Proceedings of 2004 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 28 - October 2, 2004, Sendai, Japan

# Trajectory Tracking Control of a Rotational Joint using Feature-Based Categorization Learning

Alejandro Agostini
Institut de Robòtica I Informàtica Industrial
Barcelona, Spain
agostini@iri.upc.es

Enric Celaya
Institut de Robòtica I Informàtica Industrial
Barcelona, Spain
celaya@iri.upc.es

*Abstract*— **Real world robot applications have to cope with large variations in the operating conditions due to the variability and unpredictability of the environment and its interaction with the robot. Performing an adequate control using conventional control techniques, that require the model of the plant and some knowledge about the influence of the environment, could be almost impossible. An alternative to traditional control techniques is to use an automatic learning system that uses previous experience to learn an adequate control policy. Learning by experience has been formalized in the field of Reinforcement Learning. But the application of Reinforcement Learning techniques in complex environments is only feasible when some generalization can be made in order to reduce the required amount of experience. This work presents an algorithm that performs a kind of generalization called categorization. This algorithm is able to perform efficient generalization of the observed situations, and learn accurate control policies in a short time without any previous knowledge of the plant and without the need of any kind of traditional control technique. Its performance is evaluated on the trajectory tracking control with simulated DC motors and compared with PID systems specifically tuned for the same problem.**

***Keywords- Reinforcement learning; Trajectory tracking control; categorization.***

## I. INTRODUCTION

Robotics applications are widening with the development of new technologies and the improvements reached in computer science. Day by day new difficult achievements are expected from robotics. A shift in the operation scenarios from structured environments to natural unstructured ones becomes a necessity. In order to cope with the control of robot locomotion in this kind of environment the application of conventional control techniques are unsuitable for the control task due to the inherent variability of natural environments and the difficulty to model each aspect of them. An example of this is the problem of the walk control of a legged robot, as we are facing to perform autonomous navigation tasks outdoors [1], [2]. In legged robots the requirements for the control of each joint include fast execution of long as well as short displacements, accuracy in the final position without overshooting, and fast response to varying loads and inertias. Complying with the mentioned requirements with a standard control method such as PID, requires an adequate model of the system under control, and a specific tuning for each possible working condition, what is impractical and difficult to accomplish.

To cope with this control challenge many control algorithms have been proposed [3], [4], [5], [6], [7], with different degree of success, but all of them have diverse problems when variations in the dynamics of the system are present and when non deterministic or random disturbances take place. Some works propose specialized control techniques to deal with dynamics variations and system perturbation [3] but they have the problem that the disturbance must fulfill some requirements that almost never take place in the task of robot walking. Other works mixed control techniques with machine learning techniques [4], [5], [6]. For instance in [6] a PD control system for a torque control and trajectory tracking is complemented with a feed-forward component calculated using a learned inverse dynamics model of the system in order to compensate torque variations in the trajectory tracking control. These techniques improve the control task, nevertheless they are associated to conventional control systems that require some plant information in order to tune their parameters.

An alternative to the mentioned control systems is to use an automatic learning system that uses previous experiences to improve its performance with time without any previous knowledge of the model of the plant. Learning from experience is the subject of Reinforcement Learning (RL) [7]. It is well known that the application of RL to non-toy problems suffers the problem of the curse of dimensionality [8], that is, the number of states that must be stored and experienced grows exponentially with the number of state variables. To allow the application of learning in complex tasks a new approach has been devised [9] based on the assumption that what has to be learnt is not a completely arbitrary function of the state vector, but satisfies what is called the categorizability hypothesis, which states that, in order to decide what action must be executed next, not all dimensions of the state vector are always relevant in all situations, but only relatively small subsets of them become relevant in each particular situation. In fact this is actually the case in most real world problems. The development of this ideas lead to the categorization and learning algorithm (CL algorithm), whose fundamentals are introduced below.

In this work we apply the categorization and learning algorithm to the problem of the trajectory tracking control

0-7803-8463-6/04/$20.00 ©2004 IEEE

of DC motors. The results obtained in simulations show that it is possible to learn, in short time, control policies with performances comparable to that of a traditional PID control, with the advantage that the learning system does not require the individual tuning of the parameters for each case. Moreover, the algorithm makes efficient generalization using the categorizability of the environment and the control is performed without the help of any traditional control technique making this learning system very promising for the control task in walking robots locomotion.

In Section 2 the CL algorithm is presented. Section 3 states the control problem we face, and, in Section 4, its formulation for the CL algorithm is introduced. Section 5 presents the results and compares them with the solutions of a tuned PID control system. Conclusions are in Section 6.

## II. The Learning Algorithm

The fundamental aspects of the CL algorithm are summarized in this section. For a more detailed explanation see [9] and [10].

It is assumed that the world is perceived through a set of $n$ binary feature detectors $f_i$ $i$=1...$n$. A partial view of order $m$, denoted by $v(f_{i1},..., f_{im})$, is defined as a virtual feature detector that becomes active when its $m$ component feature detectors are simultaneously active. The categorization process starts with the initial set of feature detectors (all partial views of order 1), and progressively builds partial views of higher order, depending on the requirements of the learning task.

A $q_v(a)$ value is stored for each partial view $v$ and each possible action $a$, that, in a way similar to the usual Q-learning algorithm, estimates the average discounted reward obtained from the execution of action $a$ when $v$ is active. Two more values are stored for each partial view and action:

- $e_v(a)$, the estimated average absolute error of $q_v(a)$, that provides a measure of the dispersion of the actual $q$ value obtained when the partial view $v$ was active. We consider that a partial view with a value $q_v(a)$ and error $e_v(a)$ predicts that executing action $a$ when $v$ is active will result in a $q$ value in the interval $I_v(a)$=[$q_v(a)$-2$e_v(a)$, $q_v(a)$+2 $e_v(a)$].
- $i_v(a)$, the confidence index, that estimates how much action $a$ has been tried when $v$ was active and resulted in a value of $q$ according to the prediction. This is used to estimate a confidence value for $q_v(a)$ and $e_v(a)$ using a monotonically increasing function with saturation value $\beta$:

$$c_v(a) = \min\{\beta, confidence\_function(i_v(a))\} \quad (1)$$

The value for which the *confidence_function*( ) reaches the saturation value is controlled by a user defined parameter $\eta$. In this work, a proportional relation is considered.

### A. Action Selection

As in the usual Q-learning algorithm, we must determine, for each situation, the action that maximizes the expected $q$ value. The problem in our case is that in a given situation we may have many different predictions for the same action: one for each active partial view. To address this problem we define the relevance $\rho_v(a)$ of partial view $v$ for action $a$, as

$$\rho_v(a) = \frac{e_v(a) - e_{max}(a)}{e_{min}(a) - e_{max}(a)} \quad (2)$$

where $e_{min}(a)$ and $e_{max}(a)$ are the minimum and maximum prediction errors for action $a$ considering all the active partial views.

The relevance $\rho_v(a)$ is a relative estimation of how precisely the $q$ value for action $a$ can be predicted by the partial view $v$. More relevant partial views will provide more accurate reward predictions for the action $a$ and, therefore, they will be preferred to estimate its result. On the other hand, not all partial views will have the same confidence, so that we can not simply take the partial view with highest relevance to predict the result of an action. To take into account both, the relevance and the confidence of the prediction, we define the winner partial view for action $a$ in a given situation $V$, as the active partial view for which the product $\rho_v(a).c_v(a)$ is maximum:

$$winner(V,a) = \arg\max_{\forall v \in V}\{\rho_v(a) \cdot c_v(a)\} \quad (3)$$

where $V$ is the set of active partial views. In this way, the $q$ prediction for an action in a given situation will be obtained from the winner partial view for this action.

To get an actual $q$ prediction from the winner partial view, two sources of indeterminacy are considered: On the first place, since each partial view predicts that the $q$ value is expected to lay in the interval $I_v(a)$, some value in this interval is selected at random as initial guess:

$$i\_guess(a) = \text{rand}(I_v(a)) \quad (4)$$

On the second place, a noise term is added to account for the uncertainty of the values stored in the partial view, as evaluated by the confidence:

$$guess(a) = c_v(a)\, i\_guess(a) + (1 - c_v(a))\, \text{rand}(q_{min}, q_{max}) \quad (5)$$

where $q_{min}$ and $q_{max}$ are the minimum and maximum $q$ values actually obtained so far in the learning process. Once a guess is obtained for each of the actions that are executable in the current situation, the action with highest guess is selected for execution. Note that this strategy implements an adaptive form of exploration: actions with low confidence always have some opportunity to be executed even with low $q$ predictions, but exploratory actions have little chances to occur in a situation in which

there is a strong confidence on the prediction of the $q$ values for all actions.

### B. Statistics Update

After the execution of an action, a reward $r$ is obtained and a new situation $V$ is perceived, so that the actual $q$ obtained from the execution of action $a$ is computed using a Bellman like equation:

$$q = r + \gamma . \max_{\forall a} \{ q_v(a) \mid v = winner(V, a) \} \quad (6)$$

where $\gamma$ is the discount factor. This information is used to update the estimated values for the executed action of all partial views that were active in the last situation. The $q_v(a)$ and $e_v(a)$ values are updated with identical schemas:

$$q_v(a) = c_v(a)\, q_v(a) + (1 - c_v(a))\, q \quad (7)$$

$$e_v(a) = c_v(a)\, e_v(a) + (1 - c_v(a)) \mid q - q_v(a) \mid \quad (8)$$

Note that the confidence estimation is used in the learning rate parameter, so that values with low confidence are shifted towards the observed value faster than values with higher confidence.

Finally, each confidence index $i_v(a)$ is increased by one if the actual $q$ value lies in the predicted interval $I_v(a)$ and decreased in a proportion of 0.2 when this $q$ falls outside it. With this updating policy, a partial view that repeatedly makes correct predictions will gradually reach high confidence values, but a single wrong prediction will cause a significant reduction on its confidence.

### C. Partial View Generation

If the prediction of the $q$ value is too inaccurate, $\tau$ new partial views are created to help improving the prediction in the future. For this matter, a prediction is considered inaccurate when the absolute difference between the predicted value $q_v(a)$ and $q$ is higher than a user defined amount $\delta$.

New partial views are created by combination of pairs of already existing partial views, randomly chosen among those that were active in the last situation. This random selection is biased towards the partial views with higher confidence and with better prediction of $q$.

To avoid an undesired proliferation of partial views in the system, their number is limited to a threshold $\mu$, a parameter of the system whose appropriate value depends on how much categorizable, in the sense we defined above, is the environment. To comply with this threshold, it is necessary to remove partial views when its number grows above $\mu$. There are two different elimination criteria: redundancy and utility.

A partial view is considered redundant when its reward predictions are too similar to the reward predictions of any of the partial views composed by a subset of its features [10]:

$$redundancy(v) = \min_{\forall a} \{ \max_{\forall v' \subset v} \{ sim(I_v(a), I_{v'}(a)) \} \} \quad (9)$$

where,

$$sim(I_v(a), I_{v'}(a)) = \frac{\left| I_v(a) \cap I_{v'}(a) \right|}{\max\{ \left| I_v(a) \right|, \left| I_{v'}(a) \right| \}} \quad (10)$$

Partial views with redundancy higher than a value $\lambda$ are eliminated from the system.

On the other hand, the utility of a partial view is used to eliminate partial views that are not redundant but appear to be less useful for the system. We consider that a partial view is less useful for the system if it has low relevance $\rho_v(a)$ and high confidence $c_v(a)$ for all the possible actions. So that, the utility of a partial view $v$ is defined as,

$$u(v) = \max_{\forall a} \left\{ \frac{\rho_v(a)}{c_v(a)} \right\} \quad (11)$$

## III. The Control Problem

### A. Generation of Reference Trajectories

In order to create the training set for the learning process, we generate random trajectory samples, defined as relationships time/angular-position for the motor shaft, that we call subtrajectories. Each subtrajectory starts at the final angular position of the previous one. The duration of each subtrajectory is selected randomly in the interval [2..5] (s). The range of angular positions for subtrajectory generation is limited to [-100π...100π].

In accordance with the characteristics of the subtrajectories generation, and after finding the Fourier Transform of them, we found that the maximum frequency component of these subtrajectories is about 5 Hz. The Nyquist theorem [11] states that, in order to keep all the information of the sampled signal, the sample frequency should be at least twice the maximum frequency component. Thus, we chose a frequency of 20 Hz (Δt of 50 (ms)) for action execution and learning system update.

### B. Motors Modeling

Two motors were modeled for the control task, the Maxon 118800 of nominal power rating of 70 (Watt) and the Maxon 118769 of nominal power rating of 18 (Watt) [12]. Some considerations are made for model construction in accordance with the manual specifications and motor equations [12]: the friction torque is independent of the motor speed and is calculated as the torque in the shaft when a nominal voltage is applied and no load is present. On the other hand, the inductance of the rotor is so small that can be neglected. With these assumptions the motor equations are:

$$J_R \ddot{\theta} + M_R = K_M i \quad (12)$$

$$Ri = V - K_M \dot{\theta} \quad (13)$$

where $V$ is the input voltage and $\theta$ is the output angular position. Table I shows the values of the parameters for the two motor models.

TABLE I. MAXON MOTORS PARAMETERS

| Parameter | Motor Model | |
|---|---|---|
| | *118800* | *118769* |
| $J_R$ (kg.m$^2$) rotor inertia | 65 E-7 | 10.8 E-7 |
| $M_R$ (Nm) is the friction torque | 3.98 E-3 | 1.082 E-3 |
| $K_M$ (Nm/A) is the torque constant | 56 E-3 | 21.3 E-3 |
| R ($\Omega$) is the terminal resistance | 2,75 | 2.27 |

In order to implement the simulation, an interval of $dt$=0.001 (s) was considered. This value is small enough to obtain good simulation results in accordance to the system's time constants. The set of possible actions considered for the learning system consists in a number of discrete input voltage values chosen in accordance with the motors data sheets. The nominal voltage for the motor 118800 is 42 ($V$), so we chose a range of variation of [-48.. 48] ($V$), and for the motor 118769, with 24 ($V$) of nominal voltage, we chose a range of [-28..28] ($V$). In both cases a discretization of 16 segments equally spaced was selected.

### C. PID Control

In order to evaluate the obtained results, a PID control system was implemented for performance comparison with the CL algorithm. The tuning of the proportional gain $K_p$, derivation time $T_d$ and integral time $T_i$ was made using the Ziegler-Nichols rule, in its second method [12]. Table II summarizes the tuned parameters for the control of motors 118800 and 118769.

TABLE II. PID TUNED PARAMETERS FOR MAXON MOTORS

| Case | $K_p$ | $T_d$ | $T_i$ |
|---|---|---|---|
| 118800 | 1.5 | 0.1$\Delta$t | 2$\Delta$t |
| 118769 | 0.58 | 0.02$\Delta$t | 2$\Delta$t |

For the control test using the PID system, also a sample period of 50 (ms) was used, and the voltages applied were discretized in the same way as with the CL algorithm.

## IV. PROBLEM FORMULATION

### A. CL Algorithm Implementation

The features considered in the learning algorithm for the motor control problem were the angular speed and acceleration, the difference between the current motor position and the next desired trajectory position, and the difference between the current angular speed and the next desired trajectory speed. Since these are continuous variables, a discretization is required in order to apply the CL algorithm, which demands binary feature detectors. We choose a logarithmic scale for those features that give information about the difference with respect to the next desired point of the trajectory. This logarithmic scale improves precision when the system approaches the trajectory, enabling a more accurate tracking of it. The ranges selected are in accordance with the control problem requirements. We consider 40 discretization intervals for each feature.

In addition to the mentioned features, we include a redundant feature that again consists in the difference between the current position of the motor and the next desired trajectory position, but with less discretization intervals. This feature covers the working space with fewer segments, so that, with relatively few experiences, the system can learn a rough knowledge of the motor response in most situations, favoring the convergence at early stages of the learning process.

After performing some experiments we found an adequate set of learning parameters for both motor control learning processes: $\eta$=20, $\beta$=0.95, $\delta$=50, $\gamma$=0.5, $\lambda$=0.9 and $\mu$=500.

### B. The Reward Function

Our goal is to learn a control policy to follow the reference trajectory as close as possible. So, a naive reward function could be one that depends only on the distance from the motor position to the current reference trajectory position. But, with this reward, the approach to the desired trajectory would be jagged because it does not take into account the future evolution of the trajectory. So, we consider, in addition to current position, the current angular speed of the motor and of the reference trajectory, and define an approaching target position, $\theta_{target}$, in order to reach the reference trajectory smoothly:

$$\theta_{target}(t) = \frac{[\theta_M(t-\Delta t) - \theta_T(t)] + \dot{\theta}_T(t)\Delta t}{2} + \theta_T(t) \qquad (14)$$

$$r(t) = -\left|\theta_M(t) - \theta_{target}(t)\right| \qquad (15)$$

where $\theta_M$ is the motor angular position, and $\theta_T$ is the trajectory angular position.

## V. RESULTS

All the control performances are evaluated on trajectories composed by five randomly generated subtrajectories.

### A. Control of motor 118800

Figure 1 shows the performance of the tuned PID control compared with that of the CL algorithm trained with 25000 iterations. To better appreciate the results, the respective tracking errors are shown in a different scale.

As we can see, the performance reached by the CL algorithm is comparable with that of a tuned PID control. This is a significant result considering that the training was conducted for no more than 25000 iterations, what corresponds to only 20 minutes of motor operation.

In this case, the parameter that determines the maximum number of partial views allowed in the system ($\mu$), was set to 500. This is very low compared with the large number of possible states that a conventional reinforcement learning algorithm would need considering the features available (about 3 million states). This fact demonstrates the high degree of categorization reached by the CL algorithm.

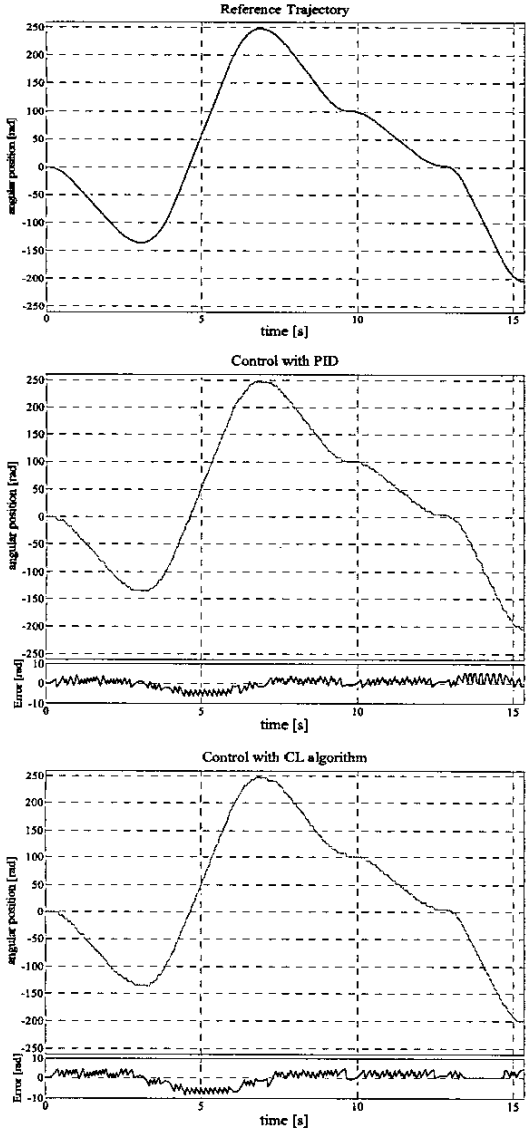


Figure 1. Reference trajectory and comparison between PID and CL algorithm for motor 118800 with their respective tracking errors.

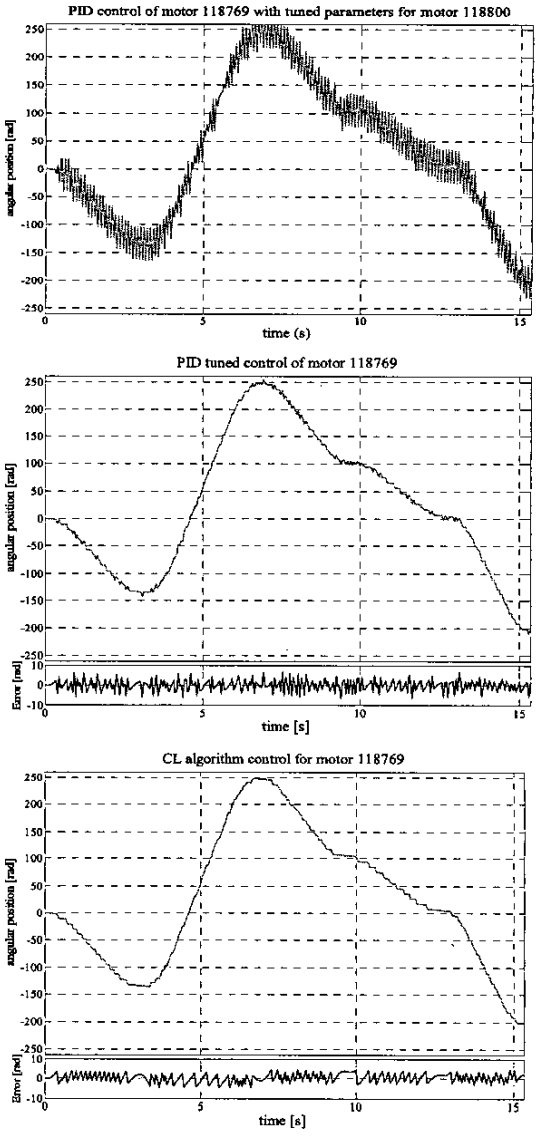


Figure 2. PID control of motor 118769 with parameters tuned for motor 118800 and comparison of PID tuned control versus CL algorithm control of motor 118769.

### B. *Control of Motor 118769*

In a second experiment we trained the CL algorithm to control the 118769 motor using the same problem formulation as in the case of motor 118800 (Figure 2).

In order to show the necessity of re-tuning the parameters of the PID, we also show in Figure 2 the results obtained with the PID tuned for the motor 118800 under the motor 118769. As expected, the system presents an unstable oscillatory behavior due to the change of system parameters. New tuned parameters of the PID system to control motor 118769 are computed as shown in Table II.

Note that the proportional gain $K_p$ decreased about three times in comparison with the $K_p$ to control the 118800 motor.

The results obtained with the controller learned with the CL algorithm after only 25,000 iterations are again comparable to those obtained with a tuned PID system.

It is important to note that the CL algorithm always keeps learning on-line and could adjust the control policy to adapt to eventual variations of the plant dynamics.

## VI. Conclusions

We have presented a new learning algorithm that takes advantage of a kind of regularity of the environment denoted as categorizability. The algorithm was applied to the problem of trajectory tracking control of a rotational joint under different model parameters.

The CL algorithm is able to make efficient generalization taking advantage of the categorizability of the environment reducing both the storage needs and the convergence time to make possible the learning of control policies whose performance can be compared with those of PID controls specifically tuned for each specific situation. No problem reformulation for the CL algorithm was needed to achieve a good performance in the different situations. Moreover, the CL algorithm does not require any previous knowledge of the plant and can learn a good control policy even in the presence of large plant parameters variations, while a PID system without an adequate tuning fails. It is also remarkable that no traditional control techniques are involved in the learning process or in the control performance of the CL algorithm.

We are now working on the application of our learning system based on the categorizability property to the control of more complex manipulators, considering high load inertias and variable perturbations.

## Acknowledgements

This work has been partially supported by the Ministerio de Ciencia y Tecnología and FEDER, under the project DPI2003-05193-C02-01 of the Plan Nacional de I+D+I.

## References

[1] E. Celaya and C. Torras. Visual Navigation Outdoors: the ARGOS Project. Proc. of the 7th. International Conf. on Intelligent Autonomous Systems, Marina del Rey, USA, March 2002, pp. 63-67.

[2] E. Celaya and J.L. Albarral. Implementation of a hierarchical walk controller for the LAURON III hexapod robot. 6th International Conference on Climbing and Walking Robots (CLAWAR 2003), Catania , Italy, September 2003, pp. 409-416.

[3] J. Randlov, A. G. Barto, and M. T. Rosenstein, "Combining Reinforcement Learning with a Local Control Algorithm". *International Conference on Machine Learning (ICML2000)*, Standford, United State, June 29-July 2, 2000.

[4] L. De S. Martins-Filho, J. Silvino, P. Presende, T. Assunçao, "Control of robotic leg joints – comparing PD and sliding mode approaches". *Proc. Of the Sixth International Conference on Climbing and Walking Robots (CLAWAR2003)*, Catania, Italy, September 2003, pp.147-153.

[5] M. Pérez Cisneros y R. Leal Ascencio "Reinforcement Learning Neurocontroller Applied to a 2-DOF Manipuator" Ref: ISIC01-040, *Proc. of the 2001 IEEE Conference on Control Applications -CCA- and the 2001 IEEE International Symposium on Intelligent Control -ISIC-*, Mexico City, September 5-7, 2001.

[6] E. Burdet, J. Luthiger, "Three learning architectures to improve robot control: A comparison", *Proc. MLnet Familiarization Workshop and Third European Workshop on Learning Robots*, Heraklion, Greece, April 28-29,1995.

[7] R. Sutton and A. Barto. Reinforcement Learning. An Introduction. MIT Press, 1998.

[8] R.E. Bellman. Dynamic Programming, Princeton University Press. Princeton.1957.

[9] J.M. Porta and E. Celaya. Learning in Categorizable Environments, *Proc. of the Sixth International Conference on the Simulation of Adaptive Behavior (SAB2000)*, Paris, September 2000, pp.343-352.

[10] A. Agostini and E. Celaya. Learning in Complex Environment with Feature Based Categorization. *Proc. of the Eigth Conference of Intelligent Autonomous Systems*, Amsterdam, March 2004, (in press).

[11] K. Ogata, *Modern Control Engineering*, 4th ed., Prentice Hall, New Jersey, United State, 2002.

[12] Maxon Motor, *High Precision Drives and Systems*, Maxon Interelectric AG, Switzerland.