Discrete Time Neural Control of a Nonholonomic Mobile Robot integrating Stereo Vision Feedback

Michel López-Franco\textsuperscript{1}, Edgar N. Sanchez\textsuperscript{1}, Alma Y. Alanis\textsuperscript{2} and Carlos López-Franco\textsuperscript{2}

Abstract—In this paper, we present a discrete time neural controller for driving a nonholonomic mobile robot integrating stereo camera sensors. The proposed approach is based on a discrete-time high order neural network (RHONN) trained with an extended Kalman filter (EKF). The desired trajectory of the robot is computed during the navigation process using the stereo camera sensor. Simulation result is presented to show the effectiveness of the proposed control scheme.

I. INTRODUCTION

In this paper, a recurrent high order neural network (RHONN) is first used to identify the plant model, under the assumption of all the state is available for measurement.

In optimal control theory the main objective is to determine the control signals which will force a process to satisfy physical constraints and at the same time minimize a performance criterion \cite{1}. It includes a cost functional, which is a function of state and control variables: Unfortunately it is required to solve the Hamilton Jacobi Bellman (HJB) equation, which is not an easy task. The target of the inverse optimal control is to avoid the solution of the HJB equation \cite{2}.

For the inverse approach, a stabilizing feedback control is designed first, and then it is shown that this control optimizes a cost functional. The main characteristic of the inverse approach is that the meaningful cost function is a posteriori determined for the stabilizing feedback control law \cite{3}, \cite{4}, \cite{5}.

The objective of this paper is to present a controller for mobile robots which includes the robot dynamics. In addition, the reference for the controller are computed using visual data, acquired from a camera mounted on the robot. Using visual data the controller drives the nonholonomic robot from its current pose toward a desired pose, Fig. 1.a.

II. VISUAL BASED CONTROL

The use of visual feedback to control a robot is commonly termed \textit{visual servoing} or \textit{visual control} \cite{6}, \cite{7}. In this work the visual data is acquired from a stereo vision system that is mounted directly on the mobile robot, in which case motion of the robot induces camera motion, see Fig. 2.b.

The visual control objective is to minimize an error $e(t)$ defined as \cite{8}

$$e(t) = s(t) - s^*$$  \hspace{1cm} (1)

where $s(t)$ denote the features extracted from the current pose, and $s^*$ denote the features extracted from the desired pose.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig1}
\caption{a) Robot moving from initial pose to the desired pose. b) Translational $v_r$, $\omega_r$ and angular velocities computations from visual data.}
\end{figure}

In this paper we consider a nonholonomic mobile robot moving on a plane as shown in (Figure 2.a). Its pose is defined as $[x \ y \ \theta]^T$. Its kinematics model is that of a wheeled unicycle mobile robot

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig2}
\caption{a) Mobile robot with two actuated wheels. b) Coordinates systems of the mobile robot and stereo vision system.}
\end{figure}

$$\begin{align*}
\dot{x} &= v_r \cos \theta \\
\dot{y} &= v_r \sin \theta \\
\dot{\theta} &= \omega_r
\end{align*}$$ \hspace{1cm} (2)

where $v_r$ and $\omega_r$ represent the translational and angular velocities, respectively.

In order to estimate $v_r$ and $\omega_r$ by using visual data, several steps must be made, (Fig. 1.b.) First, the image is converted to HSV (Hue Saturation Value) colorspace \cite{9}.

---

\textsuperscript{1}CINVESTAV, Unidad Guadalajara, Jalisco 45015, Mexico. sanchez@gdl.cinvestav.mx
\textsuperscript{2}CUCEI, Universidad de Guadalajara, Apartado Postal 51-71, Col. las aguilas, C.P. 45080, Zapopan, Jalisco, Mexico, e-mail: almayaalanis@gmail.com, carlos.lopez@cucei.udg.mx

978-1-4673-6129-3/13/$31.00 ©2013 IEEE
Using this image we apply a mask, previously computed from a reference image, and then we obtain a segmented image. From the segmented image, we compute the boundaries using the Moore-neighbour tracing algorithm [10]; then to each boundary we compute the following metric \( m = 4\pi \text{area/perimeter}^2 \), if this is close to 1 then the boundary is more likely to be a circle.

From the detected circles, we compute their centroid. Later, using the centroids of the circles from the desired image, the current image and the corresponding depths, we estimate the robot’s pose. Finally, with the current and desired poses, we compute the velocities \( v_r, \omega_r \) to drive the robot from the current pose to the desired pose.

### A. Stereo Vision

The principle of stereo vision with parallel optical axes is displayed in Fig. 3. The 3D point \( P \) is projected onto the image plane of the left camera as \( p_L = [x_1, y_1] \), similarly \( p_R = [x_R, y_R] \) represents the projection of \( P \) onto the image plane of the right camera. Since the image plane of the left and right camera are located on the same plane, the \( y \)-coordinates in these two images are the same \((y_L = y_R)\), and the disparity is equal to the difference between the horizontal coordinates \((x_R - x_L)\).

Let \( P = (X, Y, Z) \in \mathbb{R}^3 \) denote a 3D point in the world. The coordinates of \( P \) on the left camera are

\[
P_L = [X + b/2, Y, Z]^T
\]

Similarly, the point \( P \) on the right camera is

\[
P_R = [X - b/2, Y, Z]^T
\]

Using the standard projective camera projection, we obtain

\[
x_L = (X + b/2)f/Z
\]

\[
x_R = (X - b/2)f/Z
\]

Similarly

\[
y_L = Yf/Z
\]

\[
y_R = Yf/Z
\]

The depth of the point \( P \) can be recovered from the \( x \)-coordinate of the images points \( x_L \) and \( x_R \), subtracting (6) from (5) we obtain

\[
Z = \frac{bf}{x_L - x_R}
\]

Similarly, we can also solve for \( X \) using (5), (6) and (9) and obtain

\[
X = \frac{b(x_L + x_R)}{2(x_L - x_R)}
\]

The \( Y \) value can be recovered with (7) or (8) since they have the same value, and (9) to get

\[
Y = \frac{by}{x_L - x_R}
\]

### B. Pose Estimation

The mobile robot moves on a 2D plane, thus we only need two coordinates to fully determine its pose \((x, y, \theta)\). Since the robot can not move in the \( Y \) direction (orthogonal to the plane), we can estimate its pose with respect to the planar target using only the \( Z \) and \( X \) values of the point \( P \).

Let \( Q_i^* = (Z_i^*, X_i^*) \) and \( Q_i = (Z_i, X_i) \) represent the 2D Euclidean point of the feature point \( P_i \) expressed in the frames \( F^* \) and \( F \), respectively. From Euclidean geometry, the relationship between the features is defined as

\[
Q_i^* = RQ_i + t
\]

where \( R \in \mathbb{R}^{2 \times 2} \) is the 2D rotation matrix and \( t = (t_x, t_y) \in \mathbb{R}^2 \) is the translation vector, Fig. 4.

To estimate the pose of the robot given the points \( Q_i \) and \( Q_i^* \) from the current and desired pose, we need to solve the following least-squares problem

\[
E(\theta, t) = \sum_{i=1}^{n} |R_{\theta}X_i + t - X_i^*|^2
\]

this problem can be solved in a closed form [11].

### C. Kinematic Planer

Once that the pose of the robot has been estimated, the next step is the estimation of the robot velocities which minimize the error between the current pose of the robot and the desired pose of the robot.

The path to track, is defined as the line \( L_d \) passing through the center of the stereo rig parallel to the optical axes of the
left and right cameras, Fig. 4. A line on the plane can be defined using the general equation of the line \((ax + by + c = 0)\), therefore the desired line is defined at the desired pose as

\[
L_d = [0 \ 1 \ 0]
\] (14)

The signed distance from the current pose of the robot and principal axis \(L_d\) at the desired pose is defined as

\[
d = [t_x \ t_y]^T \cdot L_d
\] (15)

The angular velocity of the robot must turn the robot toward the line \(L_d\) with

\[
\dot{\theta} = -K_d d, \quad K_d > 0
\] (16)

and adjust the orientation of the robot (heading angle) with

\[
\dot{\theta}_o = K_o (\theta^* - \theta), \quad K_o > 0
\] (17)

Then, the combined kinematic control law [12] used to generate the robot’s angular velocity for path following is defined as

\[
\omega_r = \dot{\theta}_d + \beta_o
\] (18)

The value of \(v_r\) is set to a constant value (e.g. 0.2 m/s), but when the robot is close to the desired pose the velocity is computed with

\[
v_r = \kappa_v \sqrt{t_x^2 + t_y^2}
\] (19)

III. NEURAL IDENTIFICATION

Consider a MIMO nonlinear system:

\[
\chi_{k+1} = F(\chi_k, u_k)
\] (20)

where \(\chi_k \in \mathbb{R}^n\) is the state of the system, \(u_k \in \mathbb{R}^m\) is the control input and \(F \in \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n\) is nonlinear function.

To identify system (20), we use a discrete-time recurrent high order neural network (RHONN) defined as

\[
x_{i,k+1} = w_i^T \varphi_i(\chi_k, u_k), \quad i = 1, \ldots, n
\] (21)

where \(x_i\) is the state of the \(i\)-th neuron, \(L_i\) is the respective number of high-order connections, \([I_1, I_2, \ldots, I_{L_i}]\) is a collection of non-ordered subsets of \([1, 2, \ldots, n + m]\), \(n\) is the state dimension, \(m\) is the number of external inputs, \(w_i\) is the respective on-line adapted weight vector, and \(\varphi_i(\chi_k, u_k)\) is given by

\[
\varphi_i(x_k, u_k) = \begin{bmatrix} \varphi_{i_1} \\ \varphi_{i_2} \\ \vdots \\ \varphi_{i_{L_i}} \end{bmatrix} = \begin{bmatrix} \Pi_{j \in I_1} \xi_{j_1} \xi_{j_1} \left( d_{i_1}(1) \\ d_{i_1}(2) \\ \vdots \\ d_{i_1}(L_i) \end{bmatrix}
\] (22)

with \(d_{ij}(k)\) being nonnegative integers, and \(\xi_i\) defined as follows:

\[
\xi_i = \begin{bmatrix} \xi_{i_1} \\ \vdots \\ \xi_{i_{n+m}} \end{bmatrix} = \begin{bmatrix} S(x_{i_1}) \\ \vdots \\ S(x_{i_{n+m}}) \end{bmatrix} u_1
\] (23)

In (23), \(u = [u_1, u_2, \ldots, u_m]^T\) is the input vector to the neural network, and \(S(\bullet)\) is defined by

\[
S(\zeta) = \frac{1}{1 + \exp(-\beta \zeta)} \quad \beta > 0
\] (24)

where \(\zeta\) is any real value variable.

Consider the problem to approximate the general discrete-time nonlinear system (20), by the following discrete-time RHONN series-parallel representation [13]:

\[
x_{i,k+1} = w_i^* \varphi_i(x_k, u_k) + \epsilon_{i}, \quad i = 1, \ldots, n
\] (25)

where \(\epsilon_{i}\) is a bounded approximation error, which can be reduced by increasing the number of the adjustable weights [13]. Assume that there exists an ideal weights vector \(w^*_i\) such that \(\|\epsilon_{i}\|\) can be minimized on a compact set \(\Omega_z \subset \mathbb{R}^L_i\). The ideal weight vector \(w^*_i\) is an artificial quantity required for analytical purpose [13]. In general, it is assumed that this vector exists and is constant but unknown. Let us define its estimate as \(\tilde{w}_i\) and the estimation error as

\[
\tilde{w}_{i,k} = w_{i,k} - w^*_{i}
\] (26)

The RHONN is trained with an Extended Kalman Filter (EKF) algorithm (29). Then, the dynamics of the identification error (31), can be expressed as

\[
\epsilon_{i,k+1} = \tilde{w}_{i,k} \varphi_i(\chi_k, u_k) + \epsilon_{i_k}
\] (27)

On the other hand the dynamics of (26) is

\[
\tilde{w}_{i,k+1} = \tilde{w}_{i,k} - \eta_i K_{i,k} e_k
\] (28)

It is possible to identify (20) by (21) due to the following theorem.

Theorem 1: [14]: The RHONN (21) trained with the EKF-based algorithm (29) to identify the nonlinear plant (20), ensures that the identification error (31) is semiglobally uniformly ultimately bounded (SGUUB); moreover, the RHONN weights remain bounded.

A. The EKF Training Algorithm

The best well-known training approach for recurrent neural networks (RNN) is the back propagation through time learning [15]. However, it is a first order gradient descent method and hence its learning speed can be very slow [16]. Recently, Extended Kalman Filter (EKF) based algorithms have been introduced to train neural networks [4], [14]. With the EKF based algorithm, the learning convergence is improved [16]. The EKF training of neural networks, both feedforward and
recurrent ones, has proven to be reliable and practical for many applications over the past ten years [4]. It is known that Kalman filtering (KF) estimates the state of a linear system with additive state and output white noises [17], [18]. For KF-based neural network training, the network weights become the states to be estimated. In this case, the error between the neural network output and the measured plant output can be considered as additive white noise. Due to the fact that the neural network mapping is nonlinear, an EKF-type is required (see [19] and references therein).

The training goal is to find the optimal weight values which minimize the prediction error. The EKF-based training algorithm is described by [17]:

\[ K_{i,k} = P_{i,k} H_{i,k} M_{i,k} \]
\[ w_{i,k+1} = w_{i,k} + \eta_i K_{i,k} e_{i,k} \]
\[ P_{i,k+1} = P_{i,k} - K_{i,k} H_{i,k}^T P_{i,k} + Q_{i,k} \]

with

\[ M_{i,k} = [R_{i,k} + H_{i,k}^T P_{i,k} H_{i,k}]^{-1} \]
\[ e_{i,k} = \chi_{i,k} - x_{i,k} \]

where \( P_i \in \mathbb{R}^{l_i \times l_i} \) is the prediction error associated covariance matrix, \( w_i \in \mathbb{R}^{l_i} \) is the weight (state) vector, \( L_i \) is the total number of neural network weights, \( \chi_i \in \mathbb{R} \) is the \( i \)-th plant state component, \( \eta_i \) is a design parameter, \( K_i \in \mathbb{R}^{l_i \times m} \) is the Kalman gain matrix, \( Q_i \in \mathbb{R}^{l_i \times l_i} \) is the state noise associated covariance matrix, \( R_i \in \mathbb{R}^{n \times m} \) is the measurement noise associated covariance matrix, \( H_i \in \mathbb{R}^{k \times m} \) is a matrix, for which each entry \( (H_{i,j}) \) is the derivative of one of the neural network output, \( (x_i) \), with respect to one neural network weight, \( (w_{i,j}) \), as follows

\[ H_{i,j,k} = \left[ \frac{\partial x_{i,k}}{\partial w_{j,k}} \right]_{w_{i,k} = w_{i,k-1}} \]

(32)

\[ i = 1, \ldots, n \quad \text{and} \quad j = 1, \ldots, L_i \]

Usually \( P_i, Q_i \) and \( R_i \) are initialized as diagonal matrices, with entries \( P_i(0), Q_i(0) \) and \( R_i(0) \), respectively. It is important to note that \( H_{i,k}, K_{i,k} \) and \( P_{i,k} \) for the EKF are bounded [18].

IV. INVERSE OPTIMAL CONTROL

Let consider a nonlinear affine system

\[ x_{k+1} = f(x_k) + g(x_k) u_k \]
\[ x_0 = x(0) \]

(33)

where \( x_k \in \mathbb{R}^n \) is the state of the system at time \( k \in \mathbb{N} \), \( u \in \mathbb{R}^m \) : \( \mathbb{R}^n \rightarrow \mathbb{R}^n \), \( g : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m} \), are smooth and bounded mappings. We assume \( f(0) = 0 \). \( \mathbb{N} \) denotes the set of nonnegative integers. The following meaningful cost functional is associated with the trajectory tracking problem for system (33)

\[ \mathcal{L}(z_k) = \sum_{n=k}^{\infty} (l(z_n) + u_n^T R(z_n) u_n) \]

(34)

where \( z_k = x_k - x_d,k \) with \( x_d,k \) as the desired trajectory for \( x_k \); \( z_k \in \mathbb{R}^n \); \( \mathcal{L}(z_k) : \mathbb{R}^n \rightarrow \mathbb{R}^+ \); \( l(z_k) : \mathbb{R}^n \rightarrow \mathbb{R}^+ \) is a positive semi-definite function and \( R(z_k) : \mathbb{R}^n \rightarrow \mathbb{R}^{m \times m} \) is a real symmetric positive definite weighting matrix. The entries of \( R(z_k) \) can be fixed or can be functions of the system state in order to vary the weighting on control efforts according to the state value [1]. Considering the state feedback control design problem, we assume that the full state \( x_k \) is available. Using the optimal value function \( \mathcal{L}^*(z_k) \) for (34) as Lyapunov function \( V(z_k) \), equation (34) can be rewritten as

\[ V(z_k) = l(z_k) + u_k^T R(z_k) u_k \]
\[ + \sum_{n=k+1}^{\infty} (l(z_n) + u_n^T R(z_n) u_n) \]
\[ = l(z_k) + u_k^T R(z_k) u_k + V(z_{k+1}) \]

where it is required the following boundary condition \( V(0) = 0 \) so that \( V(z_k) \) becomes a Lyapunov function. From Bellman optimality principle [20] [21], it is known that, for the infinite horizon optimization case, the value function \( V(z_k) \) becomes time invariant and satisfies the discrete-time (DT) Bellman equation [21], [22], [23]

\[ V(z_k) = \min_{u_k} \{ l(z_k) + u_k^T R(z_k) u_k + V(z_{k+1}) \} \]

(35)

where \( V(z_{k+1}) \) depends on both \( z_k \) and \( u_k \) by means of \( z_{k+1} \) in (33). Note that the DT Bellman equation is solved backward in time [22]. In order to establish the conditions that the optimal control law must satisfy, we define the discrete-time Hamiltonian \( H(z_k, u_k) \) as

\[ H(z_k, u_k) = l(z_k) + u_k^T R(z_k) u_k + V(z_{k+1}) - V(z_k) \]

A necessary condition that the optimal control law should satisfy is

\[ \frac{\partial H(z_k, u_k)}{\partial u_k} = 0, \]

(36)

Therefore, the optimal control law to achieve trajectory tracking is formulated as

\[ u_k^* = -\frac{1}{2} R^{-1}(z_k) g^T(z_k) \frac{\partial V(z_{k+1})}{\partial z_{k+1}} \]

(36)

with the boundary condition \( V(0) = 0 \). For solving the trajectory tracking inverse optimal control problem, it is necessary to solve the following HJB equation

\[ l(z_k) + V(z_{k+1}) - V(z_k) + \frac{1}{4} \frac{\partial V(z_{k+1})}{\partial z_{k+1}} g(z_k) \times R^{-1}(z_k) g^T(z_k) \frac{\partial V(z_{k+1})}{\partial z_{k+1}} = 0 \]
which is a challenging task. To overcome this problem, we propose to solve the inverse optimal control problem.

**Definition 1:** Consider the tracking error as \( z_k = x_k - x_{\hat{k},k} \), being \( x_{\hat{k},k} \) the desired trajectory for \( x_k \). Let define the control law

\[
\begin{align*}
  u_k^* &= -\frac{1}{2} R^{-1}(z_k) g^T(x_k) \frac{\partial V(z_{k+1})}{\partial z_{k+1}} \\
  &= -\frac{1}{2} R^{-1}(z_k) g^T(x_k) \frac{\partial \epsilon_k}{\partial z_{k+1}} \\
  \hat{V} := V(z_{k+1}) - V(z_k) + u_k^T R(z_k) u_k^* \leq 0
\end{align*}
\]

It will be inverse optimal (globally) stabilizing along the desired trajectory \( x_{\hat{k},k} \) if:

(i) It achieves (global) asymptotic stability of \( x_k = 0 \) for system (33) along reference \( x_{\hat{k},k} \);

(ii) \( V(z_k) \) is (radially unbounded) positive definite function such that inequality

\[
\hat{V} := V(z_{k+1}) - V(z_k) + u_k^T R(z_k) u_k^* \leq 0
\]

is satisfied.

Selecting \( \hat{I}(z_k) := -\hat{V} \), then \( V(z_k) \) is a solution for (36) and cost functional (34) is minimized.

As established in **Definition 1**, the inverse optimal control law for trajectory tracking is based in knowledge of \( V(z_k) \). Then, a CLF \( V(z_k) \) is proposed, such that (i) and (ii) are guaranteed. Hence, instead of solving (36) for \( V(z_k) \) a quadratic candidate CLF \( V(z_k) \) for (37) is proposed with the form:

\[
V(z_k) = \frac{1}{2} z_k^T P z_k \quad \text{for} \quad P = P^T > 0
\]

in order to ensure stability of the tracking error \( z_k \), where

\[
z_k = x_k - x_{\hat{k},k} = \begin{bmatrix} x_{1,k} - x_{1,\hat{k},k} \\ \vdots \\ x_{n,k} - x_{n,\hat{k},k} \end{bmatrix}
\]

The control law (37) with (38), which is referred to as the inverse optimal control law, optimizes the meaningful cost functional of the form (34). Consequently, by considering \( V(z_k) \) as in (38), control law (37) takes the following form

\[
\begin{align*}
  u_k^* &= -\frac{1}{2} R^{-1}(z_k) g^T(x_k) \frac{\partial z_{k+1}}{\partial z_{k+1}} P z_{k+1} \\
  &= -\frac{1}{2} R^{-1}(z_k) g^T(x_k) \frac{\partial z_{k+1}}{\partial z_{k+1}} P z_{k+1} \\
  &= -\frac{1}{2} \left( R(z_k) g^T(x_k) P g(z_k) \right)^{-1} \\
  &\quad \times g^T(x_k) P (f(x_k) - x_{\hat{k},k+1})
\end{align*}
\]

\( P \) and \( R(z_k) \) are positive definite and symmetric matrices; thus, the existence of the inverse in (39) is ensured.

V. APPLICATION TO AN ELECTRICALLY DRIVEN NONHOLONOMIC MOBILE ROBOT

A. Robot Description

We consider a mobile robot with two actuated wheels as shown in Fig. 1. The dynamic of an electrically driven nonholonomic mobile robot can be expressed in the following state-space model [3], [5], [24], [25]

\[
\begin{align*}
  \dot{x}_1 &= J(\chi_1) x_2 \\
  \dot{x}_2 &= M^{-1}(-C(\chi_1) x_2 - D x_2 - \tau_d + N K_T x_3) \\
  \dot{x}_3 &= L_a^{-1}(u - R_a x_3 - N K_E x_2)
\end{align*}
\]

where each subsystem is defined as

\[
\begin{align*}
  \chi_1 &= [\chi_{11}, \chi_{12}, \chi_{13}]^T \\
  \chi_2 &= [\chi_{21}, \chi_{22}]^T \\
  \chi_3 &= [\chi_{31}, \chi_{32}]^T
\end{align*}
\]

with

\[
\begin{align*}
  J(\chi_1) &= 0.5 r \begin{bmatrix} \cos(\chi_{13}) & \cos(\chi_{13}) \\ \sin(\chi_{13}) & -\sin(\chi_{13}) \end{bmatrix} \\
  M &= \begin{bmatrix} m_{11} & m_{12} \\ m_{12} & m_{11} \end{bmatrix} \\
  C(\chi) &= 0.5 r^2 \begin{bmatrix} 0 & \chi_{13} \\ -\chi_{13} & 0 \end{bmatrix} \\
  D &= \begin{bmatrix} d_{11} & 0 \\ 0 & d_{22} \end{bmatrix} \\
  m_{11} &= 0.25 r^2 \sin^2(x_2) + I_w \\
  m_{12} &= 0.25 r^2 \cos^2(x_2) - I \\
  m &= m_c + 2 m_w \\
  I &= m_c d_1^2 + 2 m_w d_2^2 + I_c + 2 I_m \\
  \tau &= [\tau_1, \tau_2]^T \\
  \tau_d &= [\tau_d_1, \tau_d_2]^T
\end{align*}
\]

where \( \chi_{11} = x, \chi_{12} = y \) are the coordinates of \( P_0 \) and \( \chi_{13} = \theta \) is the heading angle of the mobile robot, \( \chi_{21} = v_1, \chi_{22} = v_2 \) represent the angular velocities of right and left wheels, respectively and \( \chi_{31} = i_{a1}, \chi_{32} = i_{a2} \) represent motor currents of right and left wheels, respectively. \( R \) 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 mass of the body and the wheel with a motor, respectively. \( I_c, I_w, \) and \( I_m \) are the moment 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. \( \tau \in \mathbb{R}^2 \) is the control torque applied to the wheels of the robot. \( \tau_d \in \mathbb{R}^2 \) is a vector of disturbances including unmodeled dynamics. \( K_T = diag[k_{a1}, k_{a2}] \) is the motor torque constant, \( i_a = [i_{a1}, i_{a2}] \) is the motor current vector, \( u \in \mathbb{R}^2 \) is the input voltage, \( R_a = diag[r_{a1}, r_{a2}] \) is the resistance, \( L_a = \frac{1}{2} [i_{a1}, i_{a2}] \) is the inductance, \( K_E = diag[k_{e1}, k_{e2}] \) is the back electromotive force coefficient and \( N = diag[n_{1}, n_{2}] \) is the gear ratio. Here, \( diag[\cdot] \) denotes the diagonal matrix. Model (40) is discretized using the Euler Methodology.
B. Neural Identification Design

The physical parameters for the mobile robot simulations are selected as:

\[
\begin{align*}
R &= 0.75m \\
d &= 0.3m \\
r &= 0.15m \\
m_c &= 30kg \\
m_w &= 1kg \\
I_c &= 15.625kgm^2 \\
I_w &= 0.005kgm^2 \\
K_F &= diag[0.048, 0.048]H \\
N &= diag[62.55, 62.55]N \cdot m/A
\end{align*}
\]

To this end, we apply the neural identifier, developed in section III, to obtain a discrete-time neural model for the electrically driven nonholonomic mobile robot (40), with \( n = 7 \) trained with the EKF (29), as follows

\[
\begin{align*}
x_{1,k+1} &= w_{11,k} S(\chi_{11,k}) + w_{12,k} S(\chi_{12,k}) + w_{13,k} x_3 + w_{14,k} x_4 \\
x_{2,k+1} &= w_{21,k} S(\chi_{11,k}) + w_{22,k} S(\chi_{12,k}) + w_{23,k} x_3 + w_{24,k} x_4 \\
x_{3,k+1} &= w_{31,k} S(\chi_{11,k}) + w_{32,k} S(\chi_{12,k}) + w_{33,k} x_3 + w_{34,k} x_4 \\
x_{4,k+1} &= w_{41,k} S(\chi_{11,k}) + w_{42,k} S(\chi_{12,k}) + w_{43,k} S(\chi_{11,k}) + w_{44,k} S(\chi_{31,k}) + w_{51,k} x_6 \\
x_{5,k+1} &= w_{51,k} S(\chi_{11,k}) + w_{52,k} S(\chi_{12,k}) + w_{53,k} S(\chi_{22,k}) + w_{54,k} S(\chi_{32,k}) + w_{55,k} x_7 \\
x_{6,k+1} &= w_{61,k} S(\chi_{11,k}) + w_{62,k} S(\chi_{12,k}) + w_{63,k} S(\chi_{21,k}) + w_{64,k} S(\chi_{31,k}) + w_{65,k} x_6 + w_{66,k} x_7 \\
x_{7,k+1} &= w_{71,k} S(\chi_{11,k}) + w_{72,k} S(\chi_{12,k}) + w_{73,k} S(\chi_{22,k}) + w_{74,k} S(\chi_{32,k}) + w_{75,k} x_7 + w_{76,k} x_6
\end{align*}
\]

where \( x_1 \) and \( x_2 \) identify the \( x \) and \( y \) coordinates, respectively; \( x_3 \) identifies the robot angle; \( x_4 \) and \( x_5 \) identify the angular velocities of right and left wheels, respectively; finally, \( x_6 \) and \( x_7 \) identify the motor currents, respectively. The NN training is performed on-line, and all of its states are initialized in a random way. The RHONN parameters are heuristically selected as:

\[
\begin{align*}
P_1 (0) &= 1 \times 10^8 \\
P_2 (0) &= 1 \times 10^2 \\
P_3 (0) &= 1 \times 10^8 \\
P_4 (0) &= 1 \times 10^2 \\
P_5 (0) &= 1 \times 10^2 \\
P_6 (0) &= 1 \times 10^2 \\
R_1 (0) &= 1 \times 10^4 \\
R_2 (0) &= 1 \times 10^4 \\
R_3 (0) &= 1 \times 10^4 \\
R_4 (0) &= 1 \times 10^3 \\
R_5 (0) &= 1 \times 10^3 \\
R_6 (0) &= 1 \times 10^3 \\
R_7 (0) &= 1 \times 10^3 \\
Q_1 (0) &= 5 \times 10^5 \\
Q_2 (0) &= 5 \times 10^5 \\
Q_3 (0) &= 5 \times 10^5 \\
Q_4 (0) &= 1 \times 10^3 \\
Q_5 (0) &= 1 \times 10^3 \\
Q_6 (0) &= 1 \times 10^3 \\
Q_7 (0) &= 1 \times 10^3
\end{align*}
\]

It is important to consider that for the EKF-learning algorithm the covariances are used as design parameters ([4], [26]). The neural network structure (41) is determined heuristically in order to minimize the state estimation error. Simulation is performed with a sampling time of 0.0005s. The results are presented as follows

C. Control Synthesis

In order to facilitate the controller synthesis, we rewrite neural network (41) in a block structure form as

\[
\begin{align*}
x_{1,k+1} &= x_{1,k} + w_{1,k} \varphi_1 (x_{1,k}) + w_{2,k} \chi_{2,k} \\
x_{2,k+1} &= x_{2,k} + w_{2,k} \varphi_2 (x_{1,k}, x_{2,k}) + w_{2,k} \chi_{3,k} \\
x_{3,k+1} &= x_{3,k} + w_{3,k} \varphi_3 (x_{1,k}, x_{2,k}, \chi_{3,k}) + w_{3,k} u_k \\
\end{align*}
\]

with \( \chi_{1,k}, \chi_{2,k}, \chi_{3,k}, \varphi_1, \varphi_2, \varphi_3, w_{1,k}, w_{2,k}, w_{3,k}, w_{1,k}, w_{2,k} \) and \( w_{3,k} \) of appropriated dimension according to (42).

The goal is to force the state \( x_{1,k} \) to track a desired reference signal \( \chi_{1,k} \). This is achieved by designing a control law as described in section IV. First the tracking error is defined as

\[
z_{1,k} = x_{1,k} - \chi_{1,k}
\]

Then using (42) and introducing the desired dynamics for \( z_{1,k} \) results in

\[
z_{1,k+1} = w_{1,k} \varphi_1 (x_{1,k}) + w_{1,k} \chi_{2,k} - \chi_{1,k+1}
\]

\[
= K_1 z_{1,k}
\]

where \( K_1 = diag \{ k_1, k_2, k_3 \} \) with \( |k_1|, |k_2|, |k_3| < 1 \). The desired value \( \chi_{2,k} \) for the pseudo-control input \( \chi_{3,k} \) is calculated from (43) as

\[
\chi_{2,k} = \left( w_{1,k} \right)^{-1} ( -w_{1,k} \varphi_1 (x_{1,k}) + \chi_{1,k+1} + K_1 z_{1,k})
\]

At the second step, we introduce a new variable as

\[
z_{2,k} = x_{2,k} - \chi_{2,k}
\]

Then using (42) and introducing the desired dynamics for \( z_{2,k} \) results in

\[
z_{2,k+1} = w_{2,k} \varphi_2 (x_{1,k}, \chi_{2,k}) + w_{2,k} \chi_{3,k} - \chi_{2,k+1}
\]

\[
= K_2 z_{2,k}
\]

where \( K_2 = diag \{ k_{12}, k_{22} \} \) with \( |k_{12}|, |k_{22}| < 1 \). The desired value \( \chi_{3,k} \) for the pseudo-control input \( \chi_{3,k} \) is calculated from (45) as

\[
\chi_{3,k} = \left( w_{2,k} \right)^{-1} ( -w_{2,k} \varphi_2 (x_{1,k}, \chi_{2,k}) + \chi_{2,k+1} + K_2 z_{2,k})
\]

At the third step, we introduce a new variable as

\[
z_{3,k} = x_{3,k} - \chi_{3,k}
\]

Taking one step ahead, we have

\[
z_{3,k+1} = w_{3,k} \varphi_3 (x_{1,k}, x_{2,k}, \chi_{3,k}) + u_k - \chi_{3,k+1}
\]
where $u_k$ is defined as

$$u_k = \frac{1}{2} \left( R(z_k) + g^T(x_k)P y(z_k) \right)^{-1} \times g^T(x_k)P(x_k) - x_{\delta,k+1}$$

(48)

where the controllers parameters are selected heuristically as

$$P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

and

$$w_1'_{1,k} = \begin{bmatrix} \cos(x_3) \\ \sin(x_3) \end{bmatrix}, w_1'_{2,k} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad w_3'_{k} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

**D. Simulation Results**

In this section we present the simulation and experimental results of our proposed discrete-time inverse optimal neural controller with stereo vision feedback. Simulations have been performed using Matlab-Simulink.

In the simulation the robot moves under the action of the proposed controller, the controller uses as references the linear and angular velocities computed from the stereo vision algorithm. In the simulation the initial pose of the robot is $[0 \ 0 \ 0]^T$, and the desired pose is $[3.8 \ -0.8 \ 0]^T$.

The sampling time of the simulation was $T=0.01s$. Simulations results are presented as follows: In Fig. 5 we show the linear and angular velocities used as references by the proposed controller, these velocities are computed by the stereo vision algorithm using the current and desired images of the target object. Fig. 6 shows the identification performance for x-axis, y-axis and θ angle. Fig. 8 shows the trajectory tracking results. In Fig. 9 we present the tracking errors. In Fig. 10 we show the applied control signal for the left and right wheels. Fig. 11 presents the current identification for simulation in left and right wheels.

**VI. CONCLUSIONS**

In this paper we have presented a discrete-time inverse optimal neural control with visual feedback. The controller is able to accomplish trajectory tracking of a nonlinear system; this controller is inverse optimal in the sense that it minimizes a meaningful cost functional. The mobile robot dynamics at the actuator level as well as its kinematics and dynamics uncertainties are considered in the construction of the controller by means of neural identification. Visual data acquired from a stereo vision sensor is used to estimate the robot’s pose; with this pose we were able to compute the reference velocities $v_r, \omega_r$ that allow the controller to drive
the nonholonomic robot from its current pose toward the desired pose. Simulation results are presented to illustrate the effectiveness of the proposed controller.

Acknowledgement: The authors thank the support CONACYT Mexico, through Projects 103191Y, 106838Y, 156567Y and 131678Y.

REFERENCES


Fig. 9. Applied control signal for the left and right wheels respectively.

Fig. 10. Current identification for simulation in left and right wheels, respectively (plant signal in solid line and neural signal in dashed line).

Fig. 11. Angular velocity identification for simulation in left and right wheels, respectively (plant signal in solid line and neural signal in dashed line).