Control Engineering Practice 19 (2011) 215–222

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

Autonomous mobile robots navigation using RBF neural compensator

Francisco G. Rossomando *, Carlos Soria, Ricardo Carelli

Instituto de Automática, Universidad Nacional de San Juan, Av. San Martín 1109 Oeste, 5400 San Juan, Argentina

ARTICLE INFO

Article history:
Received 26 October 2009
Accepted 30 November 2010
Available online 22 December 2010

Keywords:
Adaptive inverse control
MIMO system
Mobile robot control

ABSTRACT

This paper presents an approach to adaptive trajectory tracking of mobile robots which combines a feedback linearization based on a nominal model and a RBF-NN adaptive dynamic compensation. For a robot with uncertain dynamic parameters, two controllers are implemented separately: a kinematics controller and an inverse dynamics controller. The uncertainty in the nominal dynamics model is compensated by a neural adaptive feedback controller. The resulting adaptive controller is efficient and robust in the sense that it succeeds to achieve a good tracking performance with a small computational effort. The analysis of the RBF-NN approximation error on the control errors is included. Finally, the performance of the control system is verified through experiments.

1. Introduction

Real-time trajectory control of a mobile robot is a very important issue in mobile robotics. Due to slippage, disturbance, noise, robot-base interaction and sensor errors, it is very difficult to avoid the errors between the desired and the actual robot position. How to effectively control a mobile robot to precisely track a desired trajectory is still subject to active research in robotics.

Several studies have been published regarding the design of controllers to guide mobile robots during trajectory tracking. Some of the controllers designed so far are based only on the kinematics of the mobile robot, like the controllers presented in Künhe, Gomes, and Fetter (2005), Scaglia, Rosales, Quintero, Mut, and Agarwal (2010) and Wu, Chen, Wang, and Woo (1999). Other studies present the design of controllers that compensate for the robot dynamics. The work Das and Kar (2006) show an adaptive fuzzy logic-based controller where the system uncertainty, which includes mobile robot parameters variation and unknown nonlinearities, is estimated by a fuzzy logic system and its parameters are tuned on-line. In Bahita and Belarbi (2006) it has been proposed a direct adaptive controller for nonlinear systems using RBF-NN, but the results shown in this paper are just based on simulations and it is not applied to mobile robot.

In Fukao, Nakagawa, and Adachi (2000) it is presented the design of an adaptive trajectory tracking controller to generate torques based on a dynamic model whose parameters are unknown. In that work, only simulation results are shown. Other trajectory tracking controllers based on robot dynamics are developed in Bugeja and Fabri (2007), Liu, Zhang, Yang, and Yu (2004), Dong and Guo (2005) and Dong and Huo (1999), but the results are also based on simulations.

In Yang, Gu, Mita, and Hu (2004) it is presented a nonlinear tracking controller that is designed using a dynamic feedback linearization technique and implemented in a nonholonomic car-like mobile robot. Kim, Shin, and Lee (2000) propose a robust adaptive controller for a mobile robot, which is divided in two parts. The first one is based on robot kinematics and is responsible of generating references for the second one, which compensates for the modeled dynamics. However, the adapted parameters are not real parameters of the robot, and no experimental results are presented. Additionally, the control actions are given in terms of torques, while usual commercial robots accept velocity commands. In Carelli and De La Cruz (2006) it is proposed a linear parameterization of a unicycle-like mobile robot and the design of a trajectory tracking controller based on its complete model. One advantage of this controller is that its parameters are directly related to the robot parameters. However, if the parameters are not correctly identified or change with time due, for example, to load variation, the performance of the controller will be severely affected. The work Martins, Celeste, Carelli, Sarcinelli Filho, and Bastos Filho (2008) implements an adaptive dynamic controller for autonomous mobile robot trajectory tracking, with this methodology the parameters are updating on-line to compensate the dynamical variations. The theory of linear and nonlinear H_∞ is used in Hwang, Chen, and Chang (2004) and Inoue, Siqueira, and Terra (2009) to resolve the mobile robot tracking control. In Antonini, Ippoliti, and Longhi (2006), a Neural Network based control methodologies are further investigated within the context of multiple models control of mobile robots in an adaptive and learning control framework. A switching strategy among these models determines the best possible control input at any given instant. Radial basis function

* Corresponding author.
E-mail addresses: frosoma@inaut.unsj.edu.ar (F.G. Rossomando), csoria@inaut.unsj.edu.ar (C. Soria), rcarelli@inaut.unsj.edu.ar (R. Carelli).
URL: http://www.inaut.unsj.edu.ar (F.G. Rossomando).

0967-0661/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.conengprac.2010.11.011

networks have been used for identification and control in order to avoid the nonlinear optimization techniques used in the learning algorithm of the Multilayer Neural Networks (MNN). The work Chen, Li, and Yeh (2009) proposes a complete control law comprising an evolutionary programming based kinematic control (EPKC) and an adaptive fuzzy sliding-mode dynamic control (AFSMDC) for the trajectory-tracking control of nonholonomic wheeled mobile robots (WMRs). The control gains for kinematic control (KC) are trained by evolutionary programming (EP). The proposed AFSMDC not only eliminates the chattering phenomenon in the sliding-mode control, but also copes with the system uncertainties and external disturbances.

In this paper, the design of an adaptive trajectory tracking controller based on a nominal robot dynamics and a compensation neural controller is developed. The compensation neural controller has the capacity to learn the difference between the nominal and the actual dynamics of the robot. The whole control system is designed in two parts: one including a kinematics controller and another one with a dynamics controller, similar to Kim et al. (2000). As a realistic assumption, it is supposed that model uncertainties may appear in the robot dynamics alone. Therefore, the dynamic controller is designed based on a nominal model and a RBF-NN compensation controller with the capacity to learn the difference between the nominal and the actual robot dynamics. As the RBF-NN compensates only for a nominal model error dynamics, its computational cost is significantly reduced compared with a whole *NN* inverse dynamics controller. An analysis is done to study the effects of the RBF-NN approximation error on the control error when the whole control system, the kinematics and the adaptive dynamics controller working together, is applied to a tracking control task.

The paper is organized as follows: Section 2 presents a system overview and shows the mathematical representation of the complete unicycle-like robot model. The kinematics, dynamic and neural controllers are discussed, respectively, in Sections 3–5, and the corresponding error analysis is included in Section 6. Finally, Section 7 presents some experimental results to show the performance of the adaptive controller, and brief conclusions are given in Section 8.

2. Robot model

In this section, the dynamic model of the unicycle-like mobile robot presented in Fig. 1 is reviewed. Fig. 1 depicts the mobile robot, with the parameters and variables of interest. There, v and ω are, respectively, the linear and angular velocities developed by the robot, G is the center of mass of the robot, c is the position of the castor wheel, $\boldsymbol{h}$ is the point of interest with coordinates r_x, r_y in the workspace plane, ψ is the robot orientation relative to the r_x-axis, and a is the distance between the point of interest and the central point of the virtual axis linking the traction wheels. A coordinate system fixed to the robot is denoted as r'_x–r'_y.

The mathematical representation of the complete model (Carelli & De La Cruz, 2006) is given by

Kinematics model

$$\begin{bmatrix}\dot r_x\\ \dot r_y\\ \dot\psi\end{bmatrix}=\begin{bmatrix}\cos\psi & -a\sin\psi\\ \sin\psi & a\cos\psi\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix}+\begin{bmatrix}\delta_{rx}\\ \delta_{ry}\\ 0\end{bmatrix} \tag{1}$$

Dynamic model

$$\begin{bmatrix}\dot v\\ \dot\omega\end{bmatrix}=\begin{bmatrix}\frac{\vartheta_3}{\vartheta_1}\omega^2-\frac{\vartheta_4}{\vartheta_1}v\\ -\frac{\vartheta_5}{\vartheta_2}v\omega-\frac{\vartheta_6}{\vartheta_2}\omega\end{bmatrix}+\begin{bmatrix}\frac{1}{\vartheta_1} & 0\\ 0 & \frac{1}{\vartheta_2}\end{bmatrix}\begin{bmatrix}v_{ref}\\ \omega_{ref}\end{bmatrix}+\begin{bmatrix}\delta_v\\ \delta_\omega\end{bmatrix} \tag{2}$$

where v_{ref} and ω_{ref} are the linear and angular velocity inputs or control actions.

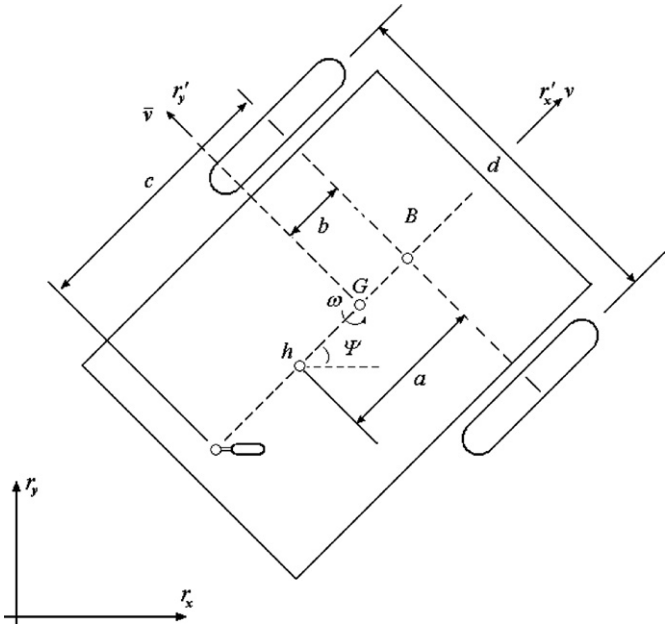

Fig. 1. Parameters of the unicycle-like mobile robot.

The vector of identified parameters and the vector of uncertainties parameters associated to the mobile robot are, respectively,

$$\vartheta=\begin{bmatrix}\vartheta_1 & \vartheta_2 & \vartheta_3 & \vartheta_4 & \vartheta_5 & \vartheta_6\end{bmatrix}^T$$
$$\boldsymbol{\delta}=\begin{bmatrix}\delta_{rx} & \delta_{ry} & 0 & \delta_v & \delta_\omega\end{bmatrix}^T \tag{3}$$

where δ_{rx} and δ_{ry} are functions of slip velocities and robot orientation, δ_v and δ_ω are functions of physical parameters as mass, inertia, wheel and tires diameters, motor and its servos parameters, forces on the wheels, and others. These are considered as disturbances.

The robot's model is split in a kinematics and a dynamic part as shown in Eqs. (1) and (2), respectively, which is also represented in Fig. 2. Therefore, two controllers are implemented, based on feedback linearization, for both the kinematics and dynamic models of the robot.

Using Euler, (1) and (2) are discretized with T_0=0.1s, by the approximation $\dot v=(v(k+1)-v(k))/T_0$, $\dot\omega=(\omega(k+1)-\omega(k))/T_0$ and in the same way with $\dot r_x,\dot r_y,\dot\psi$:

Discrete kinematics model

$$\begin{bmatrix}r_x(k+1)\\ r_y(k+1)\\ \psi(k+1)\end{bmatrix}=T_0\begin{bmatrix}\cos\psi(k) & -a\sin\psi(k)\\ \sin\psi(k) & a\cos\psi(k)\\ 0 & 1\end{bmatrix}\begin{bmatrix}v(k)\\ \omega(k)\end{bmatrix}+\begin{bmatrix}r_x(k)\\ r_y(k)\\ \psi(k)\end{bmatrix}+\begin{bmatrix}\delta_{rx}\\ \delta_{ry}\\ 0\end{bmatrix} \tag{4}$$

Discrete dynamic model

$$\begin{bmatrix}v(k+1)\\ \omega(k+1)\end{bmatrix}=\begin{bmatrix}\frac{\Omega_3}{\Omega_1}\omega^2(k)+\frac{\Omega_4}{\Omega_1}v(k)\\ -\frac{\Omega_5}{\Omega_2}v(k)\omega(k)+\frac{\Omega_6}{\Omega_2}\omega(k)\end{bmatrix}+\begin{bmatrix}\frac{1}{\Omega_1} & 0\\ 0 & \frac{1}{\Omega_2}\end{bmatrix}\begin{bmatrix}v_{ref}(k)\\ \omega_{ref}(k)\end{bmatrix}+\begin{bmatrix}\delta_v\\ \delta_\omega\end{bmatrix} \tag{5}$$

where

$$\frac{\Omega_3}{\Omega_1}=\left(\frac{\vartheta_3}{\vartheta_1}T_0\right),\quad \frac{\Omega_4}{\Omega_1}=\left(1-\frac{\vartheta_4}{\vartheta_1}T_0\right),\quad \frac{1}{\Omega_1}=\left(\frac{1}{\vartheta_1}T_0\right)$$

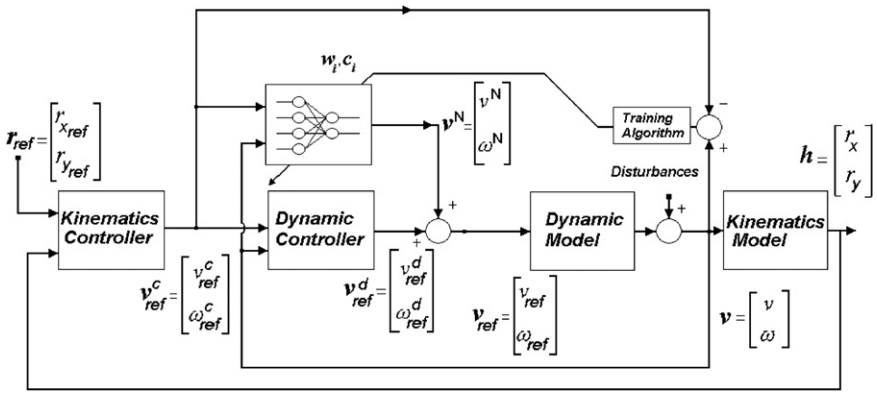

Fig. 2. Adaptive control structure.

$$\frac{\Omega_5}{\Omega_2}=\left(-\frac{\vartheta_5}{\vartheta_2}T_0\right),\quad \frac{\Omega_6}{\Omega_2}=\left(1-\frac{\vartheta_6}{\vartheta_2}T_0\right),\quad \frac{1}{\Omega_2}=\left(\frac{1}{\vartheta_2}T_0\right) \tag{6}$$

The parameters have been identified for a Pioneer DX2 mobile robot (Carelli & De La Cruz, 2006) obtaining: $\vartheta_1=0.3037$ s; $\vartheta_2=0.2768$ s; $\vartheta_3=-0.0004018$ s m/rad^2; $\vartheta_4=0.9835$; $\vartheta_5=-0.003818$ s/m; $\vartheta_6=1.0725$; $T_0=0.1$ s.

3. Kinematics controller

The design of the kinematics controller is based on the robot's kinematics model. The proposed kinematics controller is given by

$$\begin{bmatrix} v^c_{ref}(k)\\ \omega^c_{ref}(k)\end{bmatrix}=\begin{bmatrix} \frac{\cos\psi(k)}{T_0} & \frac{\sin\psi(k)}{T_0}\\ -\frac{1}{a}\frac{\sin\psi(k)}{T_0} & \frac{1}{a}\frac{\cos\psi(k)}{T_0}\end{bmatrix}\begin{bmatrix} r_{xref}(k+1)+l_x\tanh\left(\frac{k_x}{l_x}\tilde r_x(k)\right)\\ r_{yref}(k+1)+l_y\tanh\left(\frac{k_y}{l_y}\tilde r_y(k)\right)\end{bmatrix}-\begin{bmatrix} r_x(k)\\ r_y(k)\end{bmatrix} \tag{7}$$

where $\tilde r_x(k)=r_{xref}(k)-r_x(k)$, $\tilde r_y(k)=r_{yref}(k)-r_y(k)$ are the position errors and the tanh(.) function has been introduced to avoid saturation of the control actions due to large position errors. By replacing (7) in the upper part of (4) under an assumption of perfect velocity tracking, $v^c_{ref}(k)\equiv v(k)$, $\omega^c_{ref}(k)\equiv\omega(k)$, the closed-loop equation is

$$\begin{bmatrix}\tilde r_x(k+1)\\ \tilde r_y(k+1)\end{bmatrix}+\begin{bmatrix} l_x & 0\\ 0 & l_y\end{bmatrix}\begin{bmatrix}\tanh\left(\frac{k_x}{l_x}\tilde r_x(k)\right)\\ \tanh\left(\frac{k_y}{l_y}\tilde r_y(k)\right)\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix} \tag{8}$$

By defining the position error vector $\tilde{\mathbf{h}}(k)=\begin{bmatrix}\tilde r_x(k) & \tilde r_y(k)\end{bmatrix}^T$, (8) can be now written as

$$\tilde{\mathbf{h}}(k+1)=\begin{bmatrix} l_x\tanh\left(\frac{k_x}{l_x}\tilde r_x(k)\right) & l_y\tanh\left(\frac{k_y}{l_y}\tilde r_y(k)\right)\end{bmatrix}^T \tag{9}$$

By taking a Lyapunov candidate $V(k)=\tilde{\mathbf{h}}^T(k)\tilde{\mathbf{h}}(k)$, and for $k_x,k_y<1$, $k_x/l_x<1$ and $k_y/l_y<1$, then $\tilde{\mathbf{h}}(k)\to 0$ for $k\to\infty$. The perfect velocity tracking assumption will be relaxed when analyzing the stability of the whole control system.

4. Dynamic controller

The dynamic controller receives the references of linear and angular velocities generated by the kinematics controller, and produces another pair of linear and angular velocities commands to be sent to the robot servos, as shown in Fig. 2.

First, a dynamic controller is designed based on a nominal robot dynamics, which represents an estimated mean dynamics of the robot. The inverse robot dynamics from (5) without considering the uncertainties can be parameterized as follows:

$$\begin{bmatrix} v_{ref}(k)\\ \omega_{ref}(k)\end{bmatrix}=\begin{bmatrix} v(k+1) & 0 & -\omega^2(k) & v(k) & 0 & 0\\ 0 & \omega(k+1) & 0 & 0 & v(k)\omega(k) & \omega(k)\end{bmatrix}\mathbf{\Omega} \tag{10}$$

with

$$\mathbf{\Omega}=\begin{bmatrix}\Omega_1 & \Omega_2 & \Omega_3 & \Omega_4 & \Omega_5 & \Omega_6\end{bmatrix}^T \tag{11}$$

Eq. (10) can be rewritten as

$$\begin{bmatrix} v_{ref}(k)\\ \omega_{ref}(k)\end{bmatrix}=\begin{bmatrix}\Omega_1 & 0\\ 0 & \Omega_2\end{bmatrix}\begin{bmatrix} v(k+1)\\ \omega(k+1)\end{bmatrix}+\begin{bmatrix} 0 & 0 & -\omega^2(k) & v(k) & 0 & 0\\ 0 & 0 & 0 & 0 & v(k)\omega(k) & \omega(k)\end{bmatrix}\mathbf{\Omega} \tag{12}$$

or in compact form

$$\mathbf{v}_{ref}(k)=\mathbf{D}\mathbf{v}(k+1)+\boldsymbol{\eta}(k) \tag{13}$$

where

$$\mathbf{v}_{ref}(k)=\begin{bmatrix} v_{ref}(k) & \omega_{ref}(k)\end{bmatrix}^T,\quad \mathbf{v}(k)=\begin{bmatrix} v(k) & \omega(k)\end{bmatrix}^T$$

$$\boldsymbol{\eta}=\begin{bmatrix} 0 & 0 & -\omega^2(k) & v(k) & 0 & 0\\ 0 & 0 & 0 & 0 & v(k)\omega(k) & \omega(k)\end{bmatrix}\mathbf{\Omega}$$

and $\boldsymbol{D}=diag(\Omega_1,\Omega_2)$.

The proposed inverse dynamics control law is

$$\mathbf{v}^d_{ref}(k)=\mathbf{G}(v(k),\omega(k),v^c_{ref}(k),\omega^c_{ref}(k),v^c_{ref}(k+1),\omega^c_{ref}(k+1))\mathbf{\Omega} \tag{14}$$

where

$$\mathbf{G}=\begin{bmatrix}\sigma_1(k) & 0 & -\omega^2(k) & v(k) & 0 & 0\\ 0 & \sigma_2(k) & 0 & 0 & v(k)\omega(k) & \omega(k)\end{bmatrix}$$

$$\sigma_1(k)=v^c_{ref}(k+1)+k_v(v^c_{ref}(k)-v(k))$$

$$\sigma_2(k)=\omega^c_{ref}(k+1)+k_\omega(\omega^c_{ref}(k)-\omega(k)) \tag{15}$$

In matrix form:

$$\boldsymbol{\sigma}(k)=\mathbf{v}^c_{ref}(k+1)+\mathbf{K}(\mathbf{v}^c_{ref}(k)-\mathbf{v}(k)) \tag{16}$$

where $\boldsymbol{\sigma}(k)=\begin{bmatrix}\sigma_1(k) & \sigma_2(k)\end{bmatrix}^T$, $\boldsymbol{K}=diag(k_v,k_\omega)$

Similarly to (13), the dynamic control law can now be expressed as

$$\mathbf{v}^d_{ref}(k)=\mathbf{D}\boldsymbol{\sigma}(k)+\boldsymbol{\eta}(k) \tag{17}$$

In (14), $\mathbf{\Omega}$ is a parameter vector, which is constant. Due to uncertainties in the nominal model (parameter errors $\tilde{\mathbf{\Omega}}$ and non-modeled dynamics $\boldsymbol{\delta}$) the use of a RBF-NN compensation is proposed. Now the complete control law may be expressed by

$$[v_{ref}(k),\omega_{ref}(k)]^T=[\mathbf{D}\boldsymbol{\sigma}(k)+\boldsymbol{\eta}(k)] +\mathbf{v}_N(v(k),\omega(k),v^c_{ref}(k),\omega^c_{ref}(k),v^c_{ref}(k+1),\omega^c_{ref}(k+1))$$

$$\mathbf{v}_{ref}(k)=\mathbf{v}^d_{ref}(k)+\mathbf{v}_N(k) \tag{18}$$

where $\mathbf{v}^c_{ref}(k)=[v^c_{ref}(k),\omega^c_{ref}(k)]^T$ is the output of the kinematics controller; $\mathbf{v}^d_{ref}(k)=[v^d_{ref}(k),\omega^d_{ref}(k)]^T$ is the output of the dynamic controller; $\boldsymbol{v}_N(k)=[v_N(k),\omega_N(k)]^T$ is the output of the *RBF-NN* compensator, which learns the difference between the nominal and the actual dynamics of the mobile robot. The complete proposed control structure is shown in Fig. 2.

5. Neural adaptive controller

A neural compensation controller can be obtained using current and past information of the system output vector $\boldsymbol{v}_N(k)$, and control input vector $\boldsymbol{v}$. Since most NNs are implemented by digital computers, the dynamic part of a nonlinear system is then obtained by means of time delay operations. So RBF neural network (RBF-NN) implemented in a digital computer can be considered as a time delay discrete time control system (Haykin, 2001). This structure is shown in Figs. 2 and 3. Fig. 3 shows the structure of a RBF-NN controller, where z^{-1} is the backward shift operator, $\boldsymbol{v}^c_{ref}(k+1)$ is the desired system output.

In Fig. 3, t_a, t_b are integers that relate to the system order. The *RBF* vector is denoted by ξ; the input pattern vector is $\boldsymbol{S}(k)$; the hidden-to-output weight matrix is $\mathbf{w}=\begin{bmatrix}\mathbf{w}_v & \mathbf{w}_\omega\end{bmatrix}$ and M is *RBF* neurons number.

The output $\boldsymbol{v}_N(k)$ of neural compensation is the system control signal. Define vectors

$$\mathbf{S}(k)=[\mathbf{v}_N(k-1),\ldots,\mathbf{v}_N(k-t_a+1),\mathbf{v}(k), \mathbf{v}(k-1),\ldots,\mathbf{v}(k-t_b+1),\mathbf{v}^c_{ref}(k+1)] \tag{19}$$

where $\mathbf{v}^c_{ref}(k)=[v^c_{ref}(k),\omega^c_{ref}(k)]^T$, $\boldsymbol{v}_N=[v_N(k),\omega_N(k)]^T$ and $\boldsymbol{v}=[v(k),\omega(k)]^T$

$$\xi_i(\mathbf{S}(k))=\varphi(\mathbf{S}(k)-\mathbf{c}_i) \tag{20}$$

Here, φ is a nonlinear exp activation function,

$$\varphi(\mathbf{S}(k)-\mathbf{c}_i)=\exp\left(\frac{||\mathbf{S}(k)-\mathbf{c}_i||^2}{2.\Gamma^2}\right)$$

where $\boldsymbol{c}_i$ is the center of $\varphi(.)$, Γ is a constant of function $\varphi(.)$, in this case is equal to 1.

The neural output control action can be expressed by

$$\mathbf{v}_N(k)=\mathbf{w}^T(k)\xi(\mathbf{S}(k)) \tag{21}$$

The controller weights are adapted to minimize the cost function

$$J(k)=\frac{1}{2}(\mathbf{e}^T(k)\mathbf{e}(k))=\frac{1}{2}(e^2_v(k)+e^2_\omega(k)) \tag{22}$$

where $\boldsymbol{e}(k)=[e_v(k),e_\omega(k)]^T$, and $e_v(k)=v^c_{ref}(k)-v(k)$, $e_\omega(k)=\omega^c_{ref}(k)-\omega(k)$.

The correction applied to a synaptic weight is proportional to the negative gradient of J:

$$\Delta\mathbf{w}(k)=-\eta\frac{\partial J(k)}{\partial\mathbf{w}(k)} \tag{23}$$

According to the chain rule method, this gradient is expressed as

$$\Delta\mathbf{w}=-\eta\frac{\partial J}{\partial\mathbf{v}}\frac{\partial\mathbf{v}}{\partial\mathbf{v}_N}\frac{\partial\mathbf{v}_N}{\partial\mathbf{w}} \tag{24}$$

For the centers of *RBF-NN*

$$\Delta\mathbf{c}_i=-\eta\frac{\partial J}{\partial\mathbf{v}}\frac{\partial\mathbf{v}}{\partial\mathbf{v}_N}\frac{\partial\mathbf{v}_N}{\partial\xi}\frac{\partial\xi}{\partial\mathbf{c}_i} \tag{25}$$

Hence to effectively use these gradients, it is needed to know $\partial v/\partial v_N$, which is difficult to calculate when the system model is unknown. However, an approximation may be used:

$$\frac{\partial\mathbf{v}}{\partial\mathbf{v}_N}=sign\left(\frac{\Delta\mathbf{v}}{\Delta\mathbf{v}_N}\right)$$

The tuning of *RBF-NN* weights and centers can be described by

$$\Delta\mathbf{w}(k)=-\eta.\mathbf{e}(k)sign\left(\frac{\Delta\mathbf{v}}{\Delta\mathbf{v}_N}\right)\xi(\mathbf{S}(k))$$
$$\Delta\mathbf{c}_i(k)=-\eta.\mathbf{e}(k)sign\left(\frac{\Delta\mathbf{v}}{\Delta\mathbf{v}_N}\right)\mathbf{w}^T\xi(\mathbf{S}(k))||\mathbf{S}(k)-\mathbf{c}_i(k)|| \tag{26}$$

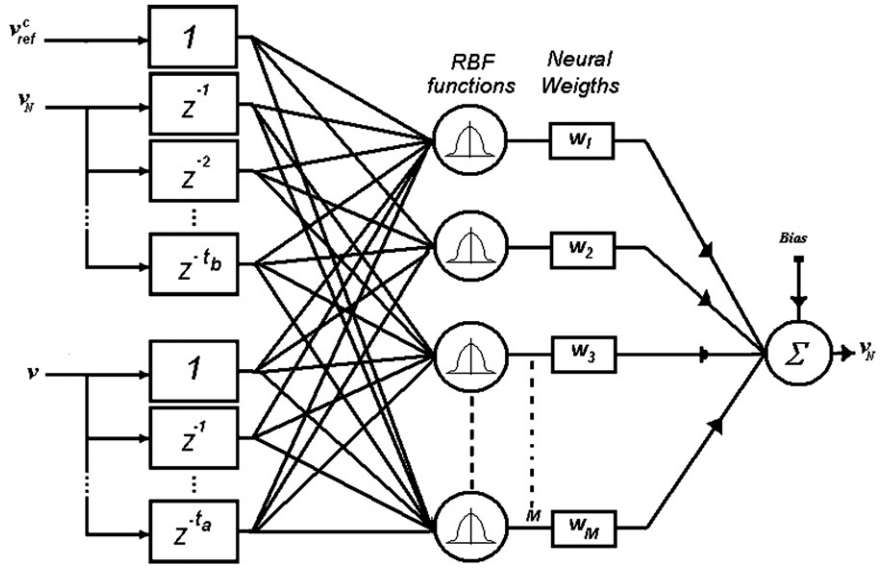

Fig. 3. Dynamic neural net.

The *RBF-NN* compensation controller can be trained by the method presented in the preceding equations and it is expected to provide a nonlinear mapping expressed by (20) and (21). For the desired output $\boldsymbol{v}^{c}_{\boldsymbol{ref}}(k+1)$, the controller will produce the signal $\boldsymbol{v}_N(k)$:

$$\mathbf{v}_N(k) = NN[\mathbf{v}_N(k-1),\ldots,\mathbf{v}_{N,}(k-t_a+1),\mathbf{v}(k),\\ \mathbf{v}(k-1),\ldots,\mathbf{v}(k-t_b+1),\mathbf{v}^c_{ref}(k+1)$$

The algorithm of this neural compensation scheme is described by the following steps:

(a) Initialize randomly weights and centers of RBF neural compensation controller $w_i(0)$, c_i.
(b) Provide neural compensation input pattern at time k:

$$\mathbf{S}(k) = [\mathbf{v}_N(k-1),\ldots,\mathbf{v}_N(k-t_a+1),\mathbf{v}(k),\\ \mathbf{v}(k-1),\ldots,\mathbf{v}(k-t_b+1),\mathbf{v}^c_{ref}(k+1)]$$

(c) Compute the output of the neural compensation controller (21), as the system control input at time k.
(d) Take the output $\mathbf{v}_N(k+1)$ and add $\mathbf{v}_{ref}(k+1)$ at time $k+1$ of the process model, then compute the system output error: $\boldsymbol{e}(k+1)=\boldsymbol{v}^{c}_{\boldsymbol{ref}}(k+1)-\boldsymbol{v}(k+1)$; if $|\boldsymbol{e}(k+1)|<\varepsilon$ (where $\varepsilon>0$ is a given constant) then go to (e); otherwise continue.
(e) Update the centers and the weights by (26).
(f) Let $k=k+1$, return to (b) for the next step.

6. Stability analysis

The inverse dynamics control law from (14) considering the parametric errors $\tilde{\boldsymbol{\Omega}}$ can be expressed as

$$\mathbf{v}^d_{ref}(k) = \mathbf{G}(k)\boldsymbol{\Omega} + \mathbf{G}(k)\tilde{\boldsymbol{\Omega}} \tag{27}$$

The complete control law including the neural compensation (18) can now be expressed by

$$\mathbf{v}_{ref}(k) = \mathbf{D}\boldsymbol{\sigma}(k) + \mathbf{G}(k)\tilde{\boldsymbol{\Omega}} + \boldsymbol{\eta} + \mathbf{v}_N(k) \tag{28}$$

From (13), the dynamic model including the uncertainties is

$$\mathbf{v}_{ref}(k) = \mathbf{D}\mathbf{v}(k+1) + \boldsymbol{\eta} + \boldsymbol{\delta} \tag{29}$$

Now, by equating (28) and (29), the closed loop system results:

$$\mathbf{D}\mathbf{v}(k+1) + \boldsymbol{\eta} + \boldsymbol{\delta} = \mathbf{D}\boldsymbol{\sigma}(k) + \mathbf{G}(k)\tilde{\boldsymbol{\Omega}} + \boldsymbol{\eta} + \mathbf{v}_N(k) \tag{30}$$

Rearranging

$$\mathbf{D}(\mathbf{v}(k+1) - \boldsymbol{\sigma}(k)) - (\mathbf{G}(k)\tilde{\boldsymbol{\Omega}} - \boldsymbol{\delta}) = \mathbf{v}_N(k) \tag{31}$$

Recalling the velocity control error:

$$\mathbf{e}(k) = \mathbf{v}^c_{ref}(k) - \mathbf{v}(k) \tag{32}$$

and the definition of $\boldsymbol{\sigma}$ in (16), Eq. (31) can be rewritten as

$$-\mathbf{D}(\mathbf{e}(k+1) + \mathbf{K}\mathbf{e}(k)) - (\mathbf{G}(k)\tilde{\boldsymbol{\Omega}} - \boldsymbol{\delta}) = \mathbf{v}_N(k) \tag{33}$$

Neural compensation tends to make the control error equal to zero by compensating the model error and the uncertainties, but it introduces its own compensation error expressed as $\mathbf{v}_N(k) = -\mathbf{w}^T(k)\xi(\mathbf{S}(k)) - \tilde{\mathbf{w}}^T(k)\xi(\mathbf{S}(k))$, with $\tilde{\mathbf{w}}(k) = \mathbf{w}^* - \mathbf{w}(k)$. It is assumed that there exists an unknown constant ideal weight vector $\boldsymbol{w}^*$ whose estimate $\boldsymbol{w}(k)$ is, and that the NN approximation errors $\tilde{\mathbf{w}}(k)$ are bounded by

$$||\tilde{\mathbf{w}}(k)|| \le \kappa_{max}$$

This way, (33) results as

$$-\mathbf{D}(\mathbf{e}(k+1) + \mathbf{K}\mathbf{e}(k)) - (\mathbf{G}(k)\tilde{\boldsymbol{\Omega}} - \boldsymbol{\delta}) = \tilde{\mathbf{w}}^T(k)\xi(\mathbf{S}(k)) + \mathbf{w}^T(k)\xi(\mathbf{S}(k)) \tag{34}$$

The ideal compensation is

$$-\mathbf{G}(k)\tilde{\boldsymbol{\Omega}} + \boldsymbol{\delta} = \mathbf{w}^T(k)\xi(\mathbf{S}(k)) \tag{35}$$

Now, rewriting (34), in the closed loop equation remains the neural network approximation error

$$\mathbf{e}(k+1) = -\mathbf{K}\mathbf{e}(k) - \mathbf{D}^{-1}(\tilde{\mathbf{w}}^T(k)\cdot\xi(\mathbf{S}(k))) \tag{36}$$

This error equation can be expressed in terms of their components:

$$\begin{cases} e_v(k+1) = -k_v e_v(k) - \Omega_1^{-1}\tilde{\mathbf{w}}_v^T(k)\xi(S(k)) \\ e_\omega(k+1) = -k_\omega e_\omega(k) - \Omega_2^{-1}\tilde{\mathbf{w}}_{\boldsymbol{\omega}}^T(k)\xi(S(k)) \end{cases} \tag{37}$$

Let us now consider the Lyapunov candidate function

$$V(k) = e_v^2(k) + e_\omega^2(k) + \tilde{\mathbf{w}}_v^T(k)\tilde{\mathbf{w}}_v(k) + \tilde{\mathbf{w}}_{\boldsymbol{\omega}}^T(k)\tilde{\mathbf{w}}_{\boldsymbol{\omega}}(k) \tag{38}$$

and the Lyapunov's difference equation:

$$\begin{aligned}\Delta V(k) = {} & e_v^2(k+1) + e_\omega^2(k+1) + \tilde{\mathbf{w}}_v^T(k+1)\tilde{\mathbf{w}}_v(k+1) \\ & + \tilde{\mathbf{w}}_{\boldsymbol{\omega}}^T(k+1)\tilde{\mathbf{w}}_{\boldsymbol{\omega}}(k+1) - e_v^2(k) - e_\omega^2(k) - \tilde{\mathbf{w}}_v^T(k)\tilde{\mathbf{w}}_v(k) \\ & - \tilde{\mathbf{w}}_{\boldsymbol{\omega}}^T(k)\tilde{\mathbf{w}}_{\boldsymbol{\omega}}(k)\end{aligned} \tag{39}$$

Replacing (37) in (39)

$$\begin{aligned}\Delta V(k) = {} & [k_v^2 e_v^2(k) + 2e_v(k)k_v\Omega_1^{-1}\tilde{\mathbf{w}}_v^T\xi(S(k)) + \rho_v] + [k_\omega^2 e_\omega^2(k) \\ & + 2e_\omega(k)k_\omega\Omega_2^{-1}\tilde{\mathbf{w}}_{\boldsymbol{\omega}}^T\xi(S(k)) + \rho_\omega] + \tilde{\mathbf{w}}_v^T(k+1)\tilde{\mathbf{w}}_v(k+1) \\ & + \tilde{\mathbf{w}}_{\boldsymbol{\omega}}^T(k+1)\tilde{\mathbf{w}}_{\boldsymbol{\omega}}(k+1) - e_v^2(k) - e_\omega^2(k) - \tilde{\mathbf{w}}_v^T(k)\tilde{\mathbf{w}}_v(k) \\ & - \tilde{\mathbf{w}}_{\boldsymbol{\omega}}^T(k)\tilde{\mathbf{w}}_{\boldsymbol{\omega}}(k)\end{aligned} \tag{40}$$

The terms ρ_v and ρ_ω given as follows, do not depend on the control error:

$$\begin{aligned}\rho_v &= \xi^T(S(k))\tilde{\mathbf{w}}_v\Omega_1^- 2\tilde{\mathbf{w}}_v^T\xi(S(k)) \\ \rho_\omega &= \xi^T(S(k))\tilde{\mathbf{w}}_{\boldsymbol{\omega}}\Omega_2^- 2\tilde{\mathbf{w}}_{\boldsymbol{\omega}}^T\xi(S(k))\end{aligned} \tag{41}$$

By taking $k_{v,\omega} < 1$ then,

$$\mathbf{Q} = \mathbf{K}^T\mathbf{K} - \mathbf{I} < 0 \tag{42}$$

where $\boldsymbol{Q}=diag[q_v,q_\omega]$. Also, $\tilde{\mathbf{w}}_v^T(k+1)\tilde{\mathbf{w}}_v(k+1)$ can be expressed from (26) as

$$\begin{aligned}\tilde{\mathbf{w}}_v^T(k+1)\tilde{\mathbf{w}}_v^T(k+1) &= [\tilde{\mathbf{w}}_v(k) - \eta e_v(k)\xi(\mathbf{S}(k))]^T.[\tilde{\mathbf{w}}_v(k) - \eta e_v(k)\xi(\mathbf{S}(k))] \\ &= \cdots = \tilde{\mathbf{w}}_v^T(k)\tilde{\mathbf{w}}_v(k) - 2\eta e_v(k)\tilde{\mathbf{w}}_v^T(k)\xi(\mathbf{S}(k)) \\ &\quad + (-\eta e_v(k)\xi(\mathbf{S}(k)))^T(-\eta e_v(k)\xi(\mathbf{S}(k)))\end{aligned} \tag{43}$$

Similarly for the product $\tilde{\mathbf{w}}_{\boldsymbol{\omega}}^T(k+1)\tilde{\mathbf{w}}_{\boldsymbol{\omega}}(k+1)$ and making, it follows that (40) can be now written as

$$\begin{aligned}\Delta V(k) = {} & [-q_v e_v^2(k) + 2e_v(k)k_v\Omega_1^{-1}\tilde{\mathbf{w}}_v^T\xi(S(k)) + \rho_v] \\ & + [-q_\omega e_\omega^2(k) + 2e_\omega(k)k_\omega\Omega_2^{-1}\tilde{\mathbf{w}}_{\boldsymbol{\omega}}^T\xi(S(k)) + \rho_\omega] \\ & - 2\eta e_v(k)\tilde{\mathbf{w}}_v^T(k)\xi(\mathbf{S}(k)) + (-\eta e_v(k)\xi(\mathbf{S}(k)))^T(-\eta e_v(k)\xi(\mathbf{S}(k))) \\ & - 2\eta e_\omega(k)\tilde{\mathbf{w}}_{\boldsymbol{\omega}}^T(k)\xi(\mathbf{S}(k)) + (-\eta e_\omega(k)\xi(\mathbf{S}(k)))^T(-\eta e_\omega(k)\xi(\mathbf{S}(k)))\end{aligned} \tag{44}$$

A sufficient condition for (44) to be negative is

$$\begin{cases} q_v e_v^2(k) \ge [2e_v(k)k_v\Omega_1^{-1}\tilde{\mathbf{w}}_v^T\xi(S(k)) + \rho_v] - 2\eta e_v(k)\tilde{\mathbf{w}}_v^T(k)\xi(\mathbf{S}(k)) \\ \qquad + (-\eta e_v(k)\xi(\mathbf{S}(k)))^T(-\eta e_v(k)\xi(\mathbf{S}(k))) \\ q_\omega e_\omega^2(k) \ge [2e_\omega(k)k_\omega\Omega_2^{-1}\tilde{\mathbf{w}}_{\boldsymbol{\omega}}^T\xi(S(k)) + \rho_\omega] - 2\eta e_\omega(k)\tilde{\mathbf{w}}_{\boldsymbol{\omega}}^T(k)\xi(\mathbf{S}(k)) \\ \qquad + (-\eta e_\omega(k)\xi(\mathbf{S}(k)))^T(-\eta e_\omega(k)\xi(\mathbf{S}(k))) \end{cases} \tag{45}$$

Applying norm and replacing η by making

$$\begin{aligned} k_v^T \Omega_1^{-1} &= \eta \\ k_\omega^T \Omega_2^{-1} &= \eta \end{aligned} \tag{46}$$

Eq. (45) can be written as

$$\begin{cases} q_v e_v^2(k) \geq |\rho_v| + \eta^2 ||\xi(\mathbf{S}(k))||^2 e_v^2(k) \\ q_\omega e_\omega^2(k) \geq |\rho_\omega| + \eta^2 ||\xi(\mathbf{S}(k))||^2 e_\omega^2(k) \end{cases} \tag{47}$$

Due to the boundedness of $\xi(\boldsymbol{S}(k))$ components by 1, and replacing $\rho_{v,\omega}$ from (41), condition (47) for negative ΔV implies that $e_{v,\omega}(k)$ is ultimately bounded by

$$\begin{cases} |e_v(k)| \leq \Omega_1^{-1} \sqrt{\frac{\sum_{i=1}^{M} |\tilde{\mathbf{w}}_{vi}(k)|}{q_v - \eta^2}} = \delta_{e1} \\ |e_\omega(k)| \leq \Omega_2^{-1} \sqrt{\frac{\sum_{i=1}^{M} |\tilde{\mathbf{w}}_{\omega i}(k)|}{q_\omega - \eta^2}} = \delta_{e2} \end{cases} \tag{48}$$

That is, the error $\boldsymbol{e}(k) \to B_{\delta e}$ with $B_{\delta e}$ a ball of size:

$$\delta_e = \max(\delta_{e1}, \delta_{e2})$$

Now, go back to the control error analysis that is the behavior of the trajectory control errors $\tilde{\mathbf{h}}(k)$. By relaxing the prefect velocity tracking assumption of the kinematics controllers in Section 3, Eq. (8) is now written as

$$\begin{bmatrix} \tilde{r}_x(k+1) \\ \tilde{r}_y(k+1) \end{bmatrix} + \begin{bmatrix} l_x & 0 \\ 0 & l_y \end{bmatrix} \begin{bmatrix} \tanh\left(\frac{k_x}{l_x} \tilde{r}_x(k)\right) \\ \tanh\left(\frac{k_y}{l_y} \tilde{r}_y(k)\right) \end{bmatrix} = \begin{bmatrix} \varepsilon_1(k) \\ \varepsilon_2(k) \end{bmatrix} \tag{49}$$

where the error vector $\boldsymbol{\varepsilon}(k) = \begin{bmatrix} \varepsilon_1(k) & \varepsilon_2(k) \end{bmatrix}^T$ is a function of the velocity tracking error and defined as $\mathbf{He}(k)$.

$$\boldsymbol{\varepsilon}(k) = \begin{bmatrix} \varepsilon_1(k) \\ \varepsilon_2(k) \end{bmatrix} = T_0 \begin{bmatrix} \cos\psi(k) & -a\sin\psi(k) \\ \sin\psi(k) & a\cos\psi(k) \end{bmatrix} \begin{bmatrix} e_v(k) \\ e_\omega(k) \end{bmatrix} = \mathbf{He}(k)$$

Rewriting (49), for small values of control error, $L(\tilde{\mathbf{h}}(k)) \approx \mathbf{K}_{xy} \tilde{\mathbf{h}}(k)$ with $\boldsymbol{K}_{xy} = diag(k_x, k_y)$

$$\tilde{\mathbf{h}}(k+1) + \mathbf{K}_{xy} \tilde{\mathbf{h}}(k) = \mathbf{He}(k) \tag{50}$$

Considering the Lyapunov candidate function

$$V(k) = \frac{1}{2} \tilde{\mathbf{h}}^T(k) \tilde{\mathbf{h}}(k) > 0 \tag{51}$$

Its discrete difference is

$$\begin{aligned} \Delta V(k) &= V(k+1) - V(k) = \tilde{\mathbf{h}}^T(k+1) \tilde{\mathbf{h}}(k+1) - \tilde{\mathbf{h}}^T(k) \tilde{\mathbf{h}}(k) \\ &= \mathbf{e}^T(k) \mathbf{H}^T \mathbf{He}(k) - 2\mathbf{e}^T(k) \mathbf{H}^T \mathbf{K}_{xy} \tilde{\mathbf{h}}(k) \\ &\quad + \tilde{\mathbf{h}}^T(k) \mathbf{K}_{xy}^T \mathbf{K}_{xy} \tilde{\mathbf{h}}(k) - \tilde{\mathbf{h}}^T(k) \tilde{\mathbf{h}}(k) \end{aligned} \tag{52}$$

A sufficient condition for ΔV to be negative is

$$||\mathbf{I} - \mathbf{K}_{xy}^2|| ||\tilde{\mathbf{h}}(k)||^2 \geq ||\mathbf{H}||^2 ||\mathbf{e}(k)||^2 + 2||\mathbf{H}|| \, ||\mathbf{K}_{xy}|| \, ||\mathbf{e}(k)|| \, ||\tilde{\mathbf{h}}(k)|| \tag{53}$$

Then, calculating the square roots,

$$\begin{aligned} ||\tilde{\mathbf{h}}|| &> \frac{-2||\mathbf{H}|| \, ||\mathbf{K}_{xy}|| \, ||\mathbf{e}(k)|| \pm \sqrt{4||\mathbf{H}||^2 ||\mathbf{K}_{xy}||^2 ||\mathbf{e}(k)||^2 + 4||\mathbf{e}(k)||^2 ||\mathbf{H}||^2}}{2||\mathbf{I} - \mathbf{K}_{xy}^2||} \\ &= \frac{-2||\mathbf{H}|| \, ||\mathbf{K}_{xy}|| \, ||\mathbf{e}(k)|| \pm 2||\mathbf{H}|| \, ||\mathbf{e}(k)||}{2||\mathbf{I} - \mathbf{K}_{xy}^2||} \end{aligned} \tag{54}$$

This condition for $\Delta V < 0$ implies that $\tilde{\mathbf{h}}$ is ultimately bounded by

$$||\tilde{\mathbf{h}}|| \leq \frac{(1 - ||\mathbf{K}_{xy}||) ||\mathbf{H}|| \, ||\mathbf{e}(k)||}{||\mathbf{I} - \mathbf{K}_{xy}{}^2||}$$

For large errors:

$$L(\tilde{\mathbf{h}}(k)) = \begin{bmatrix} l_x \tanh\left(\frac{k_x}{l_x} \tilde{r}_x(k)\right) \\ l_y \tanh\left(\frac{k_y}{l_y} \tilde{r}_y(k)\right) \end{bmatrix} = \begin{bmatrix} l_x \\ l_y \end{bmatrix} \tag{55}$$

Now considering (51) and (55) with $L = \begin{bmatrix} l_x & l_y \end{bmatrix}^T$

$$\Delta V(k) = \mathbf{e}^T(k) \mathbf{H}^T \mathbf{He}(k) - 2\mathbf{e}^T(k) \mathbf{H}^T \mathbf{L} + \mathbf{L}^T \mathbf{L} - \tilde{\mathbf{h}}^T(k) \tilde{\mathbf{h}}(k) \tag{56}$$

a sufficient condition for $\Delta V < 0$ is

$$||\tilde{\mathbf{h}}(k)||^2 \geq ||\mathbf{e}(k)||^2 ||\mathbf{H}||^2 + 2||\mathbf{e}(k)|| \, ||\mathbf{H}|| \, ||\mathbf{L}|| + ||\mathbf{L}||^2 \tag{57}$$

from (57) the following equation is obtained:

$$||\tilde{\mathbf{h}}(k)|| \geq (||\mathbf{e}(k)|| \, ||\mathbf{H}|| + ||\mathbf{L}||) \tag{58}$$

which sets the final size for the control error $\tilde{\mathbf{h}}(k)$. Both, (54) and (58) allows to state that the control error is bounded in terms of the *NN* approximation error.

7. Experimental results

The proposed controller was implemented on a Pioneer 2-DX mobile robot (Fig. 4), which admits linear and angular velocities as input reference signals. The controller was initialized with the dynamic parameters of the robot, which were obtained via identification, as proposed in Carelli and De La Cruz (2006).

In the experiment, the robot starts at $(r_x, r_y) = (0, 0)$ m, and a disturbance load of about 6 kg was placed on it. The robot must follow a circular reference trajectory:

$$\begin{cases} r_{xref} = 0.75 \sin(0.03\pi k T_0) \\ r_{yref} = 0.75 \cos(0.03\pi k T_0) \end{cases} \tag{59}$$

After 83 s, the neural compensation controller is switched-on and the error trajectory decreases alternating between 0.02 and 0.04 m. Figs. 5 and 6 depict the speeds of the robot and the control actions produced by the controller, and the references and output positions of the mobile robot, respectively.

Fig. 7 shows the trajectory followed by the mobile robot without neural compensation, and with adaptive neural compensation after

Fig. 4. The mobile robot Pioneer 2-DX.

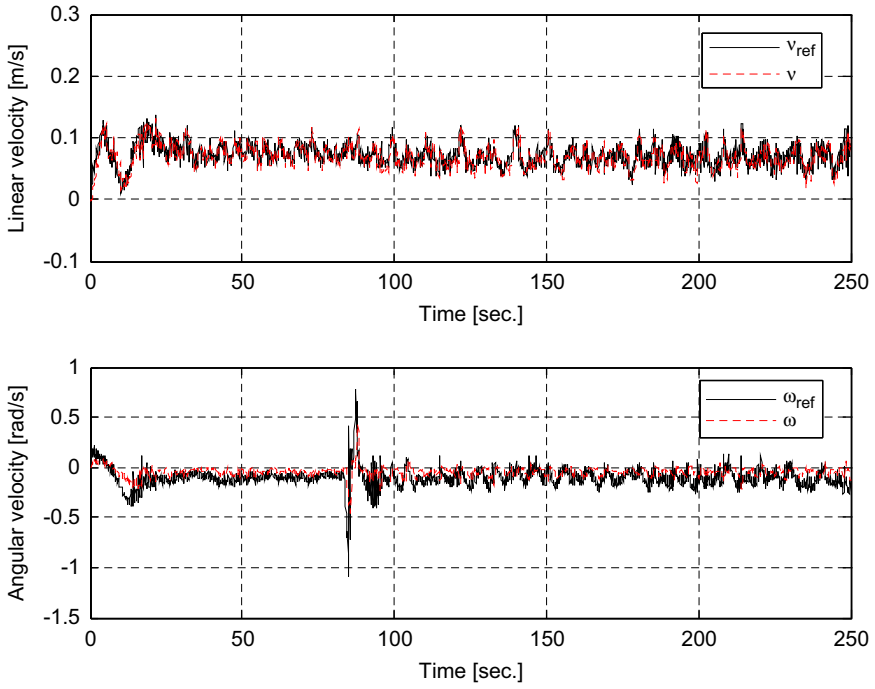

Fig. 5. Speeds of the robot and control actions by the RBF adaptive controller.

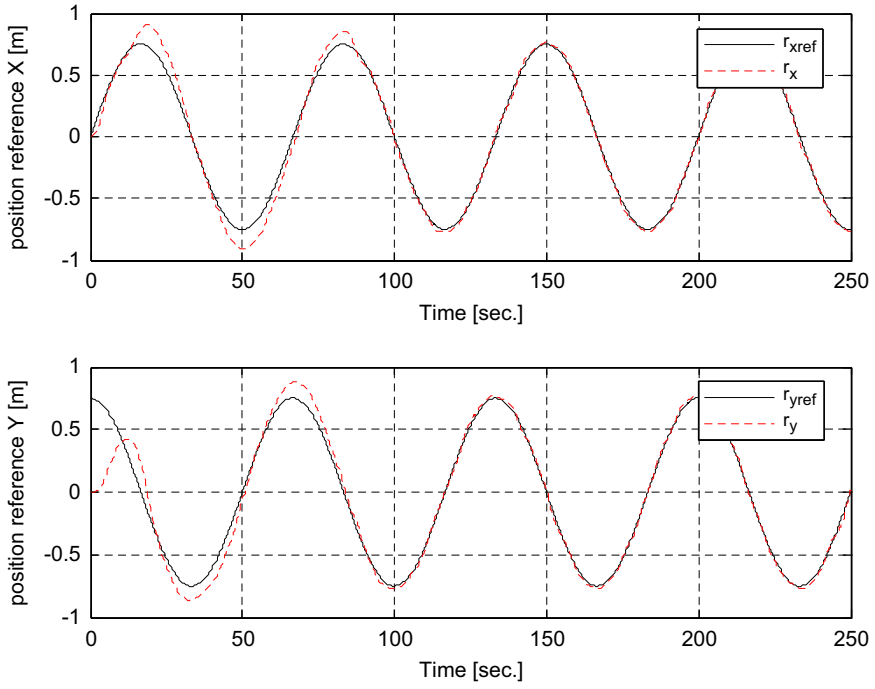

Fig. 6. References and output positions of the mobile robot.

the first 83 s of the run. Fig. 8 depicts the distance errors defined as the instantaneous distance between the reference and robot position. From the experimental results, and comparing the initial phase and the period during which the adaptive compensator is active, it remains clear the advantage of using the neural compensation. The controller is capable to compensate the dynamic uncertainties, which could be produced simply by changing the load or its position on the robot.

8. Conclusions

In this paper, a trajectory tracking controller for a unicycle-like mobile robot, including a neural adaptive compensator, is proposed. The controller is capable of generating smooth and continuous velocity commands to the robot. The tracking control errors are shown to be ultimately bounded, and the bounds are calculated as a function of the RBF-NN approximation errors. The

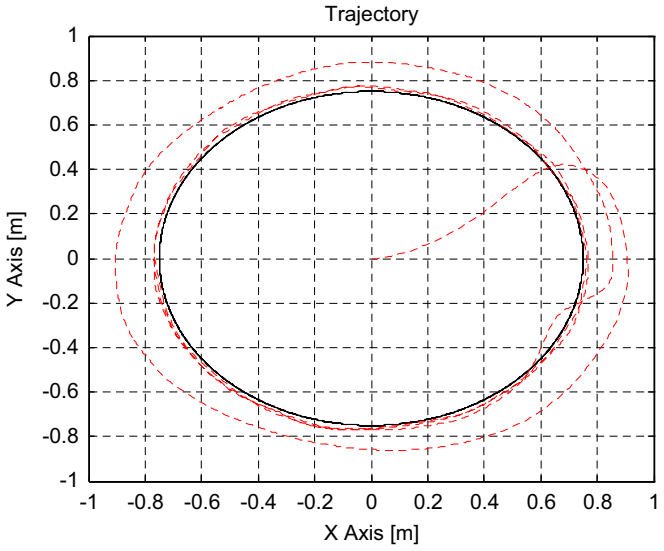

Fig. 7. Reference and actual trajectory of the mobile robot. The neural compensation is activated after 83 s as indicated in the figure.

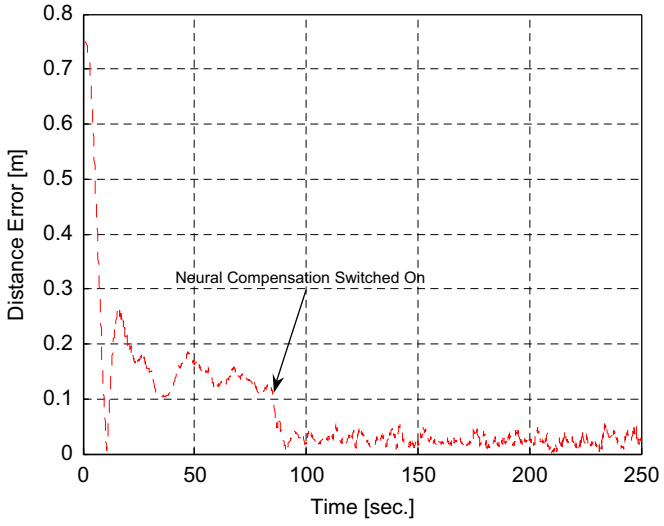

Fig. 8. Distance error without and with compensation controller.

RBF-NN controller compensates the difference between a known nominal dynamics and the actual dynamics of the robot. Therefore, the computational effort is significantly smaller than a NN learning the complete inverse model of the robot. Experimental results show the good performance of the proposed tracking controller and its capacity to adapt to the actual robot dynamics. The controller was designed in discrete time, which allows a direct implementation on a digital processing system.

References

Antonini, P., Ippoliti, G., & Longhi, S. (2006). Learning control of mobile robots using a multiprocessor system. *Control Engineering Practice*, *14*, 1279–1295.

Bahita, M., & Belarbi, K. (2006). Direct adaptive control of nonlinear systems using radial basis function network. *ACSE Journal*, *6*(1).

Bugeja, M.K., & Fabri, S.G. (2007). Dual adaptive control for trajectory tracking of mobile robots. In *Proceedings of the IEEE ICRA*, Roma, Italy, pp. 10–14.

Carelli, R., & De La Cruz, C. (2006). Dynamic modeling and centralized formation control of mobile robots. In *Proceedings of the 32nd annual conference of the IEEE Industrial Electronics Society IECON*, Paris.

Chen, C.-Y., Li, T.-H. S., & Yeh, Y.-C. (2009). EP-based kinematic control and adaptive fuzzy sliding-mode dynamic control for wheeled mobile robots. *Information Sciences*, *179*, 180–195.

Das, T., & Kar, I. N. (2006). Design and implementation of an adaptive fuzzy logic-based controller for wheeled mobile robots. *IEEE Transactions on Control Systems Technology*, *14*(3).

Dong, W., & Guo, Y. (2005). Dynamic tracking control of uncertain mobile robots. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems*, pp. 2774–2779.

Dong, W., & Huo, W. (1999). Tracking control of wheeled mobile robots with unknown dynamics. In *Proceedings of the IEEE international conference on robotics & automation*, Detroit, Michigan, pp. 2645–2650.

Fukao, T., Nakagawa, H., & Adachi, N. (2000). Adaptive tracking control of a mobile robot. *IEEE Transaction on Robotics and Automation*, *16*(5), 609–615.

Haykin, S. (2001th). *Neural networks: a comprehensive foundation* (2nd ed.). Prentice Hall, ISBN: 0132733501.

Hwang, C.K., Chen, B.S., & Chang, Y.T. (2004). Combination of kinematical and robust dynamical controllers for mobile robotics tracking control: (I) Optimal H_∞ control. In *Proceedings of IEEE international conference on control applications*, Taipei, Taiwan.

Inoue, R. S., Siqueira, A. A. G., & Terra, M. H. (2009). Experimental results on the nonlinear H-infinity control via quasi-LPV representation and game theory for wheeled mobile robots. *Robotica (Cambridge).*, *27*, 547–553.

Kim, M.S., Shin, J.H., & Lee, J.J. (2000). Design of a robust adaptive controller for a mobile robot. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems*, pp. 1816–1821.

Künhe, F., Gomes, J., & Fetter, W. (2005). Mobile robot trajectory tracking using model predictive control. In *Proceedings of the II IEEE Latin-American robotics symposium*, São Luis, Brazil.

Liu, S., Zhang, H., Yang, S.X., & Yu, J. (2004). Dynamic control of a mobile robot using an adaptive neurodynamics and sliding mode strategy. In *Proceedings of the fifth world congress on intelligent control and automation*, Hangzhou, China, pp. 5007–5011.

Martins, F. N., Celeste, W. C., Carelli, R., Sarcinelli Filho, M., & Bastos Filho, T. F. (2008). An adaptive dynamic controller for autonomous mobile robot trajectory tracking. *Control Engineering Practice*, *16*, 1354–1363.

Scaglia, G., Rosales, A., Quintero, L., Mut, V., & Agarwal, R. (2010). A linear-interpolation-based controller design for trajectory tracking of mobile robots. *Control Engineering Practice*, *18*(3), 318–329.

Wu, W., Chen, H., Wang, Y., & Woo, P. (1999). Adaptive exponential stabilization of mobile robots with uncertainties. In *Proceedings of the IEEE 38th conference on decision and control*, Phoenix, Arizona, USA, pp.3484–3489.

Yang, E., Gu, D., Mita, T., & Hu, H. (2004). *Nonlinear tracking control of a car-like mobile robot via dynamic feedback linearization, control.* UK: University of Bath.