Multilayer Perceptron Adaptive Dynamic Control of Mobile Robots: Experimental Validation

Marvin K. Bugeja †,* and Simon G. Fabri ‡

Department of Systems and Control Engineering
University of Malta, Msida, MSD 2080, Malta
† mkbuge@eng.um.edu.mt, ‡ sgfabr@eng.um.edu.mt

Abstract. This paper presents experimental results acquired from the implementation of an adaptive control scheme for nonholonomic mobile robots, which was recently proposed by the same authors and tested only by simulations. The control system comprises a trajectory tracking kinematic controller, which generates the reference wheel velocities, and a cascade dynamic controller, which estimates the robot's uncertain nonlinear dynamic functions in real-time via a multilayer perceptron neural network. In this manner precise velocity tracking is attained, even in the presence of unknown and/or time-varying dynamics. The experimental mobile robot, designed and built for the purpose of this research, is also presented in this paper.

Key words: Mobile robots, adaptive control, neural networks

1 Introduction

The vast majority of wheeled vehicles suffer from some kind of inherent mobility restriction. These restrictions manifest themselves as nonholonomic constraints in the kinematic model, and often arise due to the underactuated nature of the drive mechanism. Consequently, the linearized kinematic models of these vehicles lack controllability, full-state feedback linearization is out of reach, and smooth time-invariant feedback stabilization is unattainable [1]. These characteristics render the motion control of nonholonomic mobile robots not only practically relevant but also theoretically interesting and challenging.

A vast number of past contributions on the control of nonholonomic wheeled mobile robots (WMRs) [1, 2] completely ignore the robot dynamics and rely on the assumption that the control inputs, usually motor voltages, instantaneously establish the desired wheel velocities. As expected, controllers which explicitly account for the robot dynamics due to its mass, friction and inertia [3, 4] lead to better control performance. However, as argued in [3] *perfect* knowledge of the robot dynamics is unattainable in practice. Moreover, these parameters can even vary over time due to loading, wear, and ground conditions. In response, a

* This work was supported by the National Grant, RTDI-2004-026.

number of more advanced controllers have been proposed including: pre-trained neuro-controllers [5], parametric adaptive schemes [6], and robust sliding-mode methods [7]. A more powerful approach is that of online functional-adaptive control, where the uncertainty is not restricted to parametric terms, but covers completely the dynamic functions themselves [8, 9].

Of all the proposed adaptive controllers, only a few have ever been implemented on a physical robot, among which one finds [6, 10, 11]. In [10] D'Amico et. al. propose radial basis function (RBF) artificial neural networks (ANNs) for the adaptive control of WMRs. However, this work disregards the robot dynamics since ANNs are used solely to approximate the inverse *kinematic* model of the vehicle. On the other hand, Wang et. al. [6] and Dixon et. al. [11] do consider the robot dynamics in their adaptive control methods, but address only *parametric* uncertainty in the dynamic model. In contrast, in this paper we employ *functional-adaptive* neuro-control to handle better the uncertainty in the *dynamic functions* of the WMR, and not just its parameters.

The main contributions in this paper are the presentation and analysis of a set of experimental results which validate and compare the employed multilayer perceptron (MLP) adaptive control scheme for the first time, after it was originally proposed in our previous publication [9], and tested by simulations only. In addition this paper outlines the design and implementation of the mobile robot designed and built for the purpose of this research. The rest of the paper is organized as follows. Section 2 develops the discrete-time dynamic model of the WMR. This is then used in the neuro-adaptive dynamic controller revisited in Sect. 3. Section 4 outlines the experimental setup and the related design and implementation issues. Experimental results are presented and compared in Sect. 5, which is followed by a brief conclusion in Sect. 6.

2 Modelling

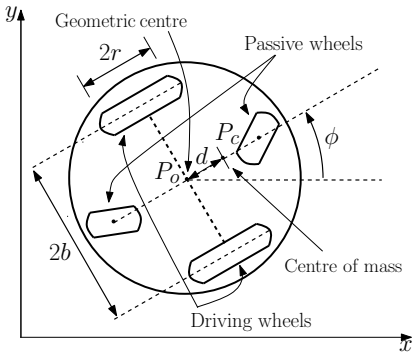
The differentially driven WMR considered in this paper is depicted in Fig. 1. The passive wheels are ignored and the driving wheels are assumed to roll without slipping. The robot state vector is given by $\boldsymbol{q} \triangleq [x \;\; y \;\; \phi \;\; \theta_r \;\; \theta_l]^T$, where (x, y) is the Cartesian coordinate of P_o, ϕ is the robot's orientation with reference to the xy frame, and θ_r, θ_l are the angular displacements of the right and left driving wheels respectively. The *pose* of the robot refers to $\boldsymbol{p} \triangleq [x \;\; y \;\; \phi]$.

2.1 Kinematics

The kinematic model of this WMR, detailed in [9], is given by:

$$\dot{\boldsymbol{q}} = \boldsymbol{S}(\boldsymbol{q})\boldsymbol{\nu} \;, \tag{1}$$

where the velocity vector $\boldsymbol{\nu} \triangleq [\nu_r \;\; \nu_l]^T \triangleq \left[\dot{\theta}_r \;\; \dot{\theta}_l\right]^T$, and $\boldsymbol{S} = \begin{bmatrix} \frac{r}{2}\cos\phi & \frac{r}{2}\cos\phi \\ \frac{r}{2}\sin\phi & \frac{r}{2}\sin\phi \\ \frac{r}{2b} & -\frac{r}{2b} \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$.



P_o	midpoint on the driving axle
P_c	centre of mass without wheels
d	distance from P_o to P_c
b	distance from each wheel to P_o
r	radius of each wheel
m_c	mass of the robot without wheels
m_w	mass of each wheel
I_c	angular mass of platform about P_c
I_w	angular mass of wheel about axle
I_m	angular mass of wheel about diameter

Fig. 1. Differentially driven wheeled mobile robot

2.2 Dynamics

The equations of motion of this WMR are given by:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{V}(\dot{\boldsymbol{q}}, \boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{F}(\dot{\boldsymbol{q}}) = \boldsymbol{E}(\boldsymbol{q})\boldsymbol{\tau} - \boldsymbol{A}^T(\boldsymbol{q})\boldsymbol{\lambda} \ , \tag{2}$$

where $\boldsymbol{M}(\boldsymbol{q})$ is the inertia matrix, $\boldsymbol{V}(\dot{\boldsymbol{q}}, \boldsymbol{q})$ is the centripetal and Coriolis matrix, $\boldsymbol{F}(\dot{\boldsymbol{q}})$ is a vector of frictional forces, $\boldsymbol{E}(\boldsymbol{q})$ is the input transformation matrix, $\boldsymbol{\tau}$ is the torque vector, and $\boldsymbol{A}^T(\boldsymbol{q})\boldsymbol{\lambda}$ is the vector of constraint forces.

Deriving the WMR dynamics, requires the elimination of the kinematic constraints $\boldsymbol{A}^T(\boldsymbol{q})\boldsymbol{\lambda}$ from (2). This is detailed in [3, 9], and yields

$$\bar{\boldsymbol{M}}\dot{\boldsymbol{\nu}} + \bar{\boldsymbol{V}}(\dot{\boldsymbol{q}})\boldsymbol{\nu} + \bar{\boldsymbol{F}}(\dot{\boldsymbol{q}}) = \boldsymbol{\tau} \ , \tag{3}$$

where:

$$\bar{\boldsymbol{M}} = \begin{bmatrix} \frac{r^2}{4b^2}(mb^2 + I) + I_w & \frac{r^2}{4b^2}(mb^2 - I) \\ \frac{r^2}{4b^2}(mb^2 - I) & \frac{r^2}{4b^2}(mb^2 + I) + I_w \end{bmatrix} , \quad \bar{\boldsymbol{V}}(\dot{\boldsymbol{q}}) = \begin{bmatrix} 0 & \frac{m_c r^2 d\dot{\phi}}{2b} \\ \frac{m_c r^2 d\dot{\phi}}{2b} & 0 \end{bmatrix} ,$$

$\bar{\boldsymbol{F}}(\dot{\boldsymbol{q}}) = \boldsymbol{S}^T(\boldsymbol{q})\boldsymbol{F}(\dot{\boldsymbol{q}})$, $I = (I_c + m_c d^2) + 2(I_m + m_w b^2)$, and $m = m_c + 2m_w$.To account for the fact that the controller is finally implemented on a digital computer, the continuous-time dynamics (3) are discretized through a first order forward Euler approximation with a sampling interval of T seconds, resulting in

$$\boldsymbol{\nu}_k - \boldsymbol{\nu}_{k-1} = \boldsymbol{f}_{k-1} + \boldsymbol{G}_{k-1}\boldsymbol{\tau}_{k-1} \ , \tag{4}$$

where the subscript integer k denotes that the corresponding variable is evaluated at time kT seconds, and vector $\boldsymbol{f}_{k-1}$ and matrix $\boldsymbol{G}_{k-1}$, which together encapsulate the WMR dynamics, are given by

$$\boldsymbol{f}_{k-1} = -T\bar{\boldsymbol{M}}_{k-1}^{-1}\left(\bar{\boldsymbol{V}}_{k-1}\boldsymbol{\nu}_{k-1} + \bar{\boldsymbol{F}}_{k-1}\right), \quad \boldsymbol{G}_{k-1} = T\bar{\boldsymbol{M}}_{k-1}^{-1} \ . \tag{5}$$

The control input $\boldsymbol{\tau}$ is assumed to remain constant over a sampling interval of T seconds, which is chosen low enough for the Euler approximation error to be negligible.

3 Control Scheme

The complete neuro-adaptive control system detailed in [9], and briefly revisited in this section is depicted in Fig. 2. Some variables in this figure are defined later in the article. At this point one should particularly note the modular architecture which enables the kinematic and dynamic control modules to be treated separately [3]. The kinematic controller computes the desired wheel velocities in order to minimize the robot tracking or stabilization error, according to the task at hand[1]. The cascaded dynamic controller, which in the case of this paper is neuro-adaptive, ensures that the robot truly tracks these velocities, by determining the torques required at the wheels.

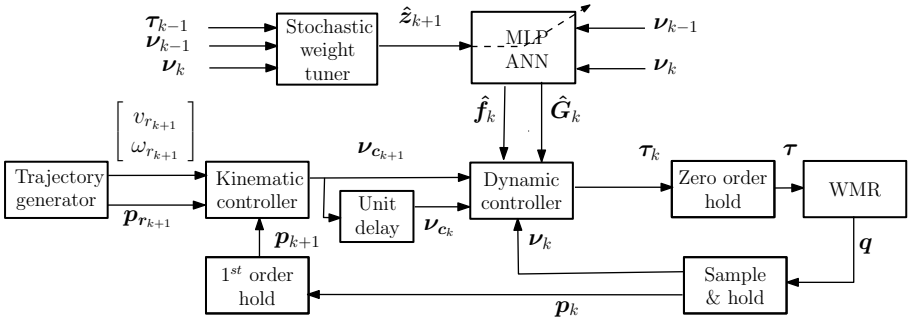


Fig. 2. MLP adaptive dynamic control scheme

3.1 Kinematic Control

To address the trajectory tracking problem we employ a discrete-time version of the trajectory tracking controller originally proposed in [2] and given by:

$$\boldsymbol{\nu_c}_k = \begin{bmatrix} \frac{1}{r} & \frac{b}{r} \\ \frac{1}{r} & -\frac{b}{r} \end{bmatrix} \begin{bmatrix} v_{rk}\cos e_{3k} + k_1 e_{1k} \\ \omega_{rk} + k_2 v_{rk} e_{2k} + k_3 v_{rk} \sin e_{3k} \end{bmatrix} , \tag{6}$$

where $\boldsymbol{\nu_c}_k$ is the wheel velocity command vector computed by the kinematic controller, k_1, k_2, and k_3 are *positive* design parameters, v_{rk} and ω_{rk} are the translational and angular *reference* velocities respectively, and e_{1k}, e_{2k}, e_{3k} make up the tracking error vector defined as

$$\boldsymbol{e}_k \triangleq \begin{bmatrix} e_{1k} \\ e_{2k} \\ e_{3k} \end{bmatrix} \triangleq \begin{bmatrix} \cos\phi_k & \sin\phi_k & 0 \\ -\sin\phi_k & \cos\phi_k & 0 \\ 0 & 0 & 1 \end{bmatrix} (\boldsymbol{p_r}_k - \boldsymbol{p}_k) , \tag{7}$$

where $\boldsymbol{p_r}_k \triangleq [x_{rk} \;\; y_{rk} \;\; \phi_{rk}]^T$ denotes the reference pose vector.

[1] In this paper only the trajectory tracking problem is considerd.

3.2 Dynamic Functional-Adaptive Control

Robot Dynamics ANN Estimator A sigmoidal MLP ANN is employed to approximate in real-time, the vector of nonlinear functions $\boldsymbol{f}_{k-1}$, the estimate of which is denoted by $\hat{\boldsymbol{f}}_{k-1}$ and given by

$$\hat{\boldsymbol{f}}_{k-1} = \begin{bmatrix} \boldsymbol{\phi}^T(\boldsymbol{x}_{k-1}, \hat{\boldsymbol{a}}_k)\hat{\boldsymbol{w}}_{\mathbf{1}k} \\ \boldsymbol{\phi}^T(\boldsymbol{x}_{k-1}, \hat{\boldsymbol{a}}_k)\hat{\boldsymbol{w}}_{\mathbf{2}k} \end{bmatrix}, \tag{8}$$

in the light of the following definitions and assumptions:[2]

1. $\boldsymbol{x}_{k-1}$ is the ANN input vector, and is set to $[\boldsymbol{\nu}_{k-1} \;\; 1]$.
2. $\boldsymbol{\phi}$ is the sigmoidal activation vector, whose ith element is given by $\phi_i = 1/\left(1+\exp(-\boldsymbol{s}_{\boldsymbol{i}}^T\boldsymbol{x})\right)$, where $\boldsymbol{s}_{\boldsymbol{i}}$ is the parameter vector of the ith neuron that is estimated in real-time.
3. The sigmoidal parameter vectors are grouped in $\hat{\boldsymbol{a}}_k \triangleq \left[\hat{\boldsymbol{s}}_{\mathbf{1}_k}^T \;\; \cdots \;\; \hat{\boldsymbol{s}}_{\boldsymbol{L}_k}^T\right]^T$.
4. L denotes the number of neurons in the network.
5. $\hat{\boldsymbol{w}}_{\boldsymbol{j}k}$ represents the synaptic weight vector of the connection between the neurons and the jth output element of the network, estimated in real-time.

Since $\boldsymbol{G}_{k-1}$ is a symmetric state-independent matrix, its estimate does not require an ANN, but is simply denoted by the parameters

$$\hat{\boldsymbol{G}}_{k-1} = \begin{bmatrix} \hat{g}_{1k-1} & \hat{g}_{2k-1} \\ \hat{g}_{2k-1} & \hat{g}_{1k-1} \end{bmatrix}, \tag{9}$$

where $\hat{g}_{1k-1}$ and $\hat{g}_{2k-1}$ are the unknown elements in $\hat{\boldsymbol{G}}_{k-1}$.

All the unknown parameters requiring estimation are grouped in a single vector $\hat{\boldsymbol{z}}_k \triangleq \begin{bmatrix} \hat{\boldsymbol{w}}_{\mathbf{1}k}^T & \hat{\boldsymbol{w}}_{\mathbf{2}k}^T & \hat{\boldsymbol{a}}_k^T & \hat{g}_{1k-1} & \hat{g}_{2k-1} \end{bmatrix}^T$. Estimates $\hat{\boldsymbol{f}}_{k-1}$ and $\hat{\boldsymbol{G}}_{k-1}$ are employed in rewriting the WMR dynamic model (4) in the following state-space form:

$$\begin{aligned} \boldsymbol{z}_{k+1}^* &= \boldsymbol{z}_k^* \\ \boldsymbol{y}_k &= \boldsymbol{h}\left(\boldsymbol{x}_{k-1}, \boldsymbol{\tau}_{k-1}, \boldsymbol{z}_k^*\right) + \boldsymbol{\epsilon}_k \;, \end{aligned} \tag{10}$$

where the vector field $\boldsymbol{h} = \hat{\boldsymbol{f}}_{k-1}(\boldsymbol{x}_{k-1}, \boldsymbol{w}_{\mathbf{1}k}^*, \boldsymbol{w}_{\mathbf{2}k}^*, \boldsymbol{a}_k^*) + \hat{\boldsymbol{G}}_{k-1}(g_{1k-1}^*, g_{2k-1}^*)\boldsymbol{\tau}_{k-1}$, the measured output is denoted by $\boldsymbol{y}_k = \boldsymbol{\nu}_k - \boldsymbol{\nu}_{k-1}$, and $\boldsymbol{\epsilon}_k$ is an independent zero-mean white Gaussian process with covariance matrix $\boldsymbol{R}_{\boldsymbol{\epsilon}}$, which accounts for measurement uncertainty. Since $\boldsymbol{h}$ is nonlinear in terms of the unknown parameter vector $\boldsymbol{z}_k^*$, which is treated as a random variable, the extended Kalman filter (EKF) is used for its real-time stochastic estimation:

$$\begin{aligned} \hat{\boldsymbol{z}}_{k+1} &= \hat{\boldsymbol{z}}_k + \boldsymbol{K}_k\boldsymbol{i}_k \\ \boldsymbol{P}_{k+1} &= \boldsymbol{P}_k - \boldsymbol{K}_k\boldsymbol{\nabla}_{\boldsymbol{h}k}\boldsymbol{P}_k \;, \end{aligned} \tag{11}$$

where $\boldsymbol{\nabla}_{\boldsymbol{h}k}$ denotes the Jacobian matrix of $\boldsymbol{h}$ with respect to $\boldsymbol{z}_k^*$ evaluated at $\hat{\boldsymbol{z}}_k$, $\boldsymbol{P}_k$ is estimate's covariance matrix, $\boldsymbol{K}_k = \boldsymbol{P}_k\boldsymbol{\nabla}_{\boldsymbol{h}k}^T\left(\boldsymbol{\nabla}_{\boldsymbol{h}k}\boldsymbol{P}_k\boldsymbol{\nabla}_{\boldsymbol{h}k}^T + \boldsymbol{R}_{\boldsymbol{\epsilon}}\right)^{-1}$ is the

[2] The $\hat{}$ and * notations denote *estimates* and *optimal* parameters respectively.

Kalman gain matrix, and $\boldsymbol{i}_k = \boldsymbol{y}_k - \boldsymbol{h}\left(\boldsymbol{x}_{k-1}, \boldsymbol{\tau}_{k-1}, \hat{\boldsymbol{z}}_k\right)$ is the innovations vector. Using (8), (9), and the definition of $\boldsymbol{h}$ it can be shown that:

$$\boldsymbol{\nabla}_{\boldsymbol{h}k} = \begin{bmatrix} \boldsymbol{\phi}_{k-1}^T & \mathbf{0}^T & \cdots \hat{w_{1,i}}(\phi_i - {\phi_i}^2)\boldsymbol{x}^T \cdots & \tau_{rk-1}\ \tau_{lk-1} \\ \mathbf{0}^T & \boldsymbol{\phi}_{k-1}^T & \cdots \hat{w_{2,i}}(\phi_i - {\phi_i}^2)\boldsymbol{x}^T \cdots & \tau_{lk-1}\ \tau_{rk-1} \end{bmatrix},$$

where: $i = 1, \ \ldots \ , L$ and $\hat{w_{j,i}}$ denotes the ith element of the synaptic weight vector $\hat{\boldsymbol{w}}_{\boldsymbol{j}k}$ which connects the neurons to the jth network output; Notationwise $\boldsymbol{\phi}_{k-1}$ implies that the activation function is evaluated for $\boldsymbol{x}_{k-1}$ and $\hat{\boldsymbol{a}}_k$; $\mathbf{0}$ denotes a zero vector having the same length as $\boldsymbol{\phi}_{k-1}$; ϕ_i and $\boldsymbol{x}$ correspond to time instant $(k-1)$; $\tau_{r_{k-1}}$, $\tau_{l_{k-1}}$ are the first and second elements of the torque vector $\boldsymbol{\tau}_{k-1}$ respectively.

Control Law At each control iteration $\hat{\boldsymbol{f}}_k$ and $\hat{\boldsymbol{G}}_k$, generated by the MLP stochastic function estimator as outlined in the previous paragraphs in this section, are used in the following control law to ensure that the robot velocity vector $\boldsymbol{\nu}_k$, tracks the velocity command vector issued by the kinematic controller $\boldsymbol{\nu}_{\boldsymbol{c}k}$:

$$\boldsymbol{\tau}_k = \hat{\boldsymbol{G}}_k^{-1}\left(\boldsymbol{\nu}_{\boldsymbol{c}k+1} - \boldsymbol{\nu}_k - \hat{\boldsymbol{f}}_k + k_d\left(\boldsymbol{\nu}_{\boldsymbol{c}k} - \boldsymbol{\nu}_k\right)\right), \tag{12}$$

where the design parameter $-1 < k_d < 1$. If we neglect the negligibly small inherent ANN approximation error (justified in the light of the *Universal Approximation Theorem* of neural networks) and the measurement noise $\boldsymbol{\epsilon}_k$, this control law (12) yields the following closed-loop dynamics: $\boldsymbol{\nu}_{k+1} = \boldsymbol{\nu}_{\boldsymbol{c}k+1} + k_d\left(\boldsymbol{\nu}_{\boldsymbol{c}k} - \boldsymbol{\nu}_k\right)$, which clearly indicate that $|\boldsymbol{\nu}_{\boldsymbol{c}_k} - \boldsymbol{\nu}_k| \to 0$ as $k \to \infty$.

4 Experimental Setup

The neuro-adaptive control scheme revisited in Sect. 3, was implemented successfully for the first time on a physical WMR pictured in Fig. 3 named NeuroBot, which was recently designed and built by the authors to serve as a testbed in the development and validation of robot neuro-control algorithms.

NeuroBot is a differentially driven WMR. Each of the two 125mm diameter solid-rubber drive wheels, is independently driven by a 70W, 24V permanent magnet dc motor from *maxon motor*, which is equipped with a 113:1 reduction gearbox, and a 500 pulses per revolution incremental encoder. Each of the two motors is driven via the LMD18200 H-Bridge IC from *National Semiconductor*, which is controlled by a 20kHz pulse width modulation (PWM) reference signal. The instantaneous current in each motor is measured using the HX 03-P/SP2 Hall effect current transducer from *LEM*, and filtered by a 4th-order continuous-time Bessel lowpass filter tuned for a corner frequency of 2kHz, and implemented using the MAX275 continuous-time filter IC from *Maxim Dallas Semiconductors.* NeuroBot is powered by four 12V, 9Ah sealed lead acid batteries (RM 12-9 HR) from *REMCO*.

The algorithms controlling NeuroBot are all implemented on a *MicroAutoBox* system from *dSPACE*. The *MicroAutoBox* is a compact stand-alone prototyping

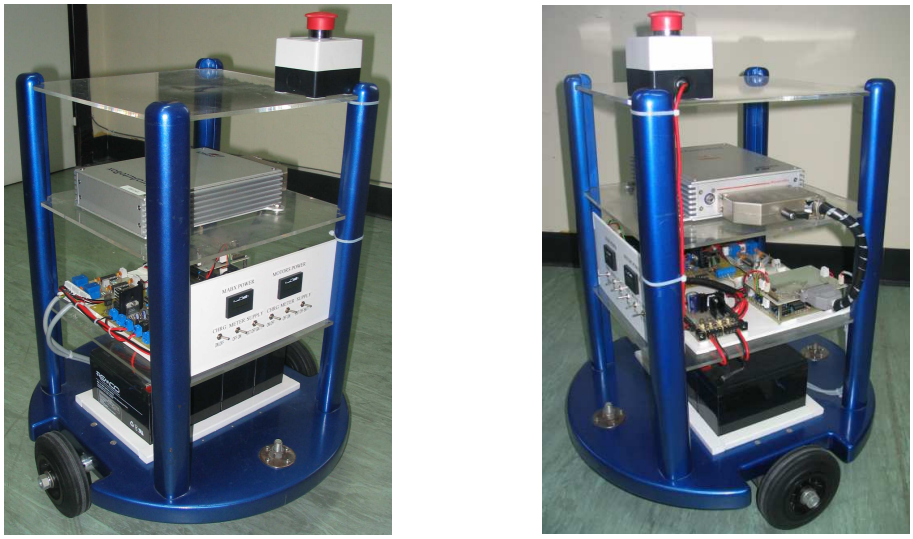


Fig. 3. NeuroBot: the experimental WMR

unit designed specifically for the rapid-prototyping of computationally demanding real-time control systems, typically requiring a number of analog/digital input and output channels to interface with ease with both sensors and actuators.

A digital pole-placement torque controller with integral action was designed and implemented in software to account for the motor dynamics. This inner torque control loop uses the current measurement as feedback and issues motor voltage commands to the motors. This ascertains that the actual torques at the wheels, which are proportional to currents in the motors, track precisely those issued by the outer loop control law (12). Naturally, this cascade approach imposes that the inner loop operates at a rate which is much faster than that of the outer loop. The sampling rates for the inner and outer loops where chosen to be 10kHz and 200Hz respectively.

A desktop computer is used to implement the control algorithms in *Simulink*® using the system blocks provided by the *dSpace Real-Time Interface*. *Real-Time Workshop*® is then used to automatically generate the required code which is then downloaded to the *MicroAutoBox* (which features also non-volatile memory) via the *dSpace Link Board* installed in the computer. From this point onwards the *MicroAutoBox* can be disconnected from the computer and the control code runs entirely on the *MicroAutoBox* which employs a multitasking approach to service each of the the two control loops in real-time. Vital information about the real-time execution of each task running on the *MicroAutoBox*, such as sampling times, priorities and execution times, can also be monitored via *ControlDesk*, developed for this purpose by *dSPACE*.

5 Experimental Results

NeuroBot was used to test and validate the MLP adaptive control scheme, reviewed in Sect. 3, for the first time on a real mobile robot. Moreover, the proposed adaptive controller was compared with its nonadaptive counterpart implemented via (12) but with $\hat{\boldsymbol{f}}_{k-1}$ and $\hat{\boldsymbol{G}}_{k-1}$ replaced by $\boldsymbol{f}_{k-1}$ and $\boldsymbol{G}_{k-1}$.

This nonadaptive controller requires that the WMR parameters are known. For this reason NeuroBot was weighed and measured accordingly and the resulting parameter values were used to tune the nonadaptive controller. These parameters are: $b = 0.2295$m, $r = 0.0625$m, $d = 0$m, $m_c = 22$kg, $m_w = 1$kg, $I_c = 0.6320\text{kgm}^2$, $I_w = 0.002\text{kgm}^2$, and $I_m = 0.0029\text{kgm}^2$. Moreover, viscous friction was included by setting $\boldsymbol{F}(\dot{\boldsymbol{q}}) = \boldsymbol{F_c}\dot{\boldsymbol{q}}$, where $\boldsymbol{F_c}$ is a diagonal matrix of coefficients, with nominal diagonal values set to $[0.001, 0.001, 0.001, 0.18, 0.18]$. In contrast, the neuro-adaptive controller presented in this paper does not require any preliminary knowledge about the robot dynamics. Consequently the initial network parameter vector $\hat{\boldsymbol{z}}_0$ was generated randomly. The MLP ANN contained 10 neurons (L=10).

A number of experimental results acquired from a typical experiment on NeuroBot as detailed above are presented in Fig. 4. Plot (a) depicts NeuroBot tracking a demanding reference trajectory (characterized by high linear and angular accelerations) for non-zero initial tracking error, when it is being controlled by the neuro-adaptive MLP scheme. It is clear that NeuroBot swiftly adapts to its own dynamics and simultaneously converges to the the reference trajectory. The WMR keeps tracking the trajectory with high precision for the rest of the experiment. This plot on its own validates experimentally the neuro-adaptive scheme presented in this paper for ultra-precise trajectory tracking. Plots (b) and (c) show the tracking errors $x_r - x$, $y_r - y$ and $\phi_r - \phi$ against time, corresponding to the same trajectory shown in Plot (a). From these plots it is also clear that the trajectory tracking errors are all reduced to zero in a few seconds and maintained there with unquestionable performance. Plot (d) shows the actual and reference angular wheel velocities ν_r and ν_l along the trajectory. The actual velocities are practically superimposed on the corresponding references. This implies that the adaptive dynamic controller achieves the wheel velocities requested by the kinematic controller with great precision, which ultimately leads to the good trajectory tracking performance depicted in the previous plots. Plots (e) and (f) compare the adaptive controller with its tuned and untuned nonadaptive counterparts, by depicting the position error norm $\sqrt{((x_r - x)^2 + (y_r - y)^2)}$ and the orientation error respectively for the three controllers following the same trajectory.

The tuned nonadaptive controller refers to the nonadaptive controller tuned with the true robot parameters. The untuned nonadaptive controller refers to the same controller but with $m_c = 10$kg, i.e. this controller believes that the robot weighs half its real weight. This scenario was included to examine the effects of uncertain and/or time-varying robot dynamics on the performance of nonadaptive dynamic controllers. From the two plots it is clear that the performance of the nonadaptive controller deteriorates from the tuned to the untuned case. This

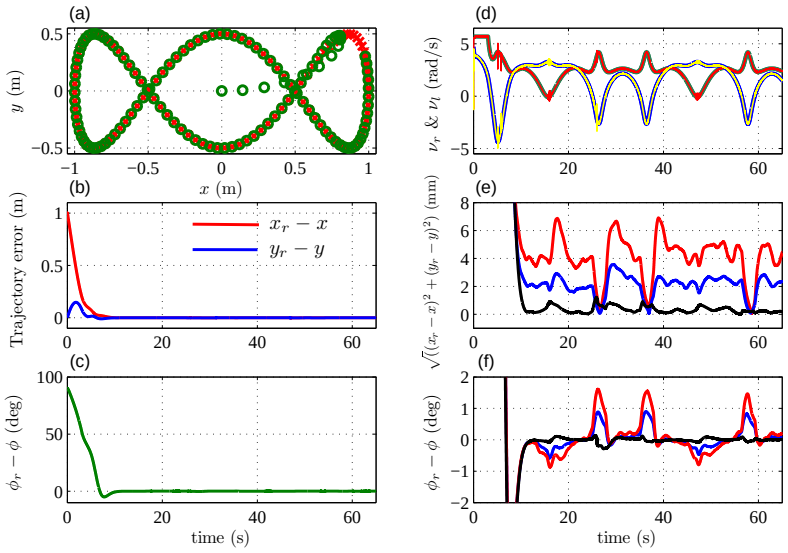


Fig. 4. (a): reference (×) & actual trajectories (◯); (b): position errors; (c): orientation error; (d): angular velocities - right wheel (red), left wheel (yellow) & their references (green & blue respectively); (e): adaptive (black), tuned nonadaptive (blue), untuned nonadaptive (red); (f): as for (e) but orientation errors.

indicates the incapability of nonadaptive controllers in handling misinformation about the robot dynamics. More impressive is the fact that the adaptive controller outperforms even the tuned nonadaptive controller. We attribute this to the fact that perfect modelling is practically impossible, and since the adaptive controller uses no predefined dynamic model of the WMR, since it learns it in real-time, it does not suffer from the consequences of unmodelled dynamics and inexact model parameters.

The inner torque control loop, operating at a sampling rate of 10kHz, takes approximately 14μs to execute. The outer speed loop, with a sampling rate of 200Hz, executes in 700μs and 350μs for the adaptive and nonadaptive cases respectively. This implies that the adaptive controller requires double the execution time of its nonadaptive counterpart. This was expected since the adaptive algorithm is more computationally demanding because it includes estimation as opposd to a nonadaptive controller. Nonetheless, the execution time of the neuro-adaptive controller is still as little as 14% of the total sampling period, implying that commercially available hardware is well endowed to handle the increased level of computational load brought about by real-time neuro-adaptive control.

6 Conclusion

The contribution of this paper comprises the experimental validation of a MLP adaptive dynamic control scheme, originally proposed in [9] for the trajectory tracking of mobile robots. In addition the recently designed robotic testbed NeuroBot is introduced, and the associated implementation issues are briefly discussed. The experimental results presented in this paper not only validate the employed neuro-adaptive control scheme in practice, but also demonstrate the great improvements in performance over non-adaptive schemes in the face of uncertain and/or time-varying robot dynamics. In addition, NeuroBot proved to be a very good research testbed and will continue to be used for the validation and development of innovative controllers for mobile robots.

References

1. Canudas de Wit, C., Khennoul, H., Samson, C., Sordalen, O.J.: Nonlinear control design for mobile robots. In Zheng, Y.F., ed.: Recent Trends in Mobile Robots. Robotics and Automated Systems. World Scientific (1993) 121–156
2. Kanayama, Y., Kimura, Y., Miyazaki, F., Noguchi, T.: A stable tracking control method for an autonomous mobile robot. In: Proc. IEEE International Conference of Robotics and Automation, Cincinnati, OH (May 1990) 384–389
3. Fierro, R., Lewis, F.L.: Control of a nonholonomic mobile robot: Backstepping kinematics into dynamics. In: Proc. IEEE 34th Conference on Decision and Control (CDC'95), New Orleans, LA (December 1995) 3805–3810
4. Corradini, M.L., Orlando, G.: Robust tracking control of mobile robots in the presence of uncertainties in the dynamic model. Journal of Robotic Systems **18**(6) (2001) 317–323
5. Oubbati, M., Schanz, M., Levi, P.: Kinematic and dynamic adaptive control of a nonholonomic mobile robot using RNN. In: Proc. IEEE Symposium on Computational Intelligence in Robotics and Automation (CIRA'05), Helsinki, Finland (June 2005)
6. Wang, T.Y., Tsai, C.C.: Adaptive trajectory tracking control of a wheeled mobile robot via lyapunov techniques. In: Proc. 30th Annual Conference of the IEEE Industrial Electronics Society, Busan, Korea (November 2004) 389–394
7. Corradini, M.L., Ippoliti, G., S.Longhi, Michelini, S.: Neural networks inverse model approach for the tracking problem of mobile robots. In: Proc. The 9th International Workshop on Robotics (RAAD'00), Maribor, Slovenia (June 2000)
8. Fierro, R., Lewis, F.L.: Control of a nonholonomic mobile robot using neural networks. IEEE Trans. Neural Netw. **9**(4) (July 1998) 589–600
9. Bugeja, M.K., Fabri, S.G.: Multilayer perceptron adaptive dynamic control for trajectory tracking of mobile robots. In: Proc. 32nd Annual Conf. of the IEEE Ind. Electronics Society (IECON'06), Paris, France (November 2006) 3798–3803
10. D'Amico, A., Ippoliti, G., Longhi, S.: A radial basis function networks apporach for the tracking problem of mobile robots. In: Proc. IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Como, Italy (2001) 498–503
11. Dixon, W.E., Dawson, D.M., Zergeroglu, E., Behal, A.: Adaptive tracking control of a wheeled mobile robot via an uncalibrated camera system. IEEE Trans. Syst., Man, Cybern. B **31**(3) (2001) 341–352