Automatica 44 (2008) 2716–2723

Contents lists available at ScienceDirect

Automatica

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

Adaptive critic motion control design of autonomous wheeled mobile robot by dual heuristic programming☆

Wei-Song Lin*, Ping-Chieh Yang

Department of Electrical Engineering, National Taiwan University, Taiwan

ARTICLE INFO

Article history:
Received 21 January 2007
Received in revised form
2 September 2007
Accepted 19 March 2008
Available online 10 October 2008

Keywords:
Adaptive critic
Approximate dynamic programming
Mobile robot
Neural networks

ABSTRACT

Autonomous wheeled mobile robot (WMR) needs implementing velocity and path tracking control subject to complex dynamical constraints. Conventionally, this control design is obtained by analysis and synthesis or by domain expert to build control rules. This paper presents an adaptive critic motion control design, which enables WMR to autonomously generate the control ability by learning through trials. The design consists of an adaptive critic velocity control loop and a self-learning posture control loop. The neural networks in the velocity neuro-controller (VNC) are corrected with the dual heuristic programming (DHP) adaptive critic method. Designer simply expresses the control objective by specifying the primary utility function then VNC will attempt to fulfill it through incremental optimization. The posture neuro-controller (PNC) learns by approximating the specialized inverse velocity model of WMR so as to map planned positions to suitable velocity commands. Supervised drive supplies variant velocity commands for PNC and VNC to set up their neural weights. During autonomous drive, while PNC halts learning VNC keeps on correcting its neural weights to optimize the control performance. The proposed design is evaluated on an experimental WMR. The results show that the DHP adaptive critic design is a useful base of autonomous control.

1. Introduction

Autonomous wheeled mobile robots (WMR) rely on using sensors to percept their surroundings and use a motion controller to drive automatically (Chen & Redmill, 2004; Maurette, 2003; Meyrowitz, Blidberg, & Michelson, 1996). In the motion control, WMR should be capable of performing trajectory tracking, path following and stabilization. However, WMR is a nonholonomic dynamic system with intrinsic nonlinearity, and commonly with unmodeled disturbance and unstructured, unmodeled dynamics (Greenwood, 1988). Unless its mass is negligible (Lee, Leung, & Tam, 1999), the motion control should deal with the complex dynamics (Bloch, Reyhanoglu, & McClamroch, 1992; Kanayama, Kimura, Miyazaki, & Noguchi, 1990). Conventionally, this control design relies on engineers to analyze the WMR system so as to synthesize the appropriate controller (Colbaugh, Barany, & Glass, 1998; Park, Cho, & Lee, 2001; Tsai, Wu, Chang, & Wang, 2002). But usually difficulties arise from absence of accurate WMR model. Fuzzy control design may skip building the model but needs domain expert to construct the fuzzy rules (Lee, Adams, & Ryoo, 1997; Pawlowski, Kozlowski, & Wroblewski, 2001). Controllers based on neural networks or neuro-fuzzy networks may construct the control function by learning training samples (Fierro & Lewis, 1998; Jang & Sun, 1995; Narendra & Parthasathy, 1990). But preparing appropriate training samples usually needs an existing controller (Gu & Hu, 2002). Alternatively, the adaptive critic motion control design presented in this paper enables WMR to develop the control ability autonomously. Neither domain expert to build control rules nor existing controller to generate training samples is required.

In our laboratory, an experimental WMR has been developed and its mathematical model has been formulated and identified (Lin, Huang, Chuang, & Liu, 2004). A hierarchical fuzzy control system has been implemented and shown able to conduct the motion of WMR (Lin, Huang, & Chuang, 2005). Furthermore, the experimental WMR has been equipped with a stereovision module to enable autonomous path finding and collision avoidance (Lin, Chuang, & Tien, 2005). This paper assumes the stereovision module foresees nearest path and the WMR system must generate the motion control function entirely through learning by trials. Essentially, this extends the definition of autonomous robot to autonomous development of the control ability. The idea is to obtain the control ability by learning through trials to fulfill the control objective. Neural networks are chosen as the basic learning model. Trials, actually supervised trials for the sake of safety,

☆ This paper was not presented at any IFAC meeting. This paper was recommended for publication in revised form by Associate Editor Jae-Bok Song under the direction of Editor Mituhiko Araki.

* Corresponding author. Tel.: +886 2 33663638; fax: +886 2 23638247.
E-mail addresses: weisong@cc.ee.ntu.edu.tw (W.-S. Lin), r93921078@ntu.edu.tw (P.-C. Yang).

0005-1098/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.automatica.2008.03.029

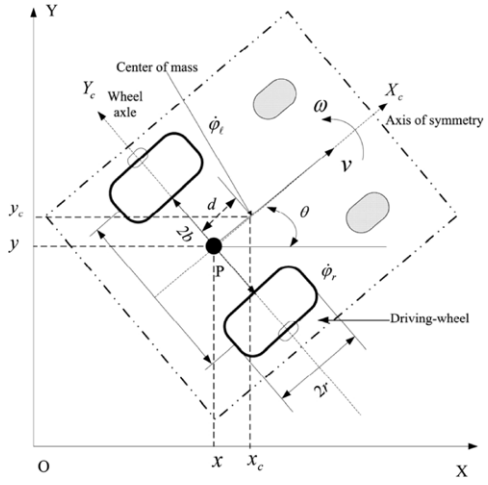

Fig. 1. A schematic top view of the mobile platform.

supply training inputs to set up the neural weights. Eventually, the motion control function is built without reference to any existing controller.

The dual heuristic programming (DHP) adaptive critic technique (Prokhorov & Wunsch, 1997; Werbos, 1992), which approximates dynamic programming, is invoked to develop the learning mechanism. Multilayered perceptrons (MLP) are used to construct the posture neuro-controller (PNC) and the velocity neuro-controller (VNC). PNC learns to map planned positions to suitable velocity commands. VNC learns to conduct the WMR motion so as to track the velocity commands. Supervised drive of WMR in variant velocities supplies training inputs for PNC and VNC to set up their neural weights. During autonomous drive, while PNC halts learning, VNC is corrected to optimize the control performance. The proposed design is successfully evaluated on the experimental WMR. The principal contribution is to develop an autonomous control design scheme for mobile robots based on the DHP adaptive critic method.

This paper is organized as follows: Section 2 illustrates the architecture of the adaptive critic motion control system of WMR. Section 3 presents the design of the DHP adaptive critic motion controller. Section 4 validates the proposed design on the experimental WMR. Section 5 is the conclusion.

2. Architecture of adaptive critic motion control system of WMR

The interested autonomous WMR has a four-wheeled mechanism as shown in Fig. 1 and Table 1. While the front wheels are passive, the rear wheels are motorized independently to give the differential rotation configuration. Such a WMR with stereovision module has been assembled in our laboratory for studying navigation and motion control (Lin et al., 2004; Lin, Huang et al., 2005; Lin, Chuang et al., 2005). The experimental WMR is completely autonomous because data are elaborated without any external aid, and its sensors are the encoders attached to the motorized wheels and the stereovision module to find the path.

Using Lagrange formalism, the dynamical model of WMR is obtained as (Lin et al., 2004; Yun & Yamamoto, 1993)

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{R}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{R} + \mathbf{F}(\dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) + \boldsymbol{\tau}_d = \mathbf{B}(\mathbf{q})\mathbf{u} \tag{1}$$

where $\mathbf{q} = [x, y, \theta, \phi_l, \phi_r]^{\mathrm{T}}$ is the generalized coordinate vector to characterize WMR, $\mathbf{R} = [v, \omega]^{\mathrm{T}}$ in which v is the linear velocity and ω is the angular velocity, $\mathbf{u} = [\tau_l, \tau_r]^{\mathrm{T}}$ are the input torques

Table 1
Notations in Fig. 1 and values of the experimental WMR

$d = 0.1$ m	Displacement from point P along X_c axis to the center of mass
$r = 0.125$ m	Radius of motorized wheels
$b = 0.265$ m	Half length of rear axle
$l = 0.8$ m	Length of WMR
$m_c = 110$ kg	Mass of the body (excluding motorized wheels and the associated rotors)
$m_w = 5$ kg	Mass of single motorized wheel and rotor set
$I_c = 1.057$ kg m^2	Inertia of the body without motorized wheels and rotor sets
$I_w = 0.004$ kg m^2	Inertia of single motorized wheel and rotor set
$I_m = 0.002$ kg m^2	Inertia of single motorized wheel and rotor set about a diameter
v	Linear velocity of WMR
ω	Angular velocity of WMR
θ	Orientation of WMR
$\dot{\varphi}_l$	Angular velocity of left motorized wheel
$\dot{\varphi}_r$	Angular velocity of right motorized wheel

generated by the left and right motors. The parameter matrices in (1) are

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} m + \dfrac{2I_w}{r^2} & 0 \\ 0 & I + \dfrac{2b^2 I_w}{r^2} \end{bmatrix},$$

$$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} 0 & -\dot{\theta} m_c d \\ \dot{\theta} m_c d & 0 \end{bmatrix},$$

$$\mathbf{B}(\mathbf{q}) = \begin{bmatrix} \dfrac{2}{r} & \dfrac{2}{r} \\ \dfrac{-2b}{r} & \dfrac{2b}{r} \end{bmatrix}$$

where $m = m_c + 2m_w$, and $\mathbf{F}(\dot{\mathbf{q}})$, $\mathbf{G}(\mathbf{q})$ and $\boldsymbol{\tau}_d$ are unknown terms corresponding to frictional, gravitational and disturbed forces, respectively. To conduct the WMR motion needs implementing velocity and trajectory tracking control. Hierarchical fuzzy control was shown a feasible approach (Lin, Chuang et al., 2005), but needs domain experts to construct the fuzzy rules. Alternatively, this paper seeks to build the motion control function entirely through learning by trials. The innovative design is called the adaptive critic motion control system, which consists of mainly a self-learning posture control loop and an adaptive critic velocity control loop. Fig. 2 illustrates the design concept. The stereovision module finds a forward path. According to the forward path, feedback positions and physical limitations of WMR, the path planner calculates the planed positions. PNC which approximates the specialized inverse velocity model (Narendra & Parthasathy, 1990) of WMR maps planned positions to suitable velocity commands. Actually, VNC is a DHP adaptive critic design which invokes incremental optimization to generate the ability of velocity control through learning. Learning begins with supervised drive to set up the neural weights in VNC and PNC. Hence, the supervised drive should excite the WMR dynamics sufficiently in the interested working domain so that the learning would be complete. During autonomous drive, while PNC halts learning VNC is corrected successively to optimize the control performance.

3. Design of DHP adaptive critic motion controller

3.1. Adaptive critic velocity neuro-controller

Adaptive critic methods are usually practiced with model-based learning structures such as neural or neuro-fuzzy networks. They have common roots as generalizations of dynamic programming for neural reinforcement learning approaches and have a capability of optimization over time under conditions of noise,

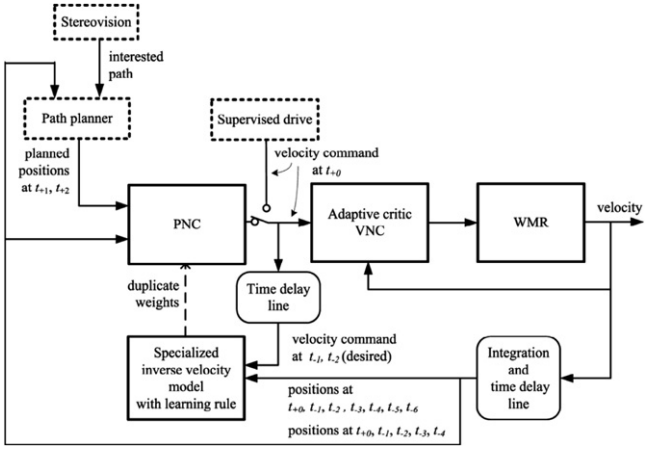

Fig. 2. Architecture of the adaptive critic motion control system.

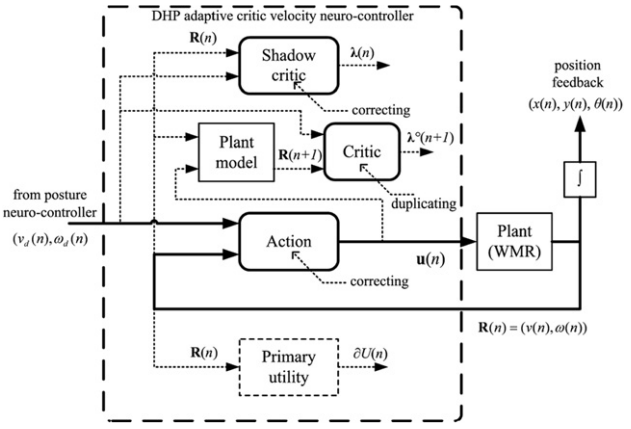

Fig. 3. Architecture of the DHP adaptive critic velocity neuro-controller. The solid lines indicate signal paths, the dashed lines indicate data paths, and the round rectangular blocks represent neural networks.

uncertainty, and nonlinearity (Werbos, 1992, 2004). Heuristic dynamic programming (HDP), dual heuristic programming (DHP), and globalized dual heuristic programming (GDHP), and their action dependent companions are the main categories of adaptive critic designs (Prokhorov & Wunsch, 1997). They can be differentiated by the critic output. HDP uses the critic to estimate the value function in the Bellman equation of dynamic programming. In DHP, the critic approximates the derivative value function to facilitate the computation in the gradient correcting rule. The critic in GDHP estimates both the value function and its derivatives. DHP was shown to have a superior performance to HDP and no observable improved performance by GDHP (Lendaris & Shannon, 1998; Prokhorov, Santiago, & Wunsch, 1995). In addition, incremental optimization based on dynamic programming is rigorous in theory. Stability of a trained DHP adaptive critic control system is governed by the optimal control theory in the sense of dynamic programming (Bertsekas, 2005).

As illustrated in Fig. 3, VNC contains blocks called the action network, critic network, shadow critic network, plant model and primary utility. The action network is responsible for producing suitable control signals while the critic and shadow critic networks form the adaptive critic to critique the action performance. The plant model can be either mathematical formulations or neural approximation of the WMR dynamics.

3.1.1. Neural computing of VNC

The action, critic and shadow critic networks are each implemented with three-layer perceptrons (Haykin, 1999). These

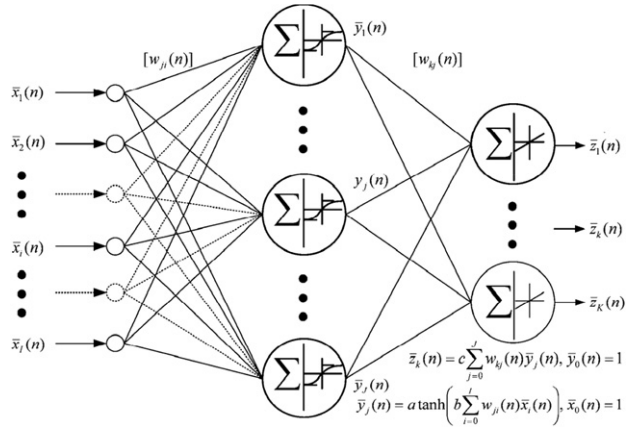

Fig. 4. Architecture of the three-layer perceptrons.

neural networks have the common architecture as shown in Fig. 4. In the neural architecture, each hidden neuron has a hyperbolic tangent activation function to obtain output as

$$\bar{y}_j(n) = a\tanh\left(b\sum_{i=0}^{I} w_{ji}(n)\bar{x}_i(n)\right), \qquad \bar{x}_0(n) = 1,$$
$$(a, b) > 0 \tag{2}$$

where n denotes time sequence. Each output neuron has a linear activation function to obtain output as

$$\bar{z}_k(n) = c\sum_{j=0}^{J} w_{kj}(n)\bar{y}_j(n), \qquad \bar{y}_0(n) = 1, \quad c > 0. \tag{3}$$

The partial derivatives pertaining to the neural architecture are derived as follows:

$$\frac{\partial \bar{z}_k(n)}{\partial w_{kj}(n)} = c\bar{y}_j(n) \tag{4}$$

$$\frac{\partial \bar{z}_k(n)}{\partial w_{ji}(n)} = \left(\frac{bc}{a}\right) w_{kj}(n)[a - \bar{y}_j(n)][a + \bar{y}_j(n)]\bar{x}_i(n) \tag{5}$$

$$\frac{\partial \bar{z}_k(n)}{\partial \bar{x}_i(n)} = \sum_{j=1}^{J}\left\{\left(\frac{bc}{a}\right) w_{kj}(n)[a - \bar{y}_j(n)][a + \bar{y}_j(n)]w_{ji}(n)\right\}. \tag{6}$$

Usually, (4) and (5) are called the sensitivity functions and (6) is called the Jacobian function. DHP adaptive critic design needs these quantities to evaluate the correcting rules.

3.1.2. Plant model and Jacobian quantities

In Fig. 3, the plant model is used to predict the immediate future states and calculate certain partial derivatives pertaining to the plant. It can be either the mathematical model or neural approximation of the plant dynamics. Since DHP adaptive critic design allows using partial or qualitative plant model (Shannon, 1999) and the WMR model is known (Lin et al., 2004). The plant model in VNC is implemented with (1) but neglecting the unknown terms corresponding to the frictional, gravitational and disturbed forces. From (1), the simplified model equations are derived as below.

$$\dot{\mathbf{R}} = -\mathbf{M}^{-1}(\mathbf{q})\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{R}} + \mathbf{M}^{-1}(\mathbf{q})\mathbf{B}(\mathbf{q})\mathbf{u}. \tag{7}$$

Rewrite (7) as the following nonlinear mappings:

$$\dot{R}_i = f_i(\mathbf{R}, \mathbf{u}), \quad i = 1, 2, \ldots, S. \tag{8}$$

Then for the operating point $(\mathbf{R}_n, \mathbf{u}_n)$ at sampling time t_n, the first-order approximation of (7) is obtained as

$$\dot{\mathbf{R}}(t) = \mathbf{A}(n)\mathbf{R}(t) + \mathbf{B}(n)\mathbf{u}(t) + \mathbf{D}(n) \tag{9}$$

where t denotes real time,

$$A_{ij}(n) = \left.\frac{\partial f_i(\mathbf{R}, \mathbf{u})}{\partial R_j}\right|_{\mathbf{R}_n, \mathbf{u}_n}, \qquad B_{ik}(n) = \left.\frac{\partial f_i(\mathbf{R}, \mathbf{u})}{\partial u_k}\right|_{\mathbf{R}_n, \mathbf{u}_n} \quad \text{and}$$

$$D_i(n) = f_i(\mathbf{R}_n, \mathbf{u}_n) - \sum_{j=1}^{S} \left.\frac{\partial f_i(\mathbf{R}, \mathbf{u})}{\partial R_j}\right|_{\mathbf{R}_n, \mathbf{u}_n} R_{nj} - \sum_{k=1}^{K} \left.\frac{\partial f_i(\mathbf{R}, \mathbf{u})}{\partial u_k}\right|_{\mathbf{R}_n, \mathbf{u}_n} u_{nk}.$$

The discrete form of the simplified WMR model is obtained as

$$\mathbf{R}(n+1) = \bar{\mathbf{A}}(n)\mathbf{R}(n) + \bar{\mathbf{B}}(n)\mathbf{u}(n) + \bar{\mathbf{D}}(n) \tag{10}$$

where $\bar{\mathbf{A}}(n) = e^{\mathbf{A}(n)\Delta}$, $\bar{\mathbf{B}}(n) = \int_0^{\Delta} e^{\mathbf{A}(n)t}dt\,\mathbf{B}(n)$, $\bar{\mathbf{D}}(n) = \int_0^{\Delta} e^{\mathbf{A}(n)t}dt\,\mathbf{D}(n)$, and where Δ represents sampling period. In VNC, the plant model uses (10) to predict the states and calculates the following Jacobian quantities

$$\frac{\partial R_i(n+1)}{\partial R_j(n)} = \bar{A}_{ij}(n), \qquad \frac{\partial R_i(n+1)}{\partial u_k(n)} = \bar{B}_{ik}(n). \tag{11}$$

3.1.3. Correcting the action network

In Fig. 3, $U(n)$ is the primary utility function defined by according to the specific application context. Since the objective of VNC is to control WMR to track the velocity command as closely as possible, the primary utility function is defined as

$$U(n) = 0.25(v(n) - v_d(n))^2 + 0.25(\omega(n) - \omega_d(n))^2 \tag{12}$$

where $(v_d(n), \omega_d(n))$ is the velocity command. To achieve the control objective, the neural weights in the action network must be corrected to minimize not only the present value but also the sum of all future values of $U(n)$. According to dynamic programming (Bellman, 1957), this goal can be achieved by minimizing the secondary utility function, i.e. value function, expressed as

$$J(n) = \sum_{k=0}^{\infty} \eta^k U(n+k) = U(n) + \eta J(n+1) \tag{13}$$

where η, $0 < \eta \leq 1$ is a discount factor. Thus, using the gradient descent method, a suitable correcting rule of the action network is

$$\begin{aligned}\Delta w_{km}(n) &= \alpha \frac{\partial J(n)}{\partial w_{km}(n)} = \alpha \frac{\partial J(n)}{\partial u_k(n)} \frac{\partial u_k(n)}{\partial w_{km}(n)} \\ &= \alpha \left(\frac{\partial U(n)}{\partial u_k(n)} + \eta \frac{\partial J(n+1)}{\partial u_k(n)} \right) \frac{\partial u_k(n)}{\partial w_{km}(n)} \\ &= \alpha \left(\underbrace{\frac{\partial U(n)}{\partial u_k(n)}}_{\text{Utility}} + \eta \sum_s \underbrace{\lambda_s^{\circ}(n+1)}_{\text{Critic}} \underbrace{\frac{\partial R_s(n+1)}{\partial u_k(n)}}_{\text{Model}} \right) \underbrace{\frac{\partial u_k(n)}{\partial w_{km}(n)}}_{\text{Action}}\end{aligned} \tag{14}$$

where α is the learning rate and $w_{km}(n)$ is the mth neural weight associated with the kth output of the action.

3.1.4. Correcting the shadow critic network and the critic network

In (14), $\lambda_s^{\circ}(n+1) = \partial J(n+1)/\partial R_s(n+1)$ is unknown. DHP design embodies in estimating this quantity by the adaptive critic which is composed of the shadow critic and critic networks. They estimate the partial derivatives of the secondary utility function at present and immediate future sampling times as

$$\lambda_s(n) = \frac{\partial J(n)}{\partial R_s(n)}, \quad s = 1, 2, \ldots, S \text{ (shadow critic)} \tag{15}$$

$$\lambda_s^{\circ}(n+1) = \frac{\partial J(n+1)}{\partial R_s(n+1)}, \quad s = 1, 2, \ldots, S \text{ (critic)} \tag{16}$$

where S denotes the dimension of $\mathbf{R}(n)$. Take partial derivative on (13) and substitute (16) into its right hand side to obtain

$$\begin{aligned}\lambda_s^{\circ}(n) &= \frac{\partial}{\partial R_s(n)} (U(n) + \eta J(n+1)) \\ &= \frac{\partial U(n)}{\partial R_s(n)} + \sum_{k=1}^{K} \left\{ \frac{\partial U(n)}{\partial u_k(n)} \frac{\partial u_k(n)}{\partial R_s(n)} \right\} + \eta \sum_{s'=1}^{S} \left\{ \frac{\partial J(n+1)}{\partial R_{s'}(n+1)} \right. \\ &\quad \left. \times \left(\frac{\partial R_{s'}(n+1)}{\partial R_s(n)} + \sum_{k=1}^{K} \left(\frac{\partial R_{s'}(n+1)}{\partial u_k(n)} \frac{\partial u_k(n)}{\partial R_s(n)} \right) \right) \right\}\end{aligned} \tag{17}$$

where K denotes the dimension of the control vector $\mathbf{u}(n)$. Since (12) shows $U(n)$ is independent of $\mathbf{u}(n)$, (17) can be rewritten as

$$\begin{aligned}\lambda_s^{\circ}(n) = \underbrace{\frac{\partial U(n)}{\partial R_s(n)}}_{\text{utility}} + \eta \sum_{s'=1}^{S} \left\{ \underbrace{\lambda_{s'}{}^{\circ}(n+1)}_{\text{Critic}} \left(\underbrace{\frac{\partial R_{s'}(n+1)}{\partial R_s(n)}}_{\text{Model}} \right.\right. \\ \left.\left. + \sum_{k=1}^{K} \left(\underbrace{\frac{\partial R_{s'}(n+1)}{\partial u_k(n)}}_{\text{Model}} \underbrace{\frac{\partial u_k(n)}{\partial R_s(n)}}_{\text{Action}} \right) \right) \right\}.\end{aligned} \tag{18}$$

In (18), $\lambda_{s'}{}^{\circ}(n+1)$ is the output of the critic network, $\partial R_{s'}(n+1)/\partial R_s(n)$ and $\partial R_{s'}(n+1)/\partial u_k(n)$ are the Jacobian functions of the plant model, $\partial u_k(n)/\partial R_s(n)$ is the Jacobian function of the action network, and $U(n)$ is a known function, therefore, $\lambda_s^{\circ}(n)$ can be calculated. The adaptive critic in DHP learns by updating the shadow critic network so that $\lambda(n)$ tracks $\lambda^{\circ}(n)$. Hence, an error measure for correcting the shadow critic network can be formulated as

$$E(n) = 0.5 \sum_s \left(\lambda_s(n) - \lambda_s^{\circ}(n)\right)^2. \tag{19}$$

Then the gradient correcting rule is

$$\Delta w_{sm}(n) = \beta \frac{\partial E(n)}{\partial w_{sm}(n)} = \beta \left(\lambda_s(n) - \lambda_s^{\circ}(n)\right) \frac{\partial \lambda_s(n)}{\partial w_{sm}(n)} \tag{20}$$

where β is the learning rate and w_{sm} is the mth neural weight associated with the sth output of the shadow critic network. The critic network duplicates the corresponding neural weights of the shadow critic network therefore no correcting rule is needed. But for learning convergence, duplication is made only for every several (typically five) sampling times.

3.2. Self-learning posture neuro-controller

PNC consists of two neural networks mentioned as the linear and angular PNC to map planned positions to linear and angular velocity commands. The self-learning mechanism is constructed by identifying the specialized inverse velocity model of WMR as shown in Fig. 5. For learning convergence, the specialized inverse velocity model and PNC are organized as standalone neural networks. The neural architecture is shown in Fig. 4. The linear PNC has twelve inputs organized from two planned positions and five feedback positions as follows:

$$\begin{aligned}&[x(n+2) - x(n+1), y(n+2) - y(n+1), x(n+1) - x(n), \\ &y(n+1) - y(n), x(n-1) - x(n), y(n-1) - y(n), \\ &x(n-2) - x(n-1), y(n-2) - y(n-1), x(n-3) \\ &- x(n-2), y(n-3) - y(n-2), x(n-4) - x(n-3), \\ &y(n-4) - y(n-3)]^{\mathrm{T}}.\end{aligned} \tag{21}$$

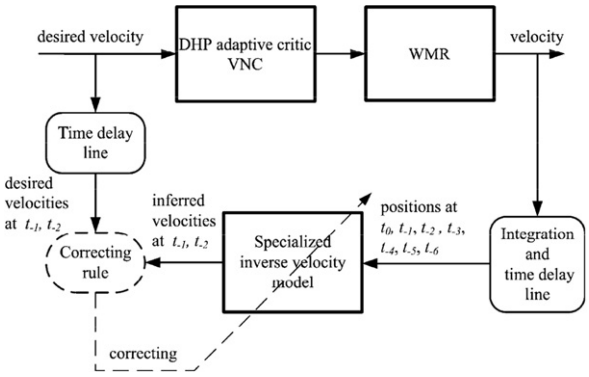

Fig. 5. Scheme of learning the specialized inverse velocity model.

Actually, (21) contains multi-step displacements to imply the velocity, acceleration and jerk for PNC to determine the outputs. The output of the linear PNC is the linear velocity command $v_d(n)$ and $v_d(n+1)$, where $v_d(n)$ is active and $v_d(n+1)$ is dummy. Similarly, the angular PNC has eighteen inputs organized as below

$$\begin{aligned}&[x(n+2)-x(n+1), y(n+2)-y(n+1),\\ &\theta(n+2)-\theta(n+1),\\ &x(n+1)-x(n), y(n+1)-y(n), \theta(n+1)-\theta(n),\\ &x(n-1)-x(n), y(n-1)-y(n), \theta(n-1)-\theta(n)\\ &x(n-2)-x(n-1), y(n-2)-y(n-1), \theta(n-2)\\ &-\theta(n-1), x(n-3)-x(n-2), y(n-3)-y(n-2),\\ &\theta(n-3)-\theta(n-2),\\ &x(n-4)-x(n-3), y(n-4)-y(n-3),\\ &\theta(n-4)-\theta(n-3)]^{\mathrm{T}}. \end{aligned} \tag{22}$$

The output of the angular PNC is the angular velocity command $\omega_d(n)$ and $\omega_d(n+1)$, where $\omega_d(n)$ is active and $\omega_d(n+1)$ is dummy.

Fig. 5 shows the scheme of learning the specialized inverse velocity model. The specialized inverse, which is not necessarily the complete inverse, covers simply the working domain excited by supervised drive. Therefore, no singularity of WMR would be encountered. Certainly, supervised drive must supply rich enough, safe velocity commands to encompass the working domain requested in autonomous drive. At the end of supervised drive, PNC duplicates the neural weights in the specialized inverse velocity model. During autonomous drive, the neural weights in PNC are kept constant so that VNC can incrementally optimize the velocity control.

Backpropagation with Levenberg Marquardt algorithm (LM) (Wilamowski, 2003) is used to correct the specialized inverse velocity model. Define the error of the velocity inferred by the specialized inverse velocity model as

$$\begin{aligned}&\mathbf{e}(n)\\ &= [(v_d(n-1)-v^{\mathrm{infer}}(n-1)), (v_d(n-2)-v^{\mathrm{infer}}(n-2)),\\ &(\omega_d(n-1)-\omega^{\mathrm{infer}}(n-1)), (\omega_d(n-2)-\omega^{\mathrm{infer}}(n-2))]^{\mathrm{T}}. \end{aligned} \tag{23}$$

For the usage of the LM algorithm, $\mathbf{e}(n)$ are collected for m sampling times. Then the error measure is constructed as

$$\varepsilon(\mathbf{W}) = 0.5\mathbf{E}^{\mathrm{T}}\mathbf{E} \tag{24}$$

where $\mathbf{E} = [\mathbf{e}^{\mathrm{T}}(n), \mathbf{e}^{\mathrm{T}}(n-1), \ldots, \mathbf{e}^{\mathrm{T}}(n-m+1)]^{\mathrm{T}}$. Then the neural weights in the specialized inverse velocity model are corrected as

$$\mathbf{W}_{k+1} = \mathbf{W}_k - [\mathbf{G}^{\mathrm{T}}\mathbf{G} + \xi\,\mathbf{I}]^{-1}\mathbf{G}^{\mathrm{T}}\mathbf{E} \tag{25}$$

where $\mathbf{G} = \nabla(\varepsilon(\mathbf{W}))$ and $\mathbf{I}$ is the identity matrix. When using (25), the scalar ξ is decreased after each successful step, i.e. reduction in the error measure, and increased only when a tentative step would increase the error measure. This provides a switching capability between the Gauss–Newton algorithm and the steepest descent method.

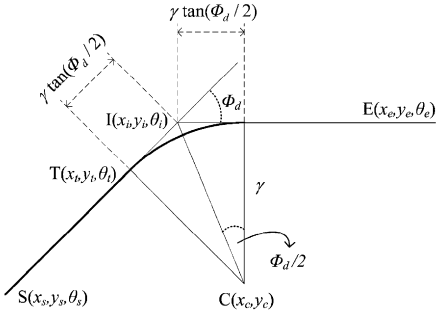

Fig. 6. Planning a smooth path with the arc-line algorithm.

3.3. Path planner

The stereovision module is responsible to locate the target and find a collision-free path. According to the viewed path and considering the physical limitations of WMR, the path planner plans a feasible, smooth path and calculates planned positions for next two steps. The arc-line algorithm (Nelson, 1989) is used to smooth the path. As illustrated in Fig. 6, this algorithm replaces the line segments around the intersection of two straight lines with a smooth curve. First, the start point $S(x_s, y_s, \theta_s)$ on the first line, the end point $E(x_e, y_e, \theta_e)$ on the second line, the intersection point $I(x_i, y_i, \theta_i)$ of these two lines, and the angle $(\phi_d = \theta_i - \theta_e)$ between these two lines are found. Then a value of curvature (γ) is assigned to find the transition point $T(x_t, y_t, \theta_t)$ on the first line, the distance $\gamma\tan(\phi_d/2)$ to the intersection point, and the center point $C(x_c, y_c)$. Finally, the original straight line segments are replaced by the arc starts at point T.

As shown in Fig. 7, denote the physical limitations of WMR on maximum displacement and steering-angle as $d_{\max}$ and $\phi_{\max}$. By constructing a displacement vector from present position (x_p, y_p, θ_p) to a target position (x_b, y_b) selected on the planned path, the desired displacement d_p and steering angle ϕ_p can be determined. Then the planed linear and angular positions are calculated as

$$\begin{aligned}x_p(n+1) &= x_p(n) + d_p\cos(\phi_p+\theta_p)\\ y_p(n+1) &= y_p(n) + d_p\sin(\phi_p+\theta_p)\\ \theta_p(n+1) &= \theta_p(n) + \phi_p. \end{aligned} \tag{26}$$

When they violate the physical limitations, the maximum allowable values are used.

4. Validation of DHP adaptive critic motion control design

In the following validation, VNC of the experimental WMR is implemented as below. The action network has four inputs $[v(n), \omega(n), v_d(n), \omega_d(n)]^{\mathrm{T}}$ and two outputs corresponding to wheel's driving torques $[\tau_l(n), \tau_r(n)]^{\mathrm{T}}$. The shadow critic network has four inputs and two outputs denoted by $[v(n), \omega(n), v_d(n), \omega_d(n)]^{\mathrm{T}}$ and $[\lambda_1(n), \lambda_2(n)]^{\mathrm{T}}$, respectively. The critic network is a duplicate of the shadow critic network except the inputs and outputs are $[v(n+1), \omega(n+1), v_d(n), \omega_d(n)]^{\mathrm{T}}$ and $[\lambda_1^{\circ}(n+1), \lambda_2^{\circ}(n+1)]^{\mathrm{T}}$, respectively. The number of hidden neurons in each neural network is chosen by experience. The parameter values in the activation functions of (2) and (3) are $a, b, c = 1$. All neural weights in the action, critic and shadow critic networks are initialized with values chosen randomly in the range $[-0.1, 0.1]$.

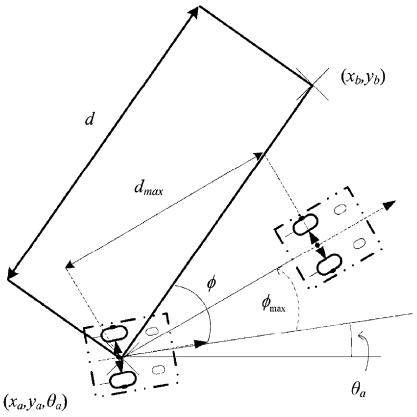

Fig. 7. Planning a feasible position.

The validation begins with supervised drive and followed by autonomous drive. Supervised drive supplies 1000 sets of velocity commands calculated from the following equations:

$$\begin{aligned} v_d(i) &= 0.5\left[\cos\left(\frac{6\pi(i-1)}{1000}+\pi\right)+1\right]\mathrm{e}^{-0.001i}, \\ &\quad i=1,2,\ldots,1000 \\ w_d(i) &= \left[\cos\left(\frac{6\pi(i-1)}{1000}+\pi\right)+1\right]\sin\left(\frac{i-1}{40}\right), \\ &\quad i=1,2,\ldots,1000. \end{aligned} \tag{27}$$

These velocity commands are fed sequentially into VNC to train the neural networks for 500 cycles. It should be noticed that supervised drive is responsible to supply rich enough, safe velocity commands without the corresponding control torques. Here rich enough velocity commands mean all possibilities covering the working domain requested in autonomous drive. Therefore, each training cycle is actually a trial to generate appropriate control torques by optimizing the secondary utility function. In the mean time of each trial, the neural weights in VNC and PNC are corrected. Hence, PNC is simply equivalent to the specialized inverse velocity model excited by supervised drive. After finishing 500 training cycles, the performance of VNC and PNC is examined. Finally, the trained WMR system is turned into autonomous drive and tested by tracking a right-turn path and a decaying sinusoidal path.

4.1. Performance of VNC

Figs. 8a and 8b compare the actual linear velocity with desired value in the 50th and 500th training cycles. The velocity error in the 500th cycle is significantly smaller than that of in the 50th cycle. The result in the angular velocity is similar but not presented. After finishing the 500 training cycles, the WMR system is commanded to track the following velocity pattern

$$v_{t2}(i) = \begin{cases} 0.002i & 1 \le i \le 300 \\ 0.6 & 300 < i \le 600 \\ 1 & 600 < i \le 700 \\ 3.1 - 0.003i & 700 < i \le 900 \\ 0.4 & 900 < i \le 1000 \end{cases} \tag{28}$$

$$w_{t2}(i) = \begin{cases} -0.002i & 1 \le i \le 300 \\ -0.6 & 300 < i \le 600 \\ -1 & 600 < i \le 700 \\ 0.003i - 3.1 & 700 < i \le 900 \\ -0.4 & 900 < i \le 1000. \end{cases}$$

Figs. 9a and 9b shows both linear and angular velocity tracking are accurate. Apparently, the DHP adaptive critic learning algorithm converges and appropriate VNC is obtained.

Fig. 8a. Result of linear velocity tracking in the 50th cycle.

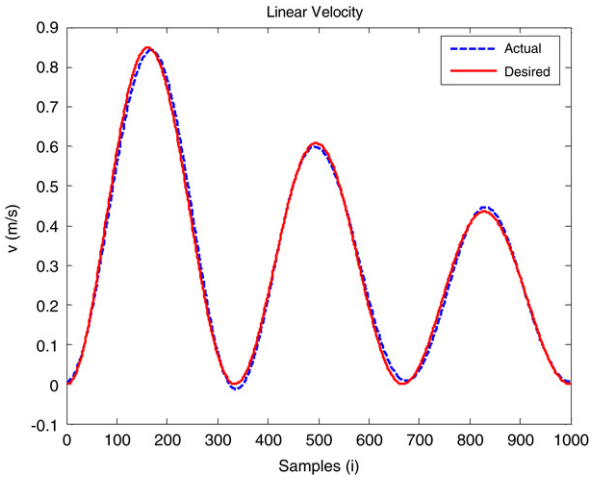

Fig. 8b. Result of linear velocity tracking in the 500th cycle.

4.2. Performance of PNC

The trained WMR system is turned into autonomous drive and commanded to track a sequence of positions calculated with

$$y(i) = \cos\left(\frac{33\pi}{1000}x(i)\right), \quad i = 1,2,\ldots,1000. \tag{29}$$

During autonomous drive, the outputs of PNC are recorded. The recorded values corresponding to the linear and angular velocities are presented as the dashed curve (actual) in Figs. 10a and 10b. The solid curve (desired) is obtained by using the fuzzy posture controller designed by Lin, Huang et al. (2005). Both curves are close to each other. Obviously, PNC performs as well as the fuzzy posture controller. The difference is that while the fuzzy posture controller was built by domain expert, PNC is obtained entirely by machine learning.

4.3. Performance of the trained WMR system

The trained WMR system is turned into autonomous drive and commanded to track a right-turn path and then a decaying sinusoidal path. The limitations on the posture control are $d_{\max} = 0.035$ and $\phi_{\max} = 0.03$.