J. Intelligent Learning Systems & Applications, **2010, 2: 80-85**
doi:10.4236/jilsa.2010.22011 Published Online May 2010 (http://www.SciRP.org/journal/jilsa)

An Experience Based Learning Controller

Debadutt Goswami, Ping Jiang*

Department of Computing, University of Bradford, Bradford, UK.
Email: p.jiang@bradford.ac.uk

Received November 10th, 2009; revised April 10th, 2010; accepted April 20th, 2010.

ABSTRACT

The autonomous mobile robots must be flexible to learn the new complex control behaviours in order to adapt effectively to a dynamic and varying environment. The proposed approach of this paper is to create a controller that learns the complex behaviours incorporating the learning from demonstration to reduce the search space and to improve the demonstrated task geometry by trial and corrections. The task faced by the robot has uncertainty that must be learned. Simulation results indicate that after the handful of trials, robot has learned the right policies and avoided the obstacles and reached the goal.

Keywords: *Mobile Robots, Learning from Demonstration, Neural Network Control, Reinforcement Learning*

1. Introduction

Most of the autonomous mobile robots or the human controlled mobile robots working in the industrial, domestic and entertainment environment lack the flexibility to learn and perform the new complex tasks. In this paper we consider the problem of motion control of an autonomous mobile robot using the control field generated by a neural controller. The proposed system enables the autonomous mobile robot to learn, modify and adapt its skills efficiently in order to react adequately in complex and unstructured environment. To incorporate the precision and flexibility in robots, we incorporated three very common natural approaches that, human uses to learn and execute any task, 1) Learning from demonstration 2) Generalization of the learned task 3) Reinforcement of the task by trial and error [1,2]. The designing of a motion controller depends on the kinematics constraint of a mobile robot and the complexity and structure of a task, and is very difficult to design whereas the proposed controller can easily and quickly learn the complex controls with some simple demonstrations and few trials. The major advantage of our technique is that any mobile robot can be taught to move in an unknown environment using the generated control field as the general control law. The control law is independent of both physical and virtual sensors. The use of domain knowledge has a great significance in reducing the learning time and the use of trial and error learning method improves the learning continuously [3]. The learning from demonstration has reduced the robot programming and made it very simple to use [4,5]. The reinforcement learning has already been used in many navigational and reactive problems [6,7,8]. The ability of reinforcement learning to interact with the environment and the domain knowledge from demonstrations has made it easier to tackle uncertainty. Due to the simplicity to operate and flexibility to learn uncertainty, the proposed controller can be operated by the non-professionals who are not skilled enough to control and program the sophisticated mobile robots in the complex industrial environment [3,9].

The paper is presented in following manner. Section II presents the previous works related to the field. Section III presents the controller architecture and the proposed algorithm. Section IV presents the simulation results. Section V presents conclusions and discusses the work further.

2. Related Works

We have incorporated two different research areas: Learning from demonstration and Reinforcement Learning, which bears some relation to a number of previous works. The importance of human robot interaction and the need of modern artificial intelligence are described in [10] that focus on creating new possibilities for the flexible and interacting robot from the engineering point of view and from the humanistic point of view. The paper incorporates these ideas and focuses on flexible learning controller of a robot which can learn and adapt the changes in the environment very easily. The proposed methodology differs from [3,11,12] and [5] in two fundamental aspects. Firstly our approach learns from demonstration, which acts as initial policy to perform the desired task and builds a model of partially observable environment. Secondly, the methodology is offline and online computations are done on the basis of generated control field and the previous experience. The proposed controller initializes the weights of the controller from the demonstrated points of the task whereas in [11] the learner is initially empty. The proposed controller utilizes *Q*-values to avoid uncertainty where as [3] and [11] used fuzzy behaviour as a reflex to act into a new situation. The aim is to generate an abstract description of the task reflecting user's intention and modelling the problem solution offline. In this context we also discuss the issues of evaluation, tuning, suitable generalization and execution of the elementary skills [9,13-15]. Instead of using radial basis as a function approximation [3,12], we used inverse distance interpolation. It works like local approximators. We also used learning through feedback, which improves the learning and generalization continuously.

3. Controller Architecture

The proposed learning controller can be represented as recurrent neural network architecture. The controller has basically 3 layers, input layer, hidden layer and output layer and a reinforcement controller which is connected to the output layer and the hidden layer. $S \in R^N$ and $A \in R^N$ are the sensor input and output. The weight is updated when the sensor and the controller try to learn an unknown model from sensors to actuators, *i.e.*

$$A = f(S) \tag{1}$$

Input Layer—Input layer consists of several neural input nodes that represent some state in the state space *i.e.* *S*. The sensor input is propagated through the hidden layer from the input layer to the output layer. Each neural node in the input layer is connected to all the neural nodes in the hidden layer.

Hidden Layer—Hidden layer is composed of several computational neurons. The output of the hidden neurons is the distance between the input layer neuron and the weight similar to a prototype fuzzy law. The weight W_i in the hidden layer neuron represents the centre *i.e.* the demonstrated data points. Each neural node of the input layer is connected to all the neural nodes of the hidden layer. The output of the hidden layer codes the distance of the input neurons from the hidden layer neurons.

Output layer—The output layer neuron computes the inverse weighted sum of the output v_i of the hidden layer neurons *i.e.*

$$A = \sum_{i=1}^{N} W_{oi} \times v_i \tag{2}$$

The output layer neuron uses inverse distance interpolation to update the weights W_{oi} of the output layer.

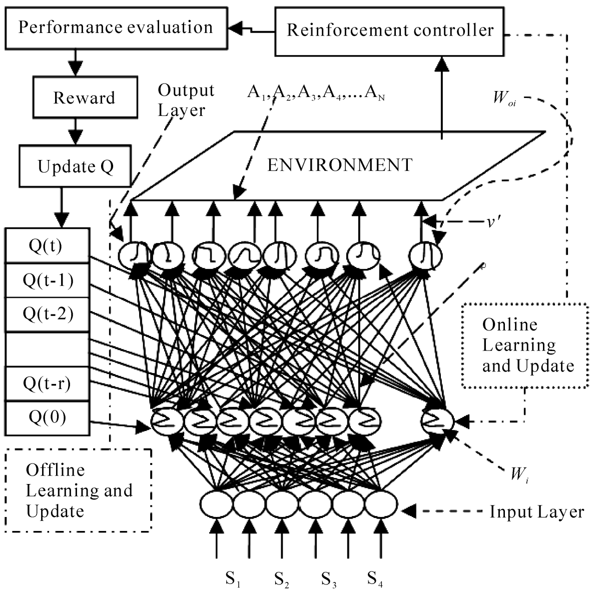

Figure 1. Neural network control architecture

The advantage of using this technique is that, the estimated data point has the influence of all the neighbouring data points depending on the distance from each prototype rule. This is the simplest generalized technique and mathematically expressed as

$$v' = \left(\sum_{i=1}^{N(v')} \frac{1}{d^p} \times v_i \right) \Bigg/ \left(\sum_{i=1}^{N(v')} \frac{1}{d^p} \right) \tag{3}$$

where, v' is the estimated control action in term of a sensor state S'; v_i is a prototyped action of the hidden layer neuron at state S; d^p is the distance between S and S'; p is the power; $N(v')$ is the number of data points in the effective window of v'. Generally the distance can be calculated by simple Euclidean distance formula. The above generalization technique only acts as a function approximation that reproduces the generalized control field based on the distance from the existing prototype rule. But to tackle uncertainty and to produce an improved control field, the robot needs to incorporate a weight that represents the significance of the previous learning history and incorporates continuous learning experiences. The idea is to improve the learning by trial and error. So *Q*-learning [16] is incorporated, which produces *Q*-matrix based on the scalar reward *R*. The *Q*-value determines the new policy to achieve the goal. Therefore in the neural network architecture, the output layer is connected to the reinforcement controller which in turn is connected to the hidden layer. Depending on the achievement of the desired goal, the reinforcement reward *R* is given to the robot by the reinforcement controller. The reward *R* updates the *Q*-matrix and the new *Q*-value is used to generate a new control field. The reinforcement controller follows the online exploration and offline learning policy [17]. So the *Q*-value is updated in following two cases:

Case 1: Online exploration—During online exploration the robot uses *Q*-matrix to avoid uncertainty like unknown obstacles. Therefore, whenever the robot finds uncertainty in its next course of action, the control switches from the generalized control field to *Q*-matrix and selects an action with other maximum *Q*-value from its effective window that corresponds to a policy with next higher probability to reach the goal and avoids the uncertainty. This earns an online reward *R* and updates the *Q*-value online in that time step. The change in the action in that time step is used as a prototype rule *i.e.* center in the hidden layer.

Case 2: Offline learning policy—In the end of the trial during offline learning policy, the reinforcement reward *R* is given offline to the robot. The reward *R* updates the corresponding *Q*-values of the trajectory based on the final payoff. After updating *Q*-value, the change is applied to the neurons in the hidden layer of the network architecture. So another factor that influences the output of the hidden layer is the *Q*-value.

The *Q*-value is then used as a measure to calculate the relevance of the data points in terms of usefulness of that particular task geometry in achieving the goal. The change in the *Q*-matrix influences all the neighbouring data points in the control field. This is called the generalization. After that the control field is transferred back to the robot. In every trial *Q*-matrix is updated and reflects the continuous learning. The steps are repeated until we get the desired control field. This technique of learning is known as online exploration and offline policy learning. The reward scheme for online exploration and offline policy learning is as follows:

$$R = \begin{cases} +1 \text{ If the goal is achieved or obstacle is aovided} \\ 0 \text{ If no obstacle or no goal} \\ -1 \text{ If failed to achieve goal or unable to aovid obstacle} \end{cases} \tag{4}$$

The reward function can represented as:

$$R = R_{uncertainty} + R_{goal} \tag{5}$$

In online exploration R_{goal} is set to zero and $R_{uncertainty}$ is awarded online to avoid obstacles on its path whereas in offline learning policy R_{goal} is awarded offline and $R_{uncertainty}$ is set to zero.

The *Q* update rule can be described in two steps.

Step 1*:* Give Reward *R* using (4) and (5) and update the weight Q_w using the weight update Equation.

$$\begin{aligned} & Q_w(s,a) \\ & = Q_w(s,a) + \alpha \times \left[R(s,a) + \lambda \times Q_w(s',a') - Q_w(s,a) \right] \end{aligned} \tag{6}$$

Here $Q_w(s,a)$ is the weight matrix to be updated; (*s*, *a*) is the current state and action; $R(s,a)$ is the reward matrix; $Q_w(s',a')$ is the future state with the highest Q_w value; α is the learning rate; λ is the discount factor in the range [0, 1]. The weight update Equation (6) is inherited from the traditional *Q*-learning algorithm [16].

Step 2*:* Estimate the data points v' based on the Q_w calculated in step 1. The new estimate updates the previously generated control field and generates a new generalized control field. To implement our idea we modify (3) by introducing Q_w.

$$v' = \left(\sum_{i=1}^{N(v')} \frac{Q_w}{d^p} \times v_i \right) \Bigg/ \left(\sum_{i=1}^{N(v')} \frac{Q_w}{d^p} \right) \tag{7}$$

Here $\frac{Q_w}{d^p}$ is the weight. A small Q_w value represents the small contribution of the point in accomplishing the goal where as a large Q_w value signifies large contribution of the point in accomplishing the goal. Q_w is inversely related to the distance, which signifies that the Q_w will have higher value when the estimated data point is closer to the good trajectories. The trajectories with lower Q_w value have less contribution in the overall generalization.

In the proposed context behaviour is improved based on scalar rewards from a critic. It does not require a person to program the desired actions in different situations. However several difficulties stand in the way. Firstly a robot using the basic reinforcement-learning framework might require an extremely long time to converge to the right track and to acquire the right action. In our framework learning from demonstration is used that reduces the search space to achieve an adequate control and it does not need any programming of the desired behaviour. The demonstration provides the domain knowledge, that gives hint to perform the desired task and reduces knowledge base significantly. Through trial and error the knowledge base can be improved further [18,19]. Secondly reinforcement learning is basically applied on the task where the state of the environment is completely observable, but in the real world tasks most of the knowledge is incomplete and inaccurate. The online exploration and offline learning policy explores the unknown environment with the help of generalized control field and tackles the uncertainty by selecting the appropriate control action which signifies maximum experience *i.e.* maximum *Q*-value in the effective window in achieving the desired goal. The trajectory generated by it changes the control field and tackles the uncertainty in the environment.

Ideally for a real robot any learning algorithm should be feasible and efficient to perform the complex computations, keeping in mind the robots internal configuration under consideration. A robot has limited memory and limited computational ability. Almost all the cases of practical interest consist of far more state observation pairs than could possibly entered into the memory. The robot is even typically unable to perform enough computation per time step to fully use it. Especially in the real world experiences, which consist of large number of action and observation pairs and at each step the robot needs to memorize and learn the action-observation pairs. Thus a good approach is to compute and store a limited number of data online which are relevant and learning the policy offline. In the proposed algorithm the old data can be used and new experiences can be collected, which are somewhat characteristic for the task. The best thing is the experience replay and the assessment of the performance of the previous experiences. Another important step is the generalization of the learned policy. The policy learned in this way keeps the learning continuous. This kind of learning is generally called as learning from experience. For a robot controller, it is also important to reduce the real time computation. The robot takes action on the generated control field only. Hence learning is made continuous and the computational overhead is also reduced. On the other hand the exploration is done online. The robot decides between exploration and exploitation based on the number of times an action has been chosen. The robot generally chooses action with the highest state action value with high probability, which is known as exploitation. If the robot oscillates in a particular region then exploitation is stopped and control switches to exploration to avoid oscillation and finds a new path. During exploration the selection of next state is made random. This approach has a major advantage of not being influenced by the limited knowledge only *i.e.* to avoid local optima. During exploration the robot can take the control actions which were never taken before and hence exploring more of the unobserved environment. The balance between exploration and exploitation is necessary, because exploitation is helping the robot to avoid uncertainty and to reach the goal where as exploration is avoiding all the oscillation and unnecessary iterations. So this methodology has a very positive approach to learn any new skill quickly. The robot can be restrained easily to adapt to environment changes and trained and learned improving the performance all the time.

The proposed algorithm can be summarized as below:

1) Build the control field by demonstrating the task, *i.e.* initializing the weights in the hidden layer with the demonstrated data points and imitating the relevant knowledge required to achieve the task.

2) Based on the demonstrated control field, using inverse distance weight interpolation, generalize the field using (7).

3) Transfer the generalized control field to the robot and do some trials to achieve the goal. Do some exploration also. If all the trials are successful in achieving the goal then the demonstrated control field is perfect and does not need any changes. If in trials, at some point the robot hits or senses any obstacle then give reward $R_{uncertainty}$ to the point using (4) and (5) and move towards the point that has next highest Q-value, without any obstacle. The next higher Q-value represents the next most experienced and favorable point heading towards the goal. Then update the Q-values using (6). This will generate a new path to reach the goal. After the goal has been reached, generalize the control field offline using (7). The control field will have influence from the points updated. This will reproduce an updated control field.

4) After reaching the goal, assign final reward R_{goal} to the robot using (4) and (5). This reward represents the successful completion of the desired task avoiding all the obstacles and is made offline. Update all the Q-values of the updated control trajectory generated in step 3 using (6).

5) Generalize the control field with the new updates using (7). The updated field represents the new control field. Repeat from Step 3 to 5 until the robot learns the desired task.

4. Simulation Results

In simulation we build a discrete workspace with a dimension of 20 × 20. The workspace contains a goal and obstacles, represented by the 'G' and the rectangles. Robot has a sensor of 3 × 3 neighbourhoods which means it can sense obstacles around its 8 neighbouring cells. The arrows represent the direction of each point in the field. The simulation is done based on the proposed algorithm in section 3. The robot was initially demonstrated to reach the goal. Based on the demonstration initial control field was built (**Figure 2**). The simulation considered two different tasks with different uncertain complexities with same initial control field.

Task 1: The robot had to avoid an obstacle and reach the goal.

Task 2: The robot had to avoid obstacles and pass through the door and reach the goal.

In both the cases robot completed the tasks in few trials avoiding all the uncertainties. The final control fields for task 1 and task 2 are shown in **Figures 3** and **4**.

The learning history of the controller for both the cases is described in **Figures 5-7**. The broken line curve and the solid line curve represents first and second task.

In **Figure 6** it can be observed that number of hidden neurons increases with the number of trials. Increase in hidden layer neurons indicates the increase in number

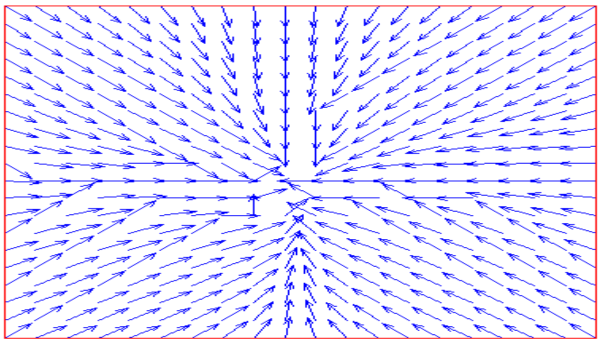
Figure 2. Demonstrated control field

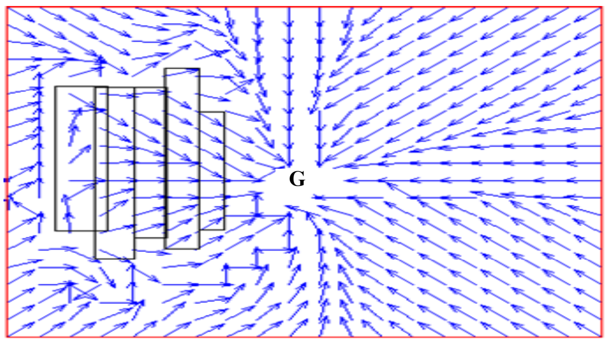

Figure 3. Final control field of task 1

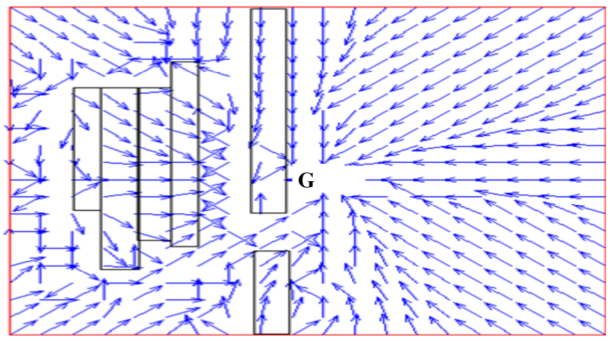

Figure 4. Final control field of task 2

centers or prototype rules in the network. It clearly indicates the learning in each trial. **Figure 7** displays the plotting of the total number reinforcements used in each trial. The total reinforcement can be defined as the sum of immediate reinforcement the learner receives till the robot reaches the goal and the offline reinforcement the learner receives after reaching the goal. It is clear in **Figure 7** that after trial 3 the controller did not use any reinforcements to complete the tasks. The controller needed only three trials 1-3 to learn a new control law to avoid uncertainty in both tasks. To see the reliability of the control field we chose different starting points. The robot was successful in completing the tasks from all the points.

Another observation is that the number of reinforcement received (**Figure 7**) and the number of steps taken

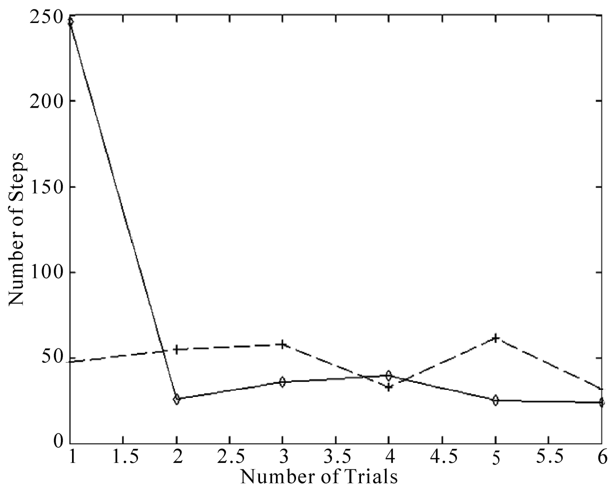

Figure 5. Number of steps taken to reach goal in each trial

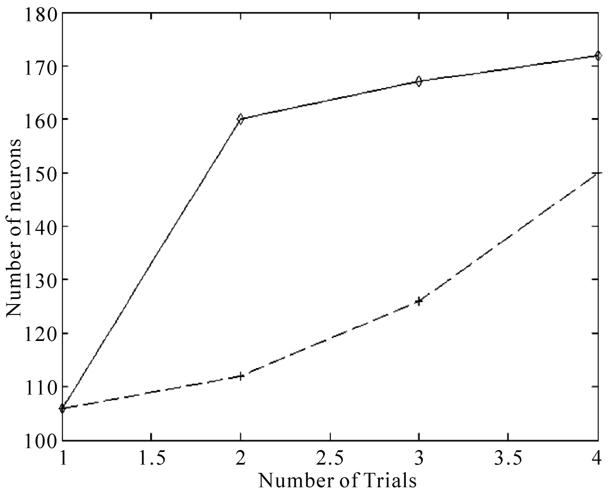

Figure 6. Number of hidden layer neurons used in each trial

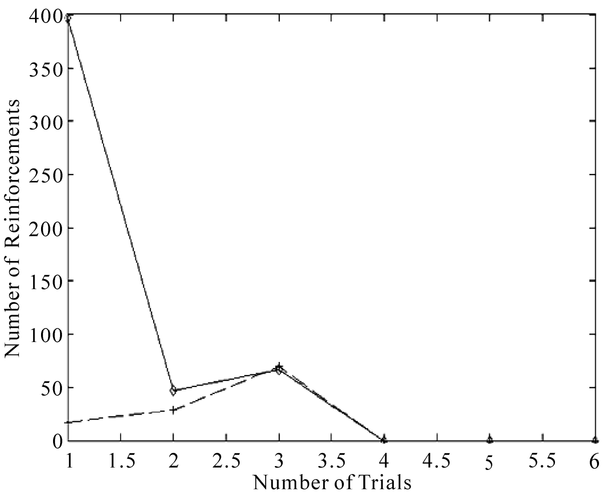

Figure 7. Number of reinforcements given in each trial

(**Figure 5**) in trial 1 in task 2 (solid line curve) is very high. It indicates that, in task 2 the maximum exploration was done in trial 1 where as in task 1 exploration was increasing with trials. Therefore with increase in exploration reinforcement is also increasing. Furthermore, it was during first three trials the robot attained reinforcements (**Figure 7**) and a sharp growth in network (**Figure 6**).

5. Conclusions

This paper presented a learning strategy to generate a control field for a mobile robot in an unknown and uncertain environment, which integrates learning, generalization, exploration and offline computation into a unified architecture. After the learning, a robot can approach the goal by following the control field. The important lessons learned from the implementation included 1) imitation of very accurate and exact action sequence is not necessary [15]; 2) a prior knowledge is required to plan a model of the task to support rapid learning; 3) generalization is improved by improving the learning policies; 4) simple method like inverse distance is adequate to generalize the task; 5) offline learning is an important method for real time applications to avoid the large online computations; 6) online exploration is required to explore other possibilities to perform a task and to improve the quality of learning; 7) balance between exploration and exploitation improves the learning policies, which reduces the learning time significantly; 8) the robot can learn from few demonstrations but it effects the learning speed. The major disadvantage of this method is the use of position information which is not always accurate in real robots due to inaccurate sensor information due to rotational or translational error caused by the slippage between robot's wheel and the floor.

REFERENCES

[1] M. N. Nicolescus and M. J. Mataric, "Natural Methods for Robot Task Learning: Instructive Demonstrations, Generalization and Practice," *In Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Melbourne, Australia, July 14-18, 2003, pp. 241-248.

[2] M. N. Nicolescus and M. J. Mataric, "Learning and Interacting in Human—Robot Domains," *IEEE Transactions on systems*, *Man and cybernetics—Part A*: *Systems and Humans*, Vol. 31, No. 5, 2001, pp. 419-430.

[3] G. Hailu,"Symbolic Structures in Numeric Reinforcement for Learning Optimum Robot Trajectory," *Robotics and Automation Systems*, Vol. 37, No. 1, 2001, pp. 53-68.

[4] D. C. Bentivegna, C. G. Atkeson and G. Cheng, "Learning Tasks from Observation and Practice," *Robotics and Automation Systems*, Vol. 47, No. 2-3, 2004, pp. 163-169.

[5] G. Hailu and G. Sommer, "Learning by Biasing," *IEEE International Conference on Robotics and Automation*, Leuvem, Belgium, 1998, pp. 16-21.

[6] J. R. Millan, "Reinforcement Learning of Goal Directed Obstacle Avoiding Reaction Strategies in an Autonomous Mobile Robot," *Robotics and Autonomous Systems*, Vol. 15, No. 4, 1995, pp. 275-299.

[7] A. Johannet and I. Sarda, "Goal-Directed Behaviours by Reinforcement Learning," *Neurocomputing*, Vol. 28, No. 1-3, 1990, pp. 107-125.

[8] P. M. Bartier and C. P. Keller, "Multivariate Interpolation to Incorporate Thematic Surface Data Using Inverse Distance Weighting (IDW)," *Computers & Geosciences*, Vol. 22, No. 7, 1996, pp. 795-799.

[9] H. Friedrich, M. Kaiser and R. Dillmann, "What Can Robots Learn from Humans," *Annual Reviews in Control*, Vol. 20, 1996, pp. 167-172.

[10] S. Thrun, "An Approach to Learning Mobile Robot Navigation," *Robotics and Automation Systems*, Vol. 15, 1995, pp. 301-319.

[11] M. Kasper, G. Fricke, K. Steuernagel and E. von Puttkamer, "A Behaviour Based Learning Mobile Robot Architecture for Learning from Demonstration," *Robotics and Automation Systems*, Vol. 34, No. 2-3, 2001, pp. 153-164.

[12] W. Ilg and K. Berns, "A Learning Architecture Based on Reinforcement Learning for Adaptive Control of the Walking Machine LAURON," *Robotics and Automation Systems*, Vol. 15, No. 4, 1995, pp. 321-334.

[13] H. Friedrich, M. Kaiser and R. Dillmann, "PBD-The Key to Service Robot Programming," AAAI Technical Report SS-96-02, American Association for Artificial Intelligence, America, 1996.

[14] H. Friedrich, M. Kaiser and R. Dillmann, "Obtaining Good Performance from a Bad Teacher," *In Programming by Demonstration vs. Learning from Examples Workshop at ML'*95, Tahoe, 1995.

[15] R. Dillmann, M. Kaiser and A. Ude, "Acquisition of Elementary Robot Skills from Human Demonstration," *In International Symposium on Intelligent Robotics Systems*, Pisa, Italy, 1995, pp. 185-192.

[16] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," The MIT Press Cambridge, Massachusetts, 1998.

[17] B. Bakker, V. Zhumatiy, G. Gruener and J. Schmidhuber, "A Robot that Reinforcement-Learns to Identify and Memorize Important Previous Observations," *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, USA, October 27-31, 2003, pp. 430-435.

[18] A. Billard, Y. Epars, S. Calinon, S. Schaal and G. Cheng, "Discovering Optimal Imitation Strategies," *Robotics and Automation Systems*, Vol. 47, No. 2-3, 2004, pp. 69-77.

[19] S. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach," Second Edition, Prentice Hall, New Jersey, USA, 2002.