Adaptive Critic Designs-Based Autonomous Unmanned Vehicles Navigation: Application to Robotic Farm Vehicles

Daniel Patiño[1] and Santiago Tosetti
Instituto de Automática, Advanced Intelligent Systems Laboratory, Universidad Nacional de San Juan
Av. Lib. San Martín 1109 (O), 5400 San Juan, Argentina.
Argentina

1. Introduction

Unmanned vehicles like Unmanned Aerial Vehicles (UAV) and Unmanned Ground Vehicles (UGV) are mechanical devices capable of moving in some environment with a certain degree of autonomy. These vehicles use IMU (Inertial Measurement Unit), high precision GPS RTK (Global Positioning Systems, Real-Time Kinematics), encoders, compass, and tilt sensors, to position them self and follow waypoints. A picture of a vehicle with these characteristics is shown in Figure 1. Its use is becoming more frequent for both intensive and extensive agriculture, in the precision agriculture context. For example, in USA or Argentina with millions of arable hectares is essential to have autonomous farm machines for handling and managing growth, quality, and yield of the crops.

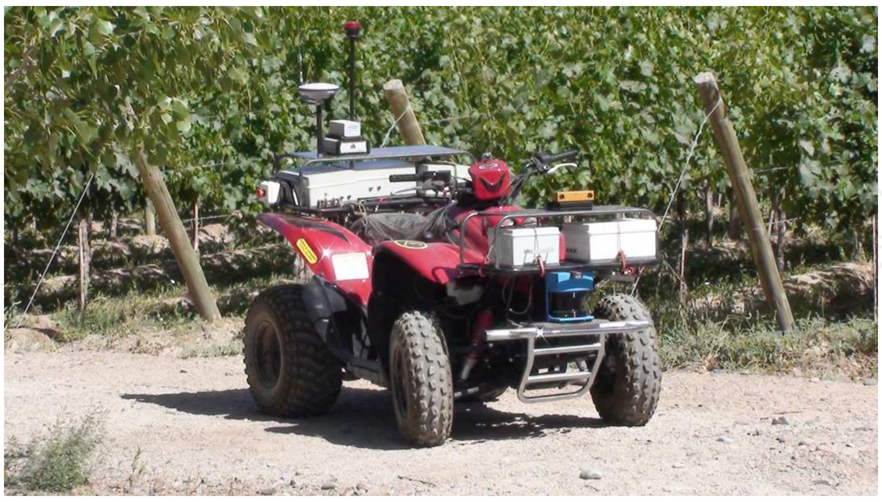


Fig. 1. Prototype of a UGV equipped with a number of sensors. This prototype belongs to the Instituto the Automática of the Universidad Nacional de San Juan.

[1] dpatino@inaut.unsj.edu.ar

The environment where these vehicles are used can be classified as:

- Structured or partially structured when it is well known and the motion can be planned in advance. In general, this is the case of navigation and guide of mobile robots.
- Not structured, when there are uncertainties which imply some on-line planning of the motion, this is the case of navigation and guide of robotic aerial vehicles.

In general, the objective of controlling the autonomous vehicles implies solving the problems of sensing, path planning and kinematic and dynamic control. Autonomy of a vehicle is related to determine its own position and velocity without external aids. Autonomy is very important to certain military vehicles and to civil vehicles operating in areas of inadequate radio-navigation coverage. Regarding the trajectory planning, there are many approaches (Aicardi et al., 1995). Many works have been published on the control of autonomous vehicles, mainly in the UGV or mobile robots. Some of them propose stable control algorithms which are based on Lyapunov theory (Singh & Fuller, 2001). Others have focused on optimization planning and control (Kuwata et al., 2005) and (Patiño et al., 2008).

In this paper we propose the use of ACDs to design autonomously an optimal path planning and control strategy for robotic unmanned vehicles, in particular for a mobile robot, following a previous work (Liu & Patiño, 1999a), and (Liu & Patiño, 1999b).

We consider a mobile robot with two actuated wheels and the autonomous control system is designed for kinematic and dynamic model. The kinematic mobile robot model for the so-called kinematic wheels under the nonholonomic constrain of *pure rolling* and *nonslipping*, is given by,

$$\dot{q} = S(q)v(t)\,, \tag{1}$$

Where $q(t), \dot{q}(t) \in \Re^3$ are defined as

$$q = [x, y, \theta]^T, \qquad \dot{q} = [\dot{x}, \dot{y}, \dot{\theta}]^T, \tag{2}$$

$x(t), y(t),$ and $\theta(t) \in \Re^3$ denote the linear position, and orientation respectively of the center of mass of the mobile vehicle; $\dot{x}(t), \dot{y}(t),$ denote the Cartesian components of the linear velocity of the vehicle; $\dot{\theta}(t),$ denotes the angular velocity of the mobile robot; the matrix $S(q) \in \Re^{3\times 2}$, is defined as,

$$S(q) = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0 \\ 0 & 1 \end{bmatrix}, \tag{3}$$

and the velocity vector $v(t) \in \Re^2$, is defined as

$$v = [v_l, w]^T\,, \tag{4}$$

with $v_l \in \Re$ denoting the constant straight line velocity, and $w(t) \in \Re$ is the angular velocity of the mobile robot.

Considering the dynamics of the car-driving device which contains a dc motor, a dc amplifier, and a gear transmission system,

$$w = \frac{Ka}{s^2 + bs + a} \cdot w_R\,, \tag{5}$$

where $w_R \in \Re$, is the reference angular velocity, and $K, a, b \in \Re^+$ are the car-driving device parameters.
The state of the mobile vehicle is given by (cf. Figure 1) the coordinates of the robot (x, y), the orientation of the vehicle, θ, and the actual turning rate of the robot, $\dot{\theta}$. The *control signal* is the *desired turning* rate of the mobile vehicle, w_R.

1.2 Control problem formulation

As was previously defined, the reference trajectory is generated via a reference vehicle which moves according to the following dynamic trajectory,

$$\dot{q}_R = S(q_R)v(t), \tag{6}$$

Where $S(\cdot)$ was defined in (3), $q_R = [x_R, y_R, \theta_R]^T \in \Re^3$ is the desired time-varying position and orientation trajectory, and $v_R = [v_l, w_l]^T \in \Re^2$ is the reference time-varying velocity. With regard to (5), it is assumed that the signal $v_R(t)$ is constructed to produce the desired motion, and that $v_R(t)$, $\dot{v}_R(t)$, $q_R(t)$, and $\dot{q}_R(t)$ are bounded for all time.
In general the vehicle motion control can be classified in: i) Positioning without prescribing orientation: in this case a final destination point is specified; ii) Positioning with prescribed orientation: in this case a destination point has to be achieved with a desired orientation; and iii) Path following: here, the path is defined through a sequence of waypoints.
In the first experiment, the *control objective* is limited to the first case, that is, given a reference point located at the workspace, (x_R, y_R), and considering the vehicle dynamical model, it is desired to obtain autonomously a *sequence of optimal control actions* (values of the turning rate) such that the vehicle achieves the target point as fast as possible (cf. Figure 2), and with minimum energy consumption. Since the mobile robot´s speed, v_l, is taken as constant, minimum-time control is equivalent to shortest-path control.
The design of the control system will be based on adaptive critic designs, in particular HDP (Werbos, 1992) and (Bellman, 1957). Next Section shows the background material needed for the present work.

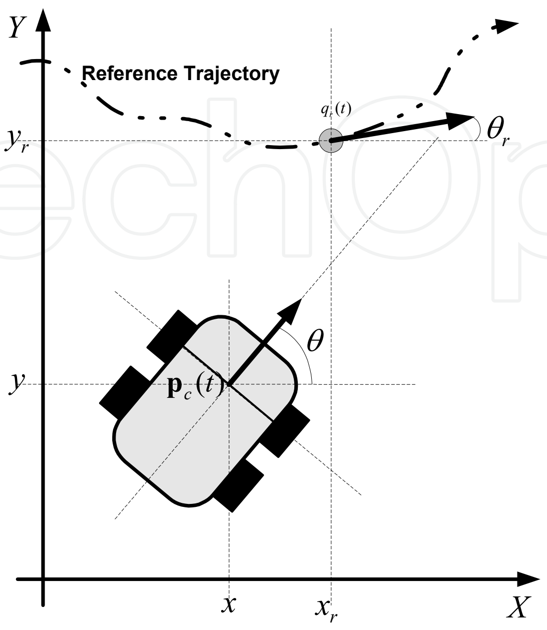


Fig. 1. The state of the unmanned ground vehicle.

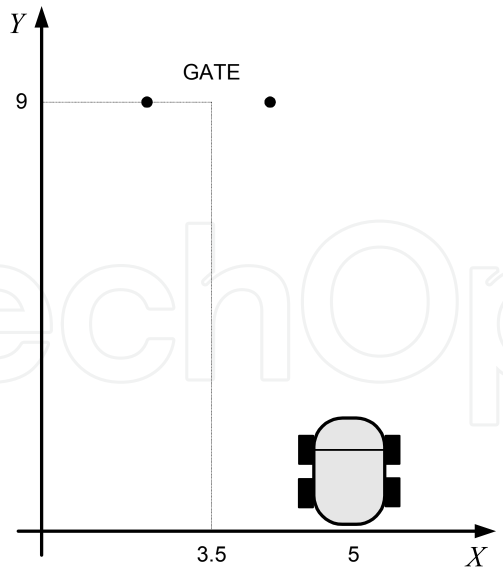


Fig. 2. The gate for the unmanned ground vehicle to go through.

2. Background in adaptive critic designs

2.1 Introduction to dynamic programming

Suppose that it is given a discrete-time nonlinear (time-varying) system,

$$x(k+1) = F[x(k),u(k),k] , \tag{7}$$

where, $x \in \Re^n$ represents the (complete) state vector of the system and $u \in \Re^m$ denotes the control action. Suppose that it is desired to minimize for (7) a performance index (or cost),

$$J[x(i),i] = \sum_{k=i}^{\infty} \gamma^{k-i} U[x(k),u(k),k], \tag{8}$$

where U is called the utility function or local cost function, and γ is the discount factor with $0 \le \gamma \le 1$. Note that J is dependent on the initial time i and the state $x(i)$, and it is referred to as the cost-to-go of the state $x(i)$. The objective is to choose the control sequence $u(k), k = i, i+1, \ldots$ so that the J function (the cost) in (8) is minimized. The cost in this case accumulates indefinitely; these kinds of problems are referred to as *infinite horizon problems* in Dynamic Programming. On the other hand, in finite horizon problems, the cost will accumulate over a finite number of steps. Dynamic programming is based on Bellman's *principle of optimality*, (Lewis & Syrnos, 1995), (Prokhorov & Wunsch, 1997), and establishes that an optimal (control) policy has the property that no matter what previous decisions (i.e., controls) have been, the remaining decisions must constitute an optimal policy with regard to the state resulting from those previous decisions.

Suppose that we have computed the optimal cost $J^*[x(k+1),k+1]$, from time $k+1$ to the terminal time for possible states $x(k+1)$, and that we have also found the optimal control sequences from time $k+1$ on. The optimal cost results when the optimal control sequence $u^*(k+1), u^*(k+2), \ldots,$ is applied to the system with initial state $x(k+1)$. Note that the optimal control sequence depends on $x(k+1)$. If we apply an arbitrary control $u(k)$ at time k and then use the known optimal control sequence from $(k+1)$ on, the resulting cost will be

$$J[x(k),k] = U[x(k),u(k),k] + \gamma J^*[x(k+1),k+1],$$

where, $x(k)$ is the state at time k and is determined by (2). According to Bellman, the optimal cost from time k on is equal to

$$\begin{aligned} J^*[x(k),k] &= \min_{u(k)} J[x(k),k] = \\ &= \min_{u(k)} \Big(U[x(k),u(k),k] + \gamma J^*[x(k+1),k+1] \Big). \end{aligned} \tag{9}$$

The optimal control $u^*(k)$ at time k is the $u(k)$ that achieves the minimum. Equation (9) is the principle of optimality for discrete-time systems. Its importance lies in the fact that it allows us to optimize over only one control vector at a time by working *backward* in time. Dynamic programming is a very useful tool in solving optimization and optimal control problems. In particular, it can easily be applied to nonlinear systems with constraints on the control and state variables, and arbitrary performance indexes.

2.2 Adaptive critic designs

In the computations in (9), whenever one knows the function J and the model F in (7), it is a simple problem in function minimization to pick the actions $u^*(k)$ which minimize J. However, due to the backward numerical process required, it is too computationally expensive to determine the exact J function for most real problems, even when the scales of the problems are considered to be small. Therefore, approximation methods are demanding in practice when performing dynamic programming (Werbos, 1992), (Bellman, 1957), (Balakrishnan & Biega, 1995).

Instead of solving for the value of J function for every possible state, one can use a function approximation structure such as a neural network to approximate the J function. There are three basic methods proposed in the literature for approximating the dynamic programming. They are collectively called Adaptive Critic Designs, which include Heuristic Dynamic Programming (HDP), Dual Heuristic Programming (DHP), and Globalized Dual Heuristic Programming (GDIHP) (Bellman, 1957), (Werbos, 1990), (Balakrishnan & Biega, 1995).

A typical adaptive critic design consists of three modules —Critic, Model, and Action. The present work considers the case where each module is a neural network; the designs in this case are referred to as neural network--based adaptive critic designs. The following introduces the HDP. In HDP (Werbos, 1990), (Werbos, 1992), (Lewis & Syrnos, 1995), (Balakrishnan & Biega, 1995, the critic network output estimates J function in equation (7). This is done by minimizing the following error measure over time,

$$\|E_1\| = \sum_k E_1(k) = \frac{1}{2}\sum_k [J(k) - U(k) - \gamma J(k+1)]^2 \tag{10}$$

where, $J(k) = J[x(k),t,W_C]$ and W_C represents the parameters of the critic network. The function U is chosen as a utility function which indicates the performance of the overall system (see examples in (Balakrishnan & Biega, 1995), (Werbos, 1990)). It is usually a function of $x(k)$, $u(k)$, and k, i.e., $U(k)=[x(k),u(k),k]$. When $E_1(k)=0$ for all k, (10) implies that

$$\begin{aligned} J(k) &= U(k)+\gamma J(k+1) = \\ &= U(k)+\gamma[U(k+1)+\gamma J(k+2)] = \ldots = \sum_{l=k}^{\infty}\gamma^{l-k}U(k) \end{aligned} \tag{11}$$

which is exactly the same as in dynamic programming [cf. (8)]. In Eq. (11), it is assumed that $J(k)<\infty$ which can usually be guaranteed by choosing the discount factor γ such that $0<\gamma<1$. The training samples for the critic network are obtained over a trajectory starting from $x(0)=x_0$ at $k=0$. The trajectory can be either over a fixed number of time steps [e.g., 300 consecutive points] or from $k=0$ until the final state is reached. The training process will be repeated until no more weight update is needed.
The weight update, during the pth training iteration, is given by

$$\begin{aligned} W_{C,i}^{(p+1)} &= W_{C,i}^{(p)} - \eta_1 \frac{\partial E_1(k)}{\partial W_{C,i}^{(p)}} = \\ &= W_{C,i}^{(p)} - \eta_1 [J(k)-U(k)-\gamma J(k+1)] \frac{\partial J(k)}{\partial W_{C,i}^{(p)}} \end{aligned} \tag{12}$$

where, $\eta_1>0$ is the learning rate and $W_{C,i}$, the ith component of W_C. Note that the gradient method is used in (12) and that the pth corresponds to certain time instant k [hence the use $E_1(k)$ in (12)]. The weight update can also be performed in batch mode, e.g., after the completion of each trajectory. The model network in an adaptive critic design predicts $x(k+1)$ given $x(k)$ and $u(k)$; it is needed for the computation of

$$J(k+1)=J[x(k+1),k+1,W_C^{(p-1)}]$$

in (12) for the weight update. The model network learns the mapping given in equation (7); it is trained previously off-line (Werbos, 1992), (Bellman, 1957), (Balakrishnan & Biega, 1995), or trained in parallel with the critic and action networks. Here, $J(k+1)$ is calculated using $W_C^{(p-1)}$ and its dependence on $W_C^{(p)}$ is not considered, according to (Liu & Patiño, 1999a). After the critic network's training is finished, the action network's training starts with the objective of minimizing $J(k+1)$. The action network generates an action signal $u(k)=[x(k),k,W_A]$; its training follows a similar procedure to the one for the critic network's training. The training process will be repeated until no more weight update is needed while keeping the critic network's weights fixed. During the pth training iteration, the weight update is given by

$$\begin{aligned} W_{A,i}^{(p+1)} &= W_{A,i}^{(p)} - \alpha_1 \frac{\partial J(k+1)}{\partial W_{A,i}^{(p)}} = \\ &= W_{A,i}^{(p)} - \alpha_1 \sum_{j=1}^{n} \frac{\partial J(k+1)}{\partial x_j(k+1)} \sum_{k=1}^{m} \frac{\partial x_j(k+1)}{\partial u_k(k)} \cdot \frac{\partial u_k(k)}{\partial W_{A,i}^{(p)}} \end{aligned} \tag{13}$$

where, $\alpha_1 > 0$. Again, the model network is required for the computation of $\partial x_i(k)/\partial u_k(k)$ in the above weight update. It can be seen in (13) that information is propagated backward through the critic network to the model network and then to the action network, as if three networks formed one large feedforward network. After action network's training cycle is completed, one may check its performance, then stop or continue the training procedure entering the critic network's training cycle again, if the performance is not acceptable yet.
It is emphasized that in the methods described above, the knowledge of desired target values for the function J and the action signal $u(k)$ is not required in the neural net-work training. In conventional applications of neural networks for function approximation, the knowledge of the desired target values of the function to be approximated is required. It should also be emphasized that the nature of the present methodology is to iteratively build a link between present actions and future consequences via an estimate of the utility function J.

3. Main results

A simulation study has been carried out using the mobile vehicle model presented in Section I. The set of parameters for this vehicle model used are the following: $K = 0.45$, $a = 102.6$, $b = 9.21$, $vl = 0.2m/s$. The three networks (critic, action, and model) are all implemented using multilayer feedforward neural networks. Each neural network has six inputs, $(x_R, y_R, x, y, \theta, w)$, where x_R and y_R denote the desired target gate. The critic network output J, the action network output w_R, and the model network is trained according to equation (1) and (5). The training samples for the critic network are obtained over trajectories starting from $x(0) = 0.5$ at $k = 0$, initial position of the vehicle, and a reference point located at position $(8m, 3.5m)$.
The discount factor is chosen as $\gamma = 0.8$, and the utility function is chosen as

$$U(k) = \frac{1}{2}\left[q\left(\tilde{x}^2(k) + \tilde{y}^2(k)\right) + rw_R^2(k)\right]$$

where, $\tilde{x} = x - x_R$ and $\tilde{y} = y - y_R$ are position errors with respect to the target point (x, y), and $q > 0$ and $r > 0$ are positive weight constants. As described previously, the training takes place in two stage: the training of model network, and then the training of critic network and action network. The objective for the training of the critic network is to match $J(k)$ with $U(k) + \gamma J(k+1)$. The objective for the training of the action network la minimize $J(k+1)$. The procedures for the training of critic and action networks are similar, and they are repeated iteratively. Figure 3 shows the result for the mobile vehicle when reaching the reference point, after 10 trials (learning cycles), and Figure 4 passing through one gate from two different initial conditions. Figure 5 shows the result for the mobile vehicle through two gates.
A second simulation study was performed using the kinematic model of both the robot and the reference trajectory virtual robot. In his case the mathematical model of the systems are defined as in Equations (1), (2) and (3) under the non-holonomic restriction

$$\theta(t) = tan^{-1}\frac{\dot{y}(t)}{\dot{x}(t)}. \quad (14)$$

In this case both the linear and angular velocities are variable, and the mobile robot follows a reference trajectory given by the equations

$$\begin{aligned}\dot{x}_r &= v_r \cos\theta_r \\ \dot{y}_r &= v_r \sin\theta_r \\ \dot{\theta}_r &= \omega_r\end{aligned} . \quad (15)$$

Once the reference trajectory is stated, the tracking error can be defined as (Kanayama et al. 1990)

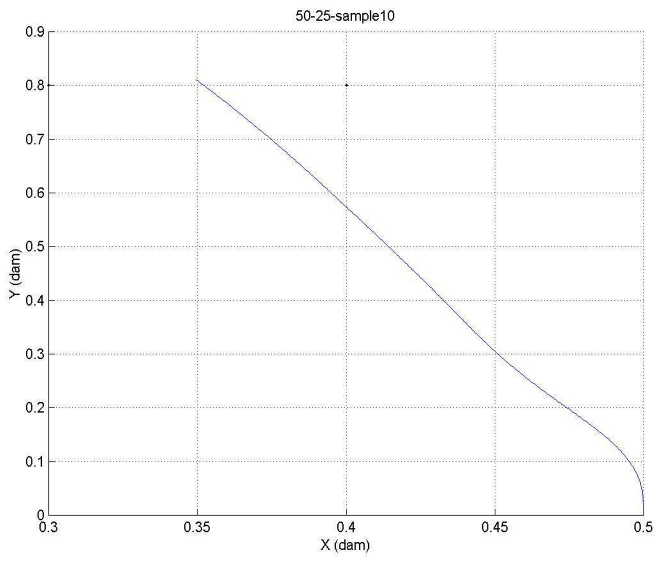


Fig. 3. Result for passing through the gate.

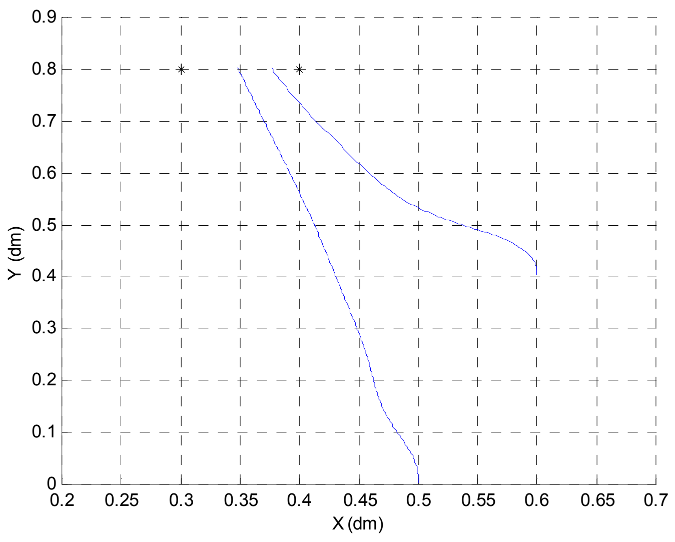


Fig. 4. Result for passing through one gate from two different initial conditions.

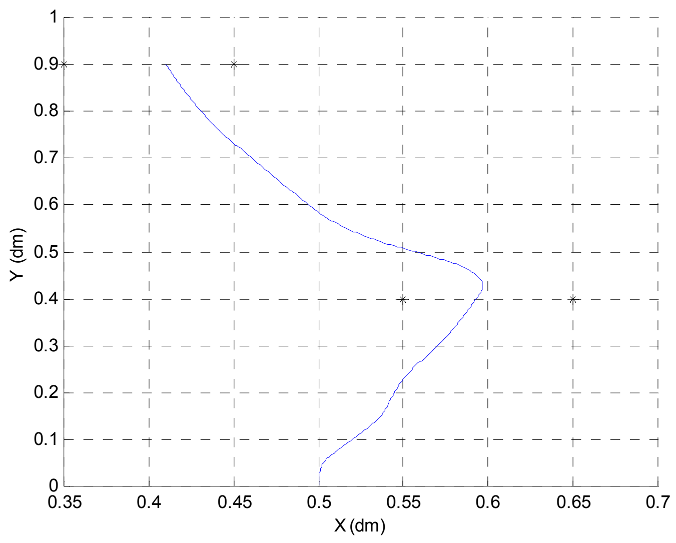


Fig. 5. Result for passing through two gates.

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix}, \tag{16}$$

and combining equations (1), (14) and (15), the tracking error model is

$$\begin{aligned} \dot{x}_e &= \omega y_e - v_r + v_r \cos\theta_e \\ \dot{y}_e &= -\omega x_e + v_r \sin\theta_e \quad . \\ \dot{\theta}_e &= \omega_r - \omega \end{aligned} \tag{17}$$

Figure 6 shows all the variables presented in the previous equations.

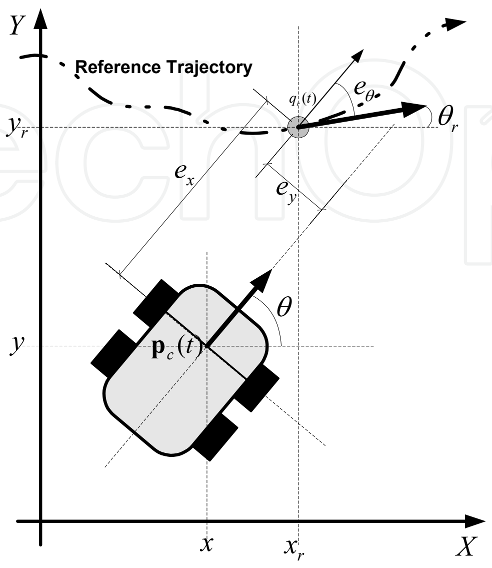


Fig. 6. Representation of the robot and virtual robot state variables.

For the sake of simplicity, a new set of coordinates is chosen, defined as

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \theta_e \\ y_e \\ -x_e \end{bmatrix}, \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} = \begin{bmatrix} \omega_r - \omega \\ v - v_r \cos x_0 \end{bmatrix}. \tag{18}$$

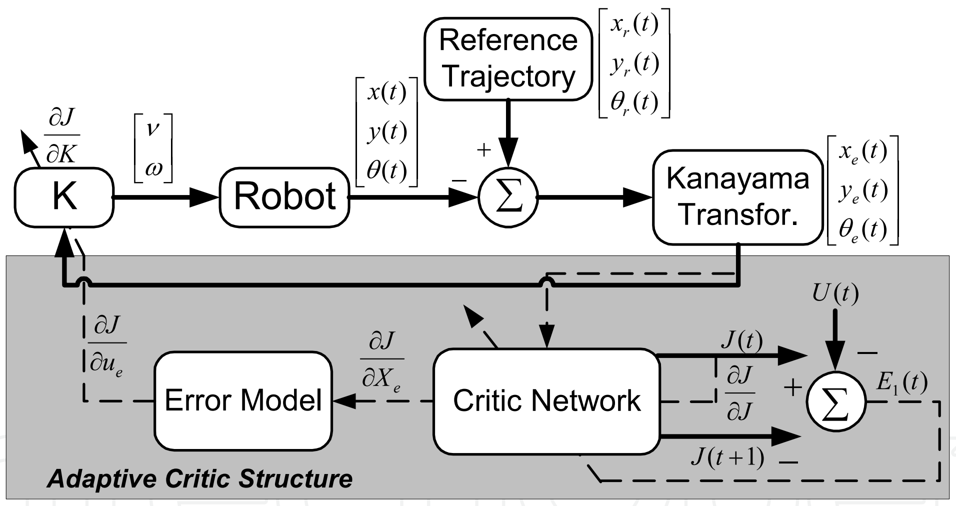
Finally, equation (16) can be written as

$$\begin{aligned} \dot{x}_0 &= u_0 \\ \dot{x}_1 &= (\omega_r - u_1)x_2 + v_r \sin x_0 \, . \\ \dot{x}_2 &= -(\omega_r - u_0)x_1 + u_1 \end{aligned} \tag{19}$$

With this change of coordinates the tracking problem is turned into a regulation one.
In this experiment the control action is given by

$$\begin{bmatrix} u_0 \\ u_1 \end{bmatrix} = K \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}. \tag{20}$$

Figure 7 shows the overall block diagram of the control system.



Reference Trajectory
$[x_r(t), y_r(t), \theta_r(t)]$
$\frac{\partial J}{\partial K}$
$[v, \omega]$
$[x(t), y(t), \theta(t)]$
+
−
K
Robot
Σ
Kanayama Transfor.
$[x_e(t), y_e(t), \theta_e(t)]$
$U(t)$
$\frac{\partial J}{\partial u_e}$
$\frac{\partial J}{\partial X_e}$
$J(t)$
−
$\frac{\partial J}{\partial J}$
+
$E_1(t)$
Error Model
Critic Network
$J(t+1)$
−
Adaptive Critic Structure

Fig. 7. Block diagram of the control system.

The virtual robot describes a circular trajectory given by the equations:

$$\begin{aligned} x_r &= x_c + R\sin(ct) \\ y_r &= y_c - R\cos(ct) \\ \theta_r &= \tan^{-1}\frac{\dot{y}_r}{\dot{x}_r} = ct \\ v_r &= \sqrt{\dot{x}_r^2 + y_r^2} = cR \\ \omega_r &= \dot{\theta}_r = c \end{aligned}$$

Where $(x_c, y_c) = (0,0)$ is the center of the trajectory, $R = 0.5$ is the radius, $c = 0.2$ and t is the time.
The utility function given for this experiment is

$$U(k) = \frac{1}{2}\left[\left(x^T(k)Qx(k)\right) + u^T(k)Ru(k)\right],$$

with

$$Q = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix} \text{ and } R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

The experience begins with a K matrix stable but not adjusted, given the results shown in Figures 8 and 9.
After 11 iterations of the training algorithm, the control system guides the robot to track the reference trajectory, as can be seen in Figures 10 and 11.

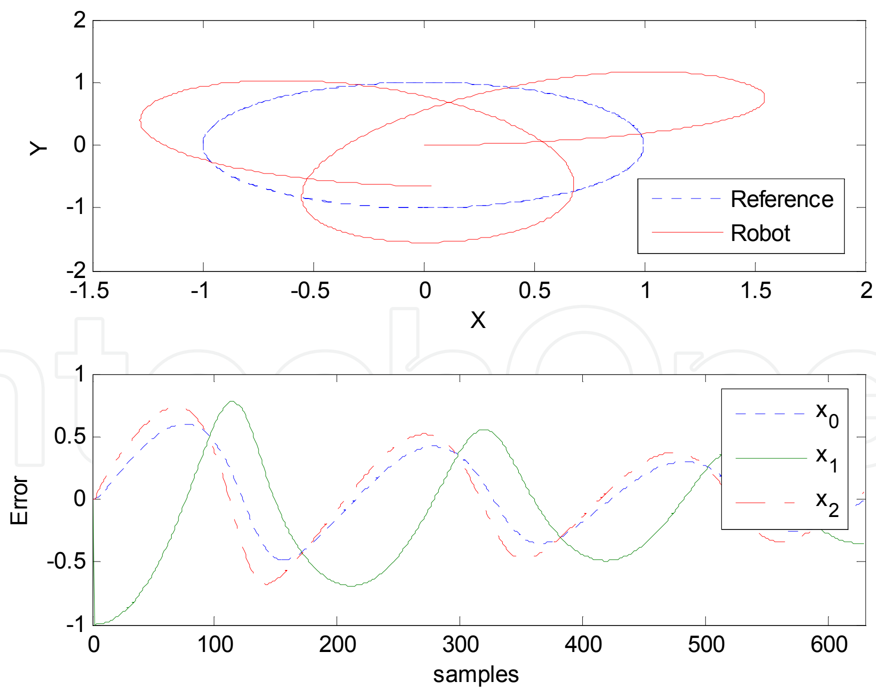


Fig. 8 Reference trajectory and initial performance of the robot.

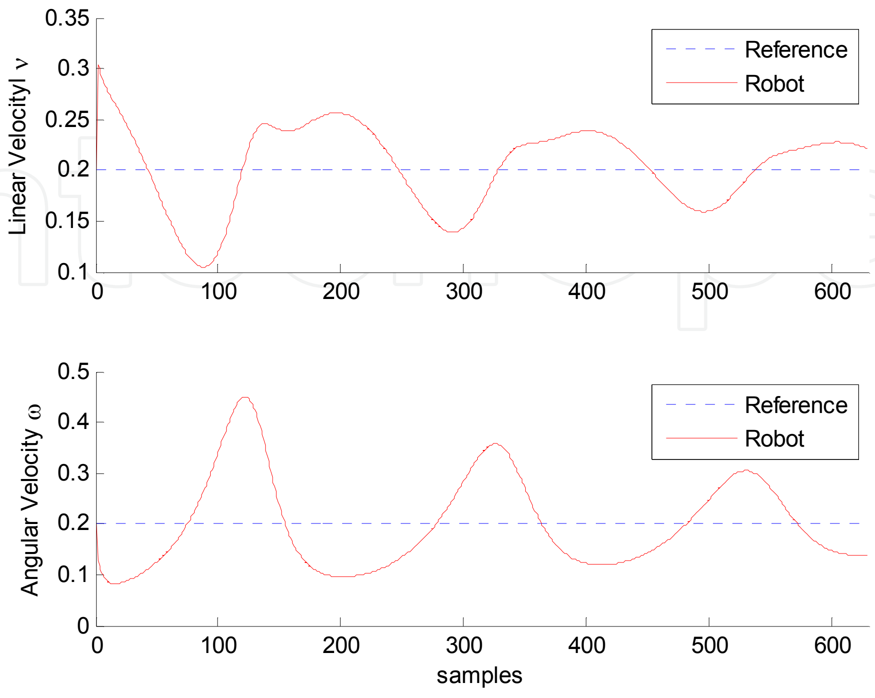


Fig. 9. Linear and angular velocity for the first trial.

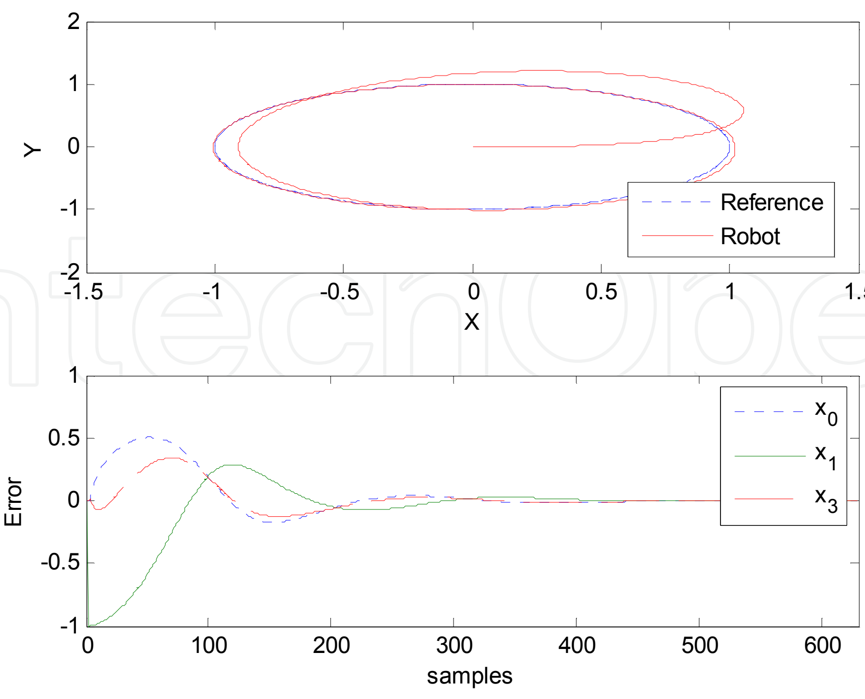


Fig. 10. Performance of the control system after 11 training iterations.

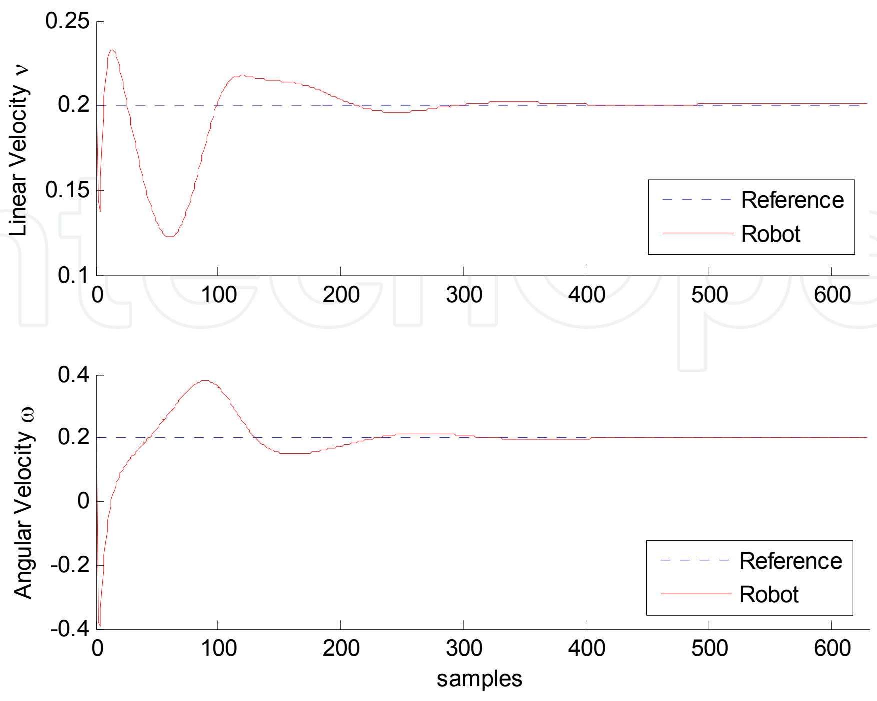


Fig. 11. Linear and angular velocity after 11 training iterations.

4. Conclusions

A solution to the problem of generating autonomously optimal control action sequence for a mobile robot control based on Adaptive Critic Designs approach has been presented. The proposed controller based on adaptive critic designs learns to guide the robot to a final point autonomously. It has been shown that using this technique we can obtain near optimal control actions which requires no external training data and gives an optimal control law for the entire range of operation. This work is extensible to UAV, assuming that is flying at a constant altitude, so the mission will be restricted to a planar motion around of a target point, and the kinematic equation of motion is similar to a UGV, see (Patiño et al., 20008). Future directions of research will be oriented to reach a final point with orientation with application to UAV. In addition, the problem of obstacle avoidance will be addressed. It will be also researched with other structures as DHP and GDHP, to use different local cost functions, and to consolidate formally a systematic design principle. From a theoretical point of view the efforts will be placed on the robustness issues of optimal control systems using adaptive critic designs.

5. Acknowledgements

This work was supported by Universidad Nacional de San Juan, and Agency for Scientific and Technological Promotion (ANPCyT), Argentina, under grant PICT/04 N° 21592, and PICT/05 Start-up N° 35398.

6. References

Aicardi M., Casalino G., Bicchi A., and Balestrino A. (1995). Closed Loop Steering of Unicycle-Like Vehicles via Lyapunov Techniques. *IEEE Robotics and Automation Magazine*, Vol. 2, No. 1, pp. 27-35.

Balakrishnan S. N., and Biega V. (1995). Adaptive critic based neural networks for control (low order systems applications). *Proceedings 1995 American Control Conference*, 335-339, Seattle, WA, 1995.

Bellman R. E., . (1957). Dynamic Programming*Princeton University Press*, Princeton, NJ.

Kanayama, Yukata, Yoshihiko Kimura, Fumio Miyazaki and Tetsuo Noguchi (1990). A stable tracking control method for an autonomous mobile robot. Proceedings IEEE International Conference on Robotics and Automation 1, 384-389.

Kuwata Y., Schouwenaars T, Richards A., and How J. (2005). Robust Constrained Receding Horizon Control for Trajectory Planning. *AIAA Guidance, Navigation, and Control Conference*, San Francisco, California, August.

Lewis F. and Syrnos V. L. (1995). Optimal Control. *John Wiley*, New York, NY, 1995.

Liu D. and Patiño H. D. (1999a). Adaptive Critic Designs for Self-Learning Ship Steering Control. *Proceedings of the 1999 IEEE, International Symposium on Intelligent Control/Intelligent Systems and Semiotics,* Cambridge, MA September, pp. 15-17.

Liu D, and Patiño H. D. (1999b) A Self-Learning Ship Steering Controller Based on Adaptive Critic Designs. *14th World Congress of IFAC*, Beijing, P.R. China, pp. 367-372.

Patiño H. D., Tosetti S., and Martinez M. (2008). A New Kinematics-Based Tracking Control for UAV using a Reference Virtual Aerial Robot Approach, *2º Conferencia/Workshop de Vehículos/Sistemas No-Tripulados (UVS) de América Latina*, Ciudad de Panamá, Panamá, Aug. 2008, pp. 5-7.

Prokhorov D. V., and Wunsch D. (1997). Adaptive Critic Designs, *IEEE Tranactions on Neural Networks*, Vol. 8, No. 5, Sep. 1997, pp. 997-1007.

Singh L. and Fuller J. (2001).Trajectory generation for a UAV in Urban Terrain, using nonlinear MPC, *Proceedings of American Control Conference,* Arlington, VA, 2001, pp. 2301-08.

Werbos P. (1990a). Consistency of HDP applied to a simple reinforcement learning problem. *Neural Networks*, 3, 1990, pp. 179-189.

Werbos P. (1990b). A menu of designs for reinforcement learning over time. *Neural Networks for Control* (W.T. Miller III, R.S. Sutton, and P.J. Werbos, Eds.). Chapter 3. The MIT Press, Cambridge, MA, 1990.

Werbos P. (1992a). Neurocontrol and supervised learning: An overview and evaluation. In: Handbook of Intelligent Control. *Neural, Fuzzy, and Adaptive Approaches* (D.A. White aud D.A. Sofge, Eds.). Chapter 3, Van Nostrand Reinhold, New York, NY, 1992.

Werbos P. (1992b).Approximate dynamic programming for real-time control and neural modeling. *Handbook of Intelligent Control Neural, Fuzzy, and Adaptive Approaches* (D.A. White and D.A. Sofge, Eds.). Chapter 13. Van Nostrand Reinhold, New York, NY, 1992.