Teaching a Robot to Perform Tasks with Voice Commands

Ana C. Tenorio-Gonzalez*, Eduardo F. Morales, and Luis Villaseñor-Pineda

National Institute of Astrophysics, Optics and Electronics,
Computer Science Department,
Luis Enrique Erro #1, 72840 Tonantzintla, México
{catanace17,emorales,villasen}@inaoep.mx
http://www.inaoep.mx

Abstract. The full deployment of service robots in daily activities will require the robot to adapt to the needs of non-expert users, particularly, to learn how to perform new tasks from "natural" interactions. Reinforcement learning has been widely used in robotics, however, traditional algorithms require long training times, and may have problems with continuous spaces. Programming by demonstration has been used to instruct a robot, but is limited by the quality of the trace provided by the user. In this paper, we introduce a novel approach that can handle continuous spaces, can produce continuous actions and incorporates the user's intervention to quickly learn optimal policies of tasks defined by the user. It is shown how the continuous actions produce smooth trajectories and how the user's intervention allows the robot to learn significantly faster optimal policies. The proposed approach is tested in a simulated robot with very promising results.

Keywords: reinforcement learning, voice command, service robotics, continuous spaces

1 Introduction

Service robots are rapidly expanding and soon will become part of everyday life. Their complete acceptance, however, will come when the robots are able to learn new tasks from natural interactions with their users. Many approaches have been developed to allow a robot to learn a new task, but particularly emphasis has recently been given to programming by demonstration and to reinforcement learning.

In reinforcement learning an agent uses its own experience to learn how to do a task, receiving rewards and punishments for good and bad actions that are scored depending on their contribution to reach a goal. However, traditional algorithms of reinforcement learning have very long convergence times and may have problems with continuous state and action spaces. Using a tabular representation is only applicable to simple domains and some value function approximations, as

* Master's thesis scholarship No. 214262 and research project No. 84162 from Conacyt

neural networks or Gaussian processes are computationally expensive that make them infeasible for on-line learning of tasks.

In programming by demonstration, a teacher shows the robot how to perform a task, however, it requires specialized hardware (e.g., gloves, special marks, etc.) and a controlled environment (i.e., controlled illumination conditions, special camera setting, etc.). They also tend to learn the task exactly as the user showed it, however human demonstrations tend to be noisy and suboptimal, and current systems are unable to improve it.

In this paper we introduce an algorithm to instruct a robot on-line using speech and reinforcement learning with continuous states and actions. A simple representation based on kernels is used to deal with continuous spaces and a novel combination of discrete actions is used to produce continuous actions. We use speech to teach a robot a task without the need of any special equipment, providing initial traces and on-line feedback to the robot while it is learning a task. With these elements the convergence times of the reinforcement learning algorithm are significantly reduced.

The rest of the paper is organized as follows. Section 2 gives an overview of related work using reinforcement learning with continuous states and actions, and with feedback provided in different ways. Section 3 has a description of the speech recognition process and vocabulary used. Section 4 describes our proposed algorithm of reinforcement learning with continuous states and actions adding feedback by the user. The experiments and results using a simulated autonomous mobile robot are presented in Section 5. Finally, conclusions and some ideas for future research work are presented in Section 6.

2 Related Work

There have been many approaches suggested in the literature to deal with continuous state and action spaces in reinforcement learning. For instance, some of them use artificial neural networks [4, 1, 3, 2, 17, 19], Gaussian process [14, 13], tile coding, regressions [16, 11, 5], trees, kernels, receptive fields [18] and approximations based on Gaussian functions [7], among others. The most widely used approaches are based on artificial neural networks and Gaussian processes, however, they have a high computational cost and require a substantial training data which makes them inadequate for a fast on-line learning approach as proposed in this paper. Methods based on regressions, tile coding and receptive fields have similarities with tabular representations and they can work on-line and have potential to be used in an incremental way. Our proposed method is similar to receptive fields however we incorporate a novel approach for continuous actions and a simple state representation that can work on-line with low computational cost. Furthermore, we incorporate voice commands as additional reinforcements into the learning process.

Several authors have considered the use of feedback in reinforcement learning [20, 9, 6, 8, 10]. But, these approaches use computer peripherals such as joystick, keyboard, mouse, and camera among others, to provide feedback to the system.

The reported work assumes discrete spaces and the feedback is provided only at certain stages of the learning process. Unlike, we used natural language and allow the user to provide feedback at any stage of the learning process.

There are a few research works that have used spoken feedback [15, 21]. In [15] a system of learning by demonstration is introduced for an autonomous mobile robot (CMAssist). The robot learns how to do a task watching a human and receiving verbal instruction. Behaviors of the robot and tasks are represented with a directed acyclic graph. Verbal instructions are structured in similar way as control structures of a programming language. Additionally, information of the environment is given to the robot with a map and the learning process is supervised. By contrast, the method that we propose does not require any knowledge about the environment and uses a more natural verbal interaction.

In [21] the authors present an Actor-Critic model based on an algorithm called IHDR (Hierarchical Discriminant Regression). They use Q-learning with a parameter of 'amnesia'. A teacher set the robot's arm in different positions, each one is named with a particular word (command) which the teacher says. Then, the complete sequence of positions is named too. So, a position or sequence of them can be invoked with its identifier. A teacher uses a bumper to mark the positions and to provide some reinforcements, so it is necessary an interaction with hardware. In this paper, we propose a method without any hardware of the robot and with a more natural interaction approach.

3 Spoken Feedback

The use of speech gives flexibility and more natural interaction to the user during training and can be used to qualify or modify in different ways the behavior of the robot. The human voice interaction can be provided at any time and with different intensities and intentions, and not require any special equipment or a deep knowledge about robot learning. We chose Spanish as our interaction language and created two corpora of vocabulary (with around 250 words). Our initial corpus was composed of isolated words which we later expanded in the second corpus to deal with short phrases. The vocabulary of the speech recognizer is composed by qualifiers and commands of actions. Some words used in the first corpus and short phrases used in the second are presented in Table 1.

The transcriptions of speech are analyzed by an interpreter designed to look for words of interest considering the stage of the learning process. After the interpreter has identified the words of interest, depending on their meaning, it changes the behavior of the learning algorithm. If the words are qualifiers, the reward function is modified adding a value corresponding to the meaning of the word. If an action command is understood, then the respective action is sent to the robot also modifying the learning process. This is explained in the next section.

Table 1. Examples of vocabulary with a rough translation into English

WORDS	SHORT PHRASES
Avanzar (forward)	Hacia adelante (move forward)
Regresar (backward)	Hacia atrás (move backwards)
Izquierda (left)	Gira a la izquierda (turn to your left)
Derecha (right)	Ve a tu derecha (go to your right)
Fin, Final (end)	Para ahí (stop there)
Bien (good)	Sigue así (keep like this)
Mal (bad)	Por ahí no (not that way)
Excelente (excellent)	Muy bien (very good)
Terrible (terrible)	Así no (not like that)
Objetivo (goal)	Hasta ahí (until there)

4 Reinforcement Learning Algorithm

In Reinforcement learning (RL) an agent explores the environment to reach a goal. The agent receives rewards or punishments for its actions, and it tries to maximize the total accumulated expected reward. Reinforcement learning can be modeled using Markov Decision Processes (MDPs). A MDP consists of:

1. S a set of states s,
2. A a set of actions a,
3. $T : SXA \rightarrow S$ a probability transition function of going to state s' given that action a was taken at state s,
4. $R : SXAXS \rightarrow R$ a reward function that evaluates how good is to do an action a from a state s according to the state reached s'.

Algorithms of reinforcement learning try to approximate a value function that estimates how good is for an agent be in a state and also, how good is to do a particular action in that state. The value function based on states an actions is defined by:

$$Q^{\pi}(s,a) = E_{\pi}\left\{\Sigma_{k=0}^{\infty}\gamma^{k} r_{t+k+1} | s_t = s, a_t = a\right\} \ . \tag{1}$$

The best policy between all policies of behavior is pursued. This optimal policy, denoted by $\pi*$ defines an optimal value function defined by:

$$Q^{*}(s,a) = max^{\pi} Q^{\pi}(s,a), \forall s \in S, \forall a \in A \ . \tag{2}$$

There are several reinforcement learning algorithms that have been proposed to find optimal policies and value functions. One of the most popular algorithms of reinforcement learning is SARSA (State-Action-Reward-State-Action), an algorithm that solves MDP's based on temporal differences (differences between successive predictions). In this paper, we proposed a modified SARSA algorithm with eligibility traces.

The algorithm that we proposed consists of three main components: (i) initial traces of how to perform a new task are provided by a teacher using voice commands, (ii) a reinforcement learning algorithm with exploration and continuous state and action spaces is used to learn an optimal policy, and (iii) the user provides voice feedback to the system during the learning process to improve the current policy and help the reinforcement learning algorithm to converge faster.

4.1 Initial Traces

In order to teach the robot a new task, the user provides spoken instructions to the robot to complete the task. The robot executes the interpreted actions until it receives a finishing instruction. The user can provide several of these traces by voice as a first stage of the learning process. But, these traces may have some erroneous actions due to misinterpretations of the commands by the speech recognition system or mistakes made by the user during the instruction phase. So, we use reinforcement learning and additional feedback from the user to quickly converge to an adequate policy during the learning process.

4.2 States, Actions and Value Functions

States are incrementally created as the robot traverses the environment. The robot receives data from its sensors, creates a new state representation and compares it with its stored states. If its current state is similar enough (as explained below) to a known state, the robot continues with its trace, otherwise, it stores the new state. There are different ways in which we can represent and compare states. In this work we used two representations and similarity functions, one using the mean and standard deviation of sensor readings as state representation and a Gaussian function to evaluate distance, and another one using directly the information from the sensors and a Pearson's correlation coefficient. With the Gaussian function, we have:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)/(2\sigma^2)} \ . \tag{3}$$

where, μ is the mean array of the sensor's data representing a state, σ is its standard deviation and x is the new array of the sensor's data.

And using the Pearson's correlation coefficient we have:

$$r = \frac{N\Sigma xy - \Sigma x \Sigma y}{(\sqrt{N\Sigma x^2 - (\Sigma x)^2})(\sqrt{N\Sigma y^2 - (\Sigma y)^2})} \ . \tag{4}$$

where N is the size of sample (number of x, y pairs), x, y are arrays of sensor's data, the first one has the values of a stored state and the second has a new set of data obtained from the sensors in the current robot's position.

Each new state is associated with a set of basic discrete actions (forward, backward, left, and right) and each state-action pair is associated with a Q-value.

During the training phase, the robot follows the actions provided by the user and incrementally builds the representation of states. During the learning phase, if the robot enters a new state then it chooses a random action, otherwise, it selects an action based on the combination of the actions with greater Q-values. The combined action is proportional to the Q-values of the selected discrete actions. For example, if in a particular state the actions with greater Q-values are 'forward' and 'right', if the 'right' action has the largest Q-value, then the robot will go more to the right than forward producing an intermediate action a_r with a value:

if $Q(s, a_1) < Q(s, a_2)$

$$va_r\left(s\right) = \left(Q\left(s, a_1\right) / Q\left(s, a_2\right)\right) * va_1 + \left(1 - Q\left(s, a_1\right) / Q\left(s, a_2\right)\right) * va_2 \ . \tag{5}$$

where va is the value of an action according to the domain of the task.

If the actions with the largest Q-values are opposite (i.e., right and left or forward and backward), one of them is randomly chosen. If all the actions have similar Q-values, also one is randomly chosen.

In our implementation we use a modified SARSA learning algorithm with eligibility traces. So, the updating of the Q-values is performed considering the combined action using the following modified Sarsa(λ) update rule:

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha \delta_t e_t(s, a) \ . \tag{6}$$

for all s, a, where

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \ . \tag{7}$$

and if $s = s_t$ and $a = a_{1_t}$ or $a = a_{2_t}$,

$$e_t(s, a) = \gamma \lambda e_{t-1}(s, a) + 1 \ . \tag{8}$$

otherwise,

$$e_t(s, a) = \gamma \lambda e_{t-1}(s, a) \ . \tag{9}$$

where s is a state, a is an action, a_{1_t} and a_{2_t} are the two actions with the largest Q-values, α is a positive step-size parameter, γ is a discount-rate parameter and λ is a decay parameter.

Since a continuous action is a combination of two basic actions (a_1, a_2), two Q-values are updated at each step instead of only one. This combined action produces continuous actions and keeps a simple updating procedure.

Reward Function During the learning process, two kinds of rewards are given. In addition to the traditional rewards given in reinforcement learning, additional rewards can be given by the user at any time. The reinforcement learning is still updating its Q-values as described before, but now the reward function is given by:

$$R = R_{RL} + R_U \ . \tag{10}$$

where R_{RL} is the traditional reward function used in reinforcement learning and R_U is an additional reward given when the user specifies an action or qualifies the behavior of the robot. This is similar to reward shaping, however, instead of being predefined in advance and given all the time during the learning process, the rewards are given by the user occasionally, can change their values over time and also can be wrong.

At the end, the initial policy constructed by the system during training phase is tested by the robot and improved with the feedback provided by the user.

5 Experiments

The experiments were focused in navigation tasks using a robot Pionner 2 with a frontal laser and a rear sonar. The robot and the environment were simulated on the Player/Stage platform running on Linux Debian. The speech recognizer used was Sphinx3 with the acoustic models level T22 based on corpus DIMEx100 [12]. The language models were created over a specific vocabulary for the tasks.

In order to test the capabilities of the algorithm, we teach the robot to do navigation tasks with different levels of complexity (increasing distance, number of doorways and corridors), shown in Figure 1. In the first task the robot only had to go inside a room from a point in a hall. The second task involved leaving a room, going through a corridor with four doorways and then entering a room. In the third task, the robot had to leave a room, go through a hall with three doorways, go through one doorway, go through another hall with two doorways and go through one final doorway. In the last task, the robot learned to leave a room, go through a corridor with five doorways, go through one doorway and then enter a final room (see Figure 1). Contrary to traditional navigation tasks, the robot in this case has no knowledge about the environment, using the laser as main sensor to identify states (using its 180 distance measures) and its sonars to detect collisions (in the rear part). A microphone was used by the user to give the feedback and the initial traces by voice during the training of the robot. The teacher could provide feedback at any moment during learning process (qualifiers and commands of actions) and could also provide wrong feedback by mistake.

For these experiments, the internal rewards were $+100$ when the robot reached the goal, -100 when the robot is close to walls and -1 for other state-action pairs. Two types of external feedback were given by the user: qualified commands which were translated into additional rewards and action commands which were translated into robot actions. In the experiments we used the following values for the qualification commands: $+100$ for reaching the goal (*objetivo*), $+50$ for "excellent" (*excelente*), $+10$ for "good" (*bien*), -50 for "terrible" (*terrible*), and -10 for "bad" (*mal*). Similar rewards were associated to other words and to short phrases. The qualifiers given depend on observable states produced by actions done. And, if the user gives a command of an action, the action is performed by the robot. Otherwise, the robot follows its normal reinforcement learning algorithm. An action command given by the user can also be accompanied by a qualifier that also modifies the reward function.

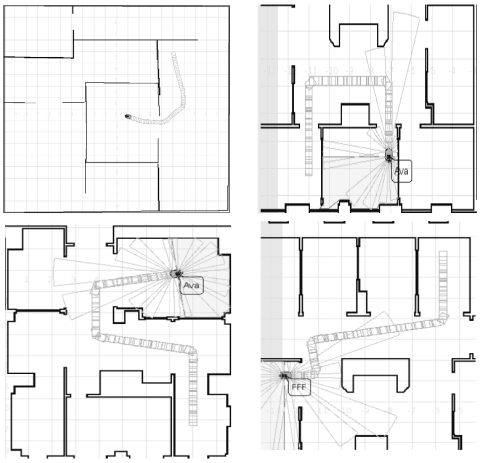

Fig. 1. Tasks 1-4 (left to right, up to down) taught to the robot.

In our initial experiments we used a Gaussian function but it created a larger number of states (200-500) when compared to the Pearson's coefficient (20-140) and it also produced more errors in the identification of states. In the following section we present results only with the Pearson's coefficient.

5.1 Results

We did four types of experiments to show the different aspects of our proposed algorithm and the advantages of the approach:

1. Reinforcement Learning (RL) with continuous states and discrete actions
2. RL with contiguous states and actions
3. RL with continuous states and actions with oral feedback from the user
4. RL with continuous states and actions, initial traces given by the user with voice commands and with oral feedback from the user

We first compared the behavior of policies learned using discrete actions with policies learned using our approach for generating continuous actions. Figure 2 shows a comparison in two traces where it is clear that the continuous action policy produces smoother and shorter paths with a reduction of about 1 meter in these experiments. Larger reductions are obtained in longer paths.

Each experiment was repeated three times and the averages are shown in the figures. Figure 3 shows the duration and number of episodes required to learn each task in all different experiments without any knowledge of the environment. As can be seen, the best results are obtained using traces and on-line feedback by the user with an important reduction in time to obtain an acceptable policy. The policies reached with the different learning strategies are very similar except for the traditional RL strategy with discrete state and action spaces.

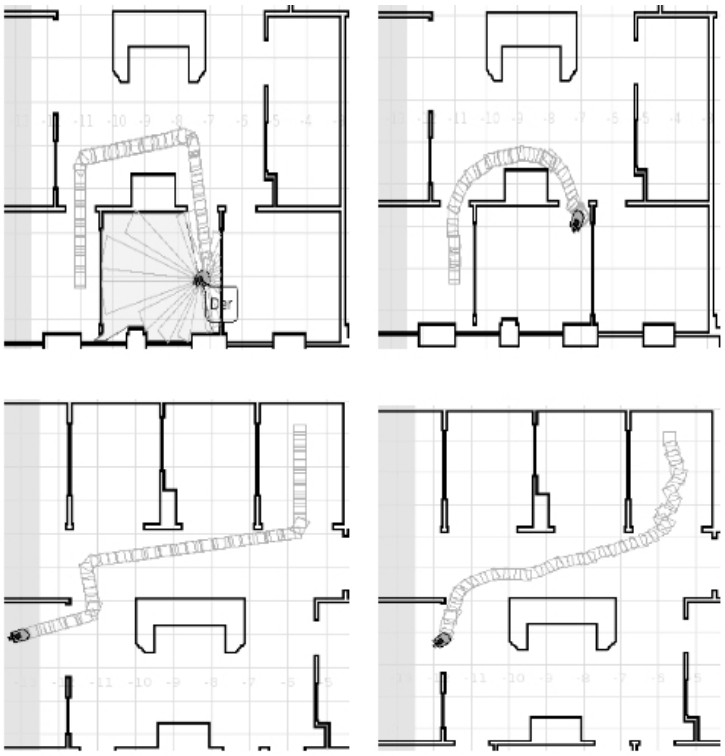

Fig. 2. Comparison of traces obtained with the algorithm using discrete actions (figures to the left) and continuous actions (figures to the right. Exp. 1). For the first task, the difference of distance between the discrete and continuous action policies was about 1m, for the second task, the difference was about 0.9m.

As can be seen in Table 2 there is a substantial reduction in the number of required episodes and total training times with the proposed approach. In these experiments the user needs to spend an average of 17 minutes to train a robot to perform a new task.

The speech recognition system is not perfect and it is common for the robot to understand a different command and acts accordingly. Even with such errors our approach is able to converge faster to a reasonable policy in few iterations.

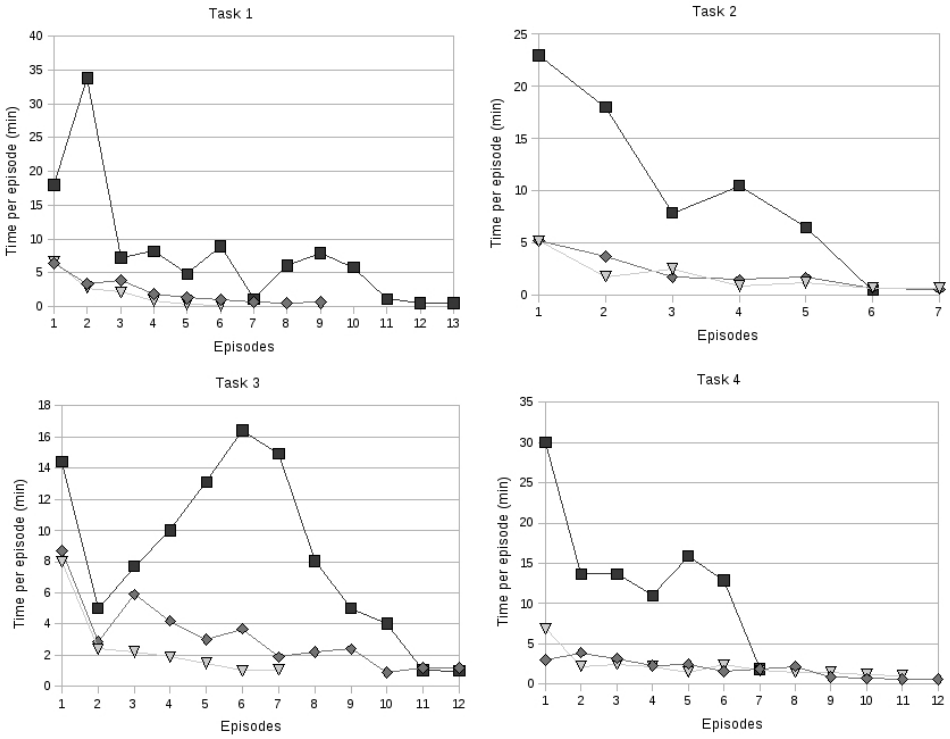

Fig. 3. Results of experiments. Times of learning, using different variations of the algorithm: without traces and without feedback (square), with feedback (diamond), with traces and feedback (triangle). The x-axis has number of episodes, the y-axis has duration (minutes) of each episode.

6 Conclusions and Future Works

This paper introduces a novel approach to teach a robot how to perform a new task using reinforcement learning and feedback provided by a teacher. The algorithm uses an incremental creation of states able to deal with continuous states

Table 2. Total number of episodes (table to the left) and times (table to the right) of the different experiments per task (RL = reinforcement learning with continuous actions, RL + F = RL with feedback from the user, and RL + T + F = RL + F with the initial traces provided by the user).

	RL	RL + F	RL + T + F
T1	13	9	6
T2	6	7	7
T3	12	12	7
T4	7	12	11
Avg	9.5	10	7.75

	RL	RL + F	RL + T + F
T1	103.93	19.59	12.85
T2	66.4	15.1	13
T3	100.65	38.2	18.09
T4	99.1	23.43	24.61
Avg	92.54	24.08	17.13

and a simple, yet effective, combination of discrete actions to produce continuous actions. The algorithm works on-line and was used successfully on a simulated autonomous mobile robot for different navigation tasks. Our experiments show that by including the user into the learning process a substantial reduction in the convergence times for obtaining an adequate policy can be obtained, even with errors performed by the user or by the speech recognition system. We think that teaching a robot with voice commands and providing oral feedback during learning is a natural way to instruct robots and opens a new possibility for non-experts to train robots how to perform new tasks.

As future work, we are planning to include a mechanism to quickly forget mistakes during the learning process. We would also like to increase the vocabulary used by the user and test our approach with external users. We would also like to test our algorithm with other robots and with different tasks.

References

1. Coulom R.: Feedforward Neural Networks in Reinforcement Learning Applied to High-dimensional Motor Control. In *13th International Conference on Algorithmic Learning Theory*. LNCS. Vol. 2533. Springer. (2002)
2. Chohra A., Madani K., Kanzari D.: Reinforcement Q-Learning and Neural Networks to Acquire Negotiation Behaviors. In *Studies in Computational Intelligence*. Springer Series. The Special Session New Challenges in Applied Intelligence Technologies (2008)
3. Dongli W., Yang G., Pei Y.: Applying Neural Network to Reinforcement Learning in Continuous Space. In *International Symposium on Neural Networks*. (2005)
4. Gromann A. , Poli R.: Continual robot learning with constructive neural networks. In *Sixth European Workshop*, No. 1545. LNAI. (1997)
5. Guenter F., Hersch M., Calinon S., Billard Aude: Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics*, Vol. 21, No. 13, pp. 152-154 (2007)
6. Iida F., Tabata M., Hara F.: Generating Personality Character in a Face Robot through Interaction with Human. In *7th IEEE International Workshop on Robot and Human Communication*, pp. 481-486 (1998)

7. Kimura H., Yamashita T., Kobayashi S.: Reinforcement Learning of Walking Behavior for a Four-Legged Robot. In *40th IEEE Conference on Decisions and Control*. (2001)
8. Lockerd T. A., Breazeal C.: Reinforcement Learning with Human Teachers: Evidence of Feedback and Guidance with Implications for Learning Performance. In *21st National Conference on Artificial Intelligence*. Vol.1 (2006)
9. Lockerd T. A., Hoffman G., Breazeal C.: Real-Time Interactive Reinforcement Learning for Robots. In *AAAI Workshop on Human Comprehensible Machine Learning*. (2005)
10. Lockerd T. A., Hoffman G., Breazeal C.: Reinforcement Learning with Human Teachers: Understanding How People Want to Teach Robots. In *15th IEEE International Symposium on Robot and Human Interactive Communication*, pp. 352–357 (2006)
11. Melo F. S., Lopes M.: Fitted natural actor-critic: A new algorithm for continuous state-action MDPs. In *European conference on Machine Learning and Knowledge Discovery in Databases*. Part II. LNAI, Vol. 5212 (2008)
12. Pineda L.: Corpus DIMEx100 (Level T22). DIME Project. Computer Sciences Department. ISBN:970-32-3395-3. IIMAS, UNAM
13. Rasmussen C. E.: Gaussian Processes in Machine Learning. In *Advanced Lectures on Machine Learning*, ML Summer Schools 2003. LNCS, Springer (2004)
14. Rottmann, A. Plagemann, C. Hilgers, P. Burgard, W.: Autonomous Blimp Control using Model-free Reinforcement Learning in a Continuous State and Action Space. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. (2007)
15. Rybski P. E., Yoon K., Stolarz J., Veloso Manuela M.: Interactive Robot Task Training through Dialog and Demonstration. In *ACM/IEEE international conference on Human Robot Interaction*, pp. 255-262 (2007)
16. Smart W. D., Pack Kaelbling L.: Effective Reinforcement Learning for Mobile Robots. In *IEEE International Conference on Robotics and Automation* (2002)
17. Tadashi T., Koichi S., Yasuhiro W.: Robot Task Learning based on Reinforcement Learning in Virtual Space. *Neural Information Processing, Letters and Reviews*. pp. 165-174. (2007)
18. Tamosiunaite, M., Asfour, T. and Wrgtter, F.: Learning to reach by reinforcement learning using a receptive field based function approximation approach with continuous actions. *Biological Cybernetics*. Vol. 100, Issue 3. (2009)
19. Van Hasselt H., Wiering M.: Using Continuous Action Spaces to Solve Discrete Problems. In *International Joint Conference on Neural Networks*. (2009)
20. Wang Y., Huber M., Papudesi V.N., Cook D.J.: User-Guided Reinforcement Learning of Robot Assistive Tasks for an Intelligent Environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems.*(2003)
21. Yilu Z., Juyang W.: Action Chaining by a Developmental Robot with a Value System. In *2nd International Conference on Development and Learning*. (2002)