitech (Logitech is a registered trademark of

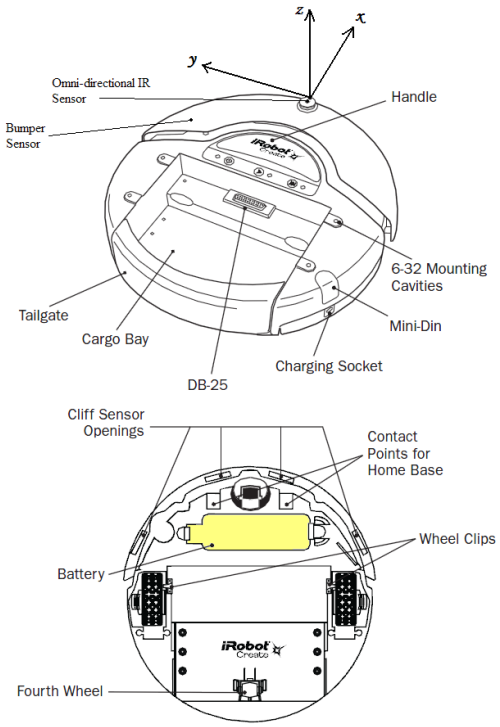

Fig. 2. Robot specifications

Logitech Inc.) Quickcam Pro 9000 USB camera, the diameter of the vehicle is 34 cm, and its height (without camera attachment) is 7 cm, it is driven by two differential drive wheels and comes with a bumper sensor and an omni-directional infrared receiver. The Quanser Controller Module (QCM) is an embedded system mounted on the vehicle, which uses the Gumstix computer to run QuaRC, Quanser's real-time control software [31].

The Qbot is accessible through three different block sets: the Roomba block set to drive the vehicle, the HIL block set to read from sensors and/or write to servo outputs, and, finally, the OpenCV block set to access the camera. The controllers are developed in Simulink with QuaRC on the host computer through wireless communication. This type of communication implies noise, delays, and uncertainties, which are absorbed by the neural network which learns in real time the system behavior and is capable of predicting the next state in order to avoid the lost/corrupted information.

To test the proposed system, the environment is established by a map which represents almost completely the environment to explore, with initial state (05; 01) and final goal (07; 13) (Figure 3). In this case a map of 33x19 pixels is used obtaining a total of 627 possible states and 5016 possible actions, in this case each state represents $1/3\ m^2$.

Several experiments were conducted in order to show the system capability to complete its goal despite changing environments. The experiments were performed with a sampling time of 0:05 seconds, a range of 33 cm vision for SLAM, an ability to detect an obstacle 33 cm away, a rate of 0:05 m/s, and some positions blocked, then the tests performed were adapted to the workbench.

As it was mentioned, the communication with the robot is wireless, which implies loss of information, noise, delays, uncertainties, attenuations, fading, among other problems. Due to the nature of the neural network used, the algorithm is able to learn from the previous experience generating its own weight distributions on the links. This learning ability to organize the information causes the power to appropriately respond to data or situations to which the robot has not been previously exposed.

Once the initial route is obtained, the navigation is started following the path, but once an obstacle is encountered by the SLAM observations, it checks the map state action and avoids the obstacle by decisions that come from the previous experience. It is important to note that each optimal route is modified by the robot in order to avoid obstacles, since there is a little displacement between the error in the pose estimation from SLAM and the noise that the control actions and data measurement have. Thus, the state map is corrupted each time with noise in the route planning part. This noise is discarded each time the planning takes places, thus, each time the system perceives a different map. The peaks on the graphics are caused by dynamic obstacles or unexpected changes on the environment; when this occurs, RL is used to correct the path with the learned optimal policy. Even though the changes are fast, the RL responds in real time to these changes as it can be seen in the figures. The results obtained on the different tests between the model (reference) and the system output for the X, Y position and an angle θ are shown in Figures 4

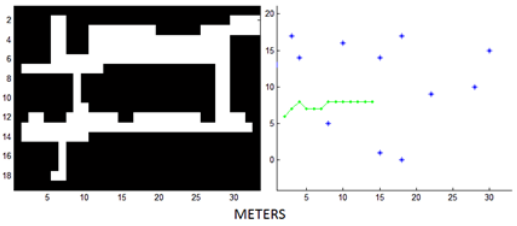

Fig. 3. Map and expected route for the test, where 1 pixel represents 1/3 meters and * are landmarks

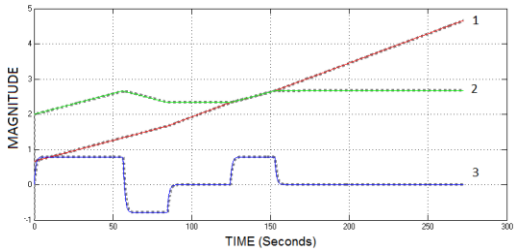

Fig. 4. System and reference behavior, in which 1) X position y 2) Y position in meters 3) angle in radians (Test 1)

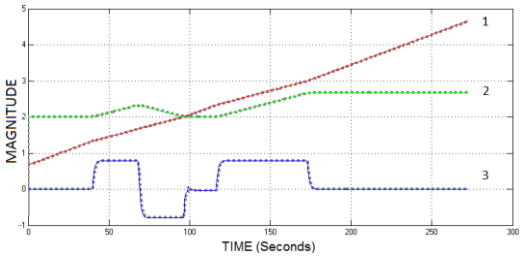

Fig. 5. System and reference, changing environment with some obstacles in real time (Test 2)

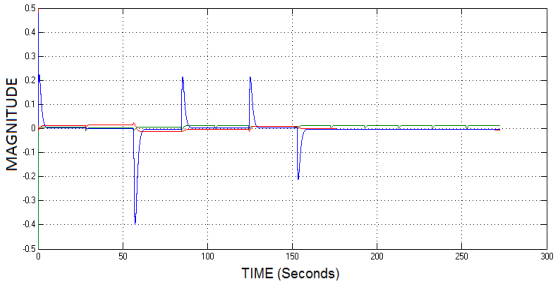

Fig. 6. Position error between system and reference, red: X position (cm), green: Y position (cm), blue: angle (radians) (Test 1)

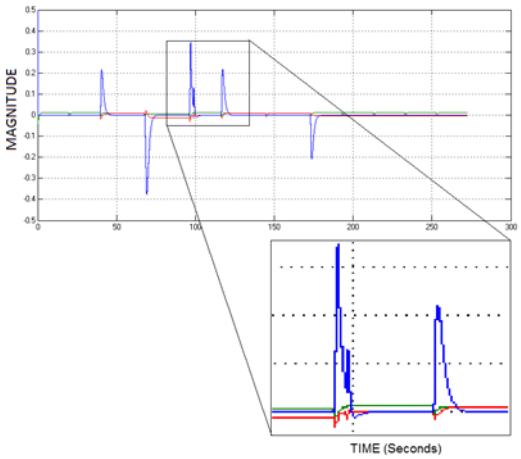

Fig. 7. Details of a disturbance rejection made for the neural network controller (Test 2)

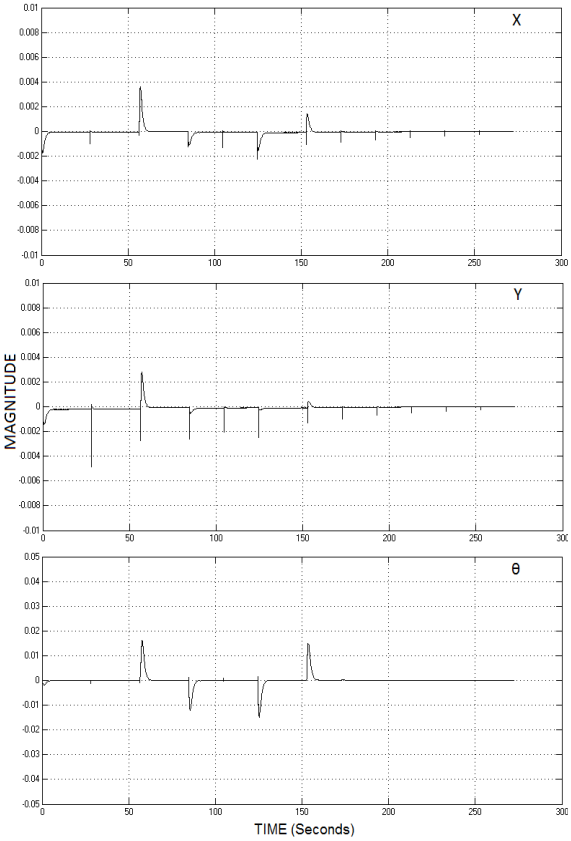

Fig. 8. Identification errors, where (X,Y) positions are in meters and θ is angle in radians (Test 1)

and 5, including environment movement. Position errors in tracking the trajectory can be seen in Figures 6 and 7, respectively.

The results of on-line neural identification performance for robot states (X,Y,θ) are shown in Figures 8 and 9. Finally, Figure 10 shows the maps created in the performed tests from a real environment as shown in Figure 11.

It is important to note that in this paper we do not consider a comparative analysis with previous works. In [32] a comparative analysis of the neural controller against sliding mode controllers is included, showing the superiority of the neural controllers for mobile robot control.

Regarding the RL-SLAM system, it is also relevant to mention that this integration allows the robot to solve the simultaneous localization and mapping problem (as it is done using any classical SLAM solver) but, in addition and at the same time and with the same information that is used to solve the SLAM problem, the path planning problem is solved thanks to the Reinforcement Learning algorithm. So, on each navigation episode the navigation agent 1) produces the map of the unknown environment, 2) localizes itself at each step of the navigation on this map, and at the same time 3) solves the global path planning problem as well as evades dynamic obstacles thanks to the RL system (this last step is not performed by the SLAM classical algorithms).

To summarize, in this work we achieved the integration of RL with a SLAM algorithm and a neural controller into a cycle in which the position on the map of the navigation agent estimated by SLAM is provided to the RL algorithm as input in order for this to be capable to compute the action that the neural controller has to execute to produce and follow a path from a start to a goal state. The action produced by the RL algorithm is sent as feedback to SLAM to follow the cycle until the goal state is reached.

4 Conclusions

A robot system capable of navigating unknown environments even with uncertainties in the robot model or in the environment has been developed. This can be attained because a RHONN structure

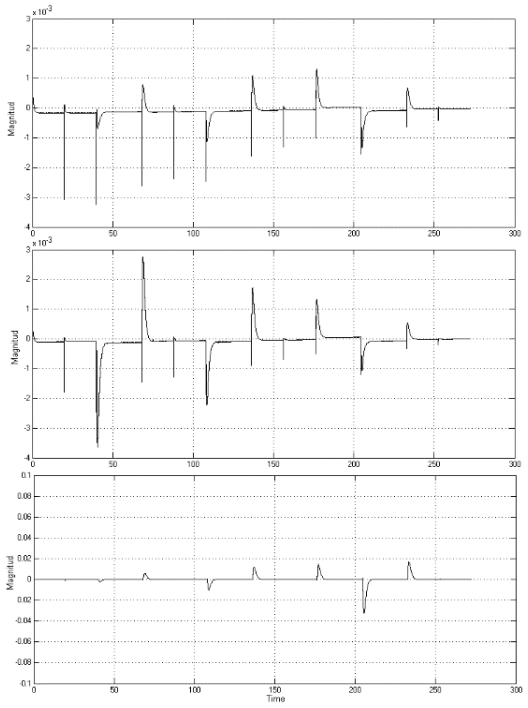

Fig. 9. Identification errors (X, Y, θ) respectively (Test 2).

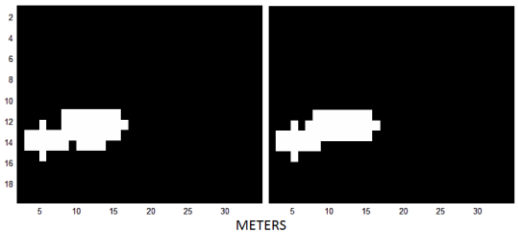

Fig. 10. Constructed maps in the performed tests

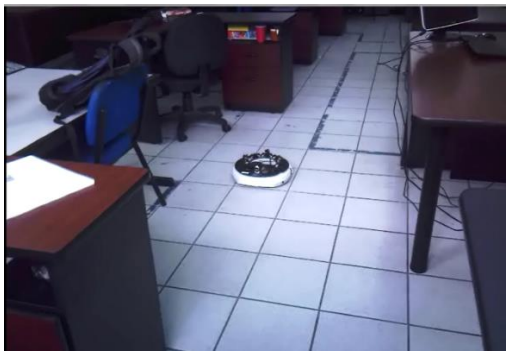

Fig. 11. Real environment

is used to design a neural identifier which is flexible and robust to noise. The results show the effectiveness of the proposed schemes; in addition, the qualities of RL are added to the algorithm to obtain a robust system capable of handling unknown and long state dynamic noisy environments.

Acknowledgements

The authors are thankful for the support by CONACYT Mexico through Projects 103191Y, 106838Y, 156567Y, and INFR-229696.

References

1. **Durrant-Whyte, H. & Bailey, T. (2006).** Simultaneous localization and mapping: part I. *IEEE Robotics & Automation Magazine*, Vol. 13, No. 2, pp. 99–110, doi: 10.1007/978-3-540-30301-5_38.
2. **Durrant-Whyte, H. & Bailey, T. (2006).** Simultaneous localization and mapping (slam), part II: State of the art. *IEEE Robotics & Automation Magazine,* Vol. 13, No. 2, pp. 108–117.
3. **Do, K., Jiang, Z.P., & Pan, J. (2004).** Simultaneous tracking and stabilization of mobile robots: an adaptive approach. *IEEE Transactions on Automatic Control,* Vol. 49, No. 7, pp.1147–1151.
4. **Fu, K.S., Gonzalez, R.C., & Lee, C.S.G (1987).** *Robotics: control, sensing, vision, and intelligence*. New York: McGraw-Hill.
5. **Salome, A., Alanis, A.Y., & Sanchez, E.N. (2011).** Discrete-time sliding mode controllers for nonholonomic mobile robots trajectory tracking problem. *Proc. of the 8th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE 2011),* pp. 1–6, doi: 10.1109/ICEEE.2011.6106564.
6. **Fierro, R. & Lewis, F.L. (1998).** Control of a nonholonomic mobile robot using neural networks. *IEEE Transactions on Neural Networks*, Vol. 9, No. 4, pp. 589–600, doi: 10.1109/72.701173.
7. **Jiang, Z.-P. & Nijmeijer, H. (1999).** A recursive technique for tracking control of nonholonomic systems in chained form. *IEEE Transactions on Automatic Control*, Vol. 44, No. 2, pp. 265–279, doi: 10.1109/9.746253.
8. **Kumbla, K.K. & Jamshidi, M. (1997).** Neural network based identification of robot dynamics used for neuro-fuzzy controller. *IEEE International Conference on Robotics and Automation (ICRA*

1997), Vol. 2, No. 1, pp. 1118–1123, doi: 10.1109/ROBOT.1997.614286.

9. **Raghavan, V. & Jamshidi, M. (2007).** Sensor fusion based autonomous mobile robot navigation. *IEEE International Conference on System of Systems Engineering (SOSE 2007)*, doi: 10.1109/SYSOSE.2007.4304295.
10. **Yang, J.-M. & Kim, J.-H. (1999).** Sliding mode control for trajectory tracking of nonholonomic wheeledmobile robots. *IEEE Transactions on Robotics and Automation*, Vol. 5, No. 3, pp. 578–587, doi: 10.1109/70.768190.
11. **Holland, J. (2003).** *Designing Autonomous Mobile Robots: Inside the Mind of an Intelligent Machine*. Newnes, Melbourne, Australia.
12. **Das, T. & Kar, I. (2006).** Design and implementation of an adaptive fuzzy logic-based controller for wheeled mobile robots. *IEEE Transactions on Control Systems Technology,* Vol. 14, No. 3, pp. 501–510, doi: 10.1109/TCST.2006.872536.
13. **Park, B.S., Yoo, S.J., Park, J.-B., & Choi, Y.-H. (2010).** A simple adaptive control approach for trajectory tracking of electrically driven nonholonomic mobile robots. *IEEE Transactions on Control Systems Technology,* Vol. 18, No. 5, pp. 1199–1206, doi: 10.1109/TCST.2009.2034639.
14. **Lopez-Franco, M., Salome-Baylon, A., Alanis, A.Y., & Arana- Daniel, N. (2011).** Discrete super twisting control algorithm for the nonholonomic mobile robots tracking problem. *8th International Conference on Electrical Engineering Computing Science and Automatic Control (CCE)*, pp. 1–5, doi: 10.1109/ICEEE.2011.6106692.
15. **Arana-Daniel, N., Rosales-Ochoa, R., & Lopez-Franco, C. (2011).** Reinforced slam for path planning and mapping in dynamic environments. *8th International Conference on Electrical Engineering Computing Science and Automatic Control (CCE)*, pp. 1–6, doi: 10.1109/ICEEE.2011.6106563.
16. **Sutton, R.S. & Barto, A.G. (1998).** *Reinforcement Learning: An Introduction*. MIT Press.
17. **Watkins, C.J.C.H. & Dayan, P. (1992).** Q-learning. *Machine Learning*, Vol. 8, No. 3, pp. 279–292, doi: 10.1007/BF00992698.
18. **Smart, W. & Kaelbling, L. (2002).** Effective reinforcement learning for mobile robots. *IEEE International Conference on Robotics and Automation,* Vol. 4, pp. 3404–3410, doi: 10.1109/ROBOT.2002.1014237.
19. **Andrade-Cetto, J. & Sanfeliu, A. (2003).** Temporal landmark validation in cml. *IEEE Int. Conf. Robot. Automat.,* pp. 1576–1581, doi: 10.1109/ROBOT.2003.1241819.
20. **Davison, A.J. (1998).** *Mobile Robot Navigation Using Active Vision.* PhD thesis, Univ. of Oxford.
21. **Davison, A.J. (2001).** 3d simultaneous localization and map-building using active vision for a robot moving on undulating terrain. *IEEE Conf. Comput. Vision Pattern Recognition,* pp. 384–391, doi: 10.1109/CVPR.2001.990501.
22. **Grisetti, G., Grzonka, S., Stachniss, C., Pfaff, P., & Burgard, W. (2007).** Efficient estimation of accurate maximum likelihood maps in 3D. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems,* pp. 3472–3478, doi: 10.1109/IROS.2007.4399030.
23. **Watkins, C.J.C.H. (1989).** *Learning from delayed rewards*. Ph.D. dissertation, King's College, Cambridge, UK.
24. **Huang, S., Wang, Z., & Dissanayake, G. (2004).** Time optimal robot motion control in simultaneous localization and map building (slam) problem. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3, pp. 3110–3115, doi: 10.1109/IROS.2004.1389884.
25. **Garulli, A., Giannitrapani, A., Rossi, A., & Vicino, A. (2005).** Mobile robot slam for line-based environment representation. *European Decision and Control Conference*, pp. 2041–2046, doi: 10.1109/CDC.2005.1582461.
26. **Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002).** Fastslam: A factored solution to the simultaneous localization and mapping problem. *AAAI National Conference on Artificial Intelligence*, pp. 593–598.
27. **Viñuela P. & Leon, I. (2004).** *Redes de neuronas artificiales: un enfoque práctico*. Pearson Education-Prentice Hall.
28. **Ricardo, V.G., (1999).** *Control de sistemas mediante redes neuronales, aprendizaje por refuerzo*. Master's thesis, Universidad Carlos III de Madrid, Madrid.
29. **Narendra, K. & Parthasarathy, K. (1990).** Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks,* Vol. 1, No. 1, pp. 4–27, doi: 10.1109/72.80202.
30. **Rovithakis, G. & Christodoulou, M. (2000).** Adaptative Control with Recurrent High-order Neural Networks: Theory and Industrial Applications. *Advances in Industrial Control.* Springer Verlag.
31. **Qbot (2014).** Quanser qbot: User manual. Number 830, revision 7, https://www.gumstix.com.
32. **Lopez-Franco, C., Lopez-Franco, M., Alanis, A. Y., Gomez-Avila, J. & Arana-Daniel, N. (2014).** Real-time inverse optimal neural control for image based visual servoing with nonholonomic mobile

robots. *Mathematical Problems in Engineering*, pp. 1–13, doi: 10.1155/2015/347410.

Alma Y. Alanis received the B.Sc. degree from Instituto Tecnologico de Durango (ITD), Durango Campus, Durango, Durango, in 2002, the M.Sc. and the Ph.D. degrees in Electrical Engineering from the Center of Research and Advanced Studies of the National Polytechnic Institute (CINVESTAV-IPN), Guadalajara Campus, Mexico, in 2004 and 2007, respectively. Since 2008 she has been with the University of Guadalajara, where she is currently a Chair Professor at the Department of Computer Science and member of the Intelligent Systems Research Group. She is also a member of the Mexican National Research System (SNI-1). Her research interest centers on neural control, backstepping control, block control, and their applications to electrical machines, power systems, and robotics.

Nancy Arana-Daniel received the M.Sc. degree in Computer Science in 2003 and the Ph.D. in Computer Science in 2007, both from the Center of Research and Advanced Studies (CINVESTAV-IPN), Guadalajara Campus, Mexico. She is currently a research fellow at the Department of Computer Science of the University of Guadalajara, Mexico, where she works together with other researchers of the Intelligent Systems Research Group. Her research interests focus on applications of geometric algebra, machine learning, optimization, computer vision, pattern recognition, and visually guided robot navigation.

Carlos Lopez-Franco received the Ph.D. degree in Computer Science from the Center of Research and Advanced Studies (CINVESTAV-IPN), Mexico, in 2007. He is currently a professor at the Department of Computer Science of the University of Guadalajara, Mexico, and a member of the Intelligent Systems Research Group. His research interests include geometric algebra, computer vision, robotics, and intelligent systems.

Edgar Guevara-Reyes received the B.Sc. in Computer Engineering in 2011 and the M.Sc. in Electrical and Computer Engineering from the University of Guadalajara in 2014. His interests include optimal and adaptive control and neural network controllers for dynamic systems.

Article received on 03/12/2014; accepted 24/04/2015.
Corresponding author is Alma Y. Alanis.