Proceedings of the 2006 IEEE/RSJ
International Conference on Intelligent Robots and Systems
October 9 - 15, 2006, Beijing, China

Towards Direct Policy Search Reinforcement Learning for Robot Control

Andres El-Fakdi, Marc Carreras and Pere Ridao
Institute of Informatics and Applications
University of Girona
Edifici Politecnica 4, Campus Montilivi
17071, Girona (Spain)
Email: aelfakdi@eia.udg.es

***Abstract*— This paper proposes a high-level Reinforcement Learning (RL) control system for solving the action selection problem of an autonomous robot. Although the dominant approach, when using RL, has been to apply *value* function based algorithms, the system here detailed is characterized by the use of Direct Policy Search methods. Rather than approximating a value function, these methodologies approximate a policy using an independent function approximator with its own parameters, trying to maximize the future expected reward. The policy based algorithm presented in this paper is used for learning the internal state/action mapping of a behavior. In this preliminary work, we demonstrate its feasibility with simulated experiments using the underwater robot GARBI in a target reaching task.**

I. Introduction

Reinforcement Learning is a widely used methodology in robot learning [1]. In RL, an agent tries to maximize a scalar evaluation obtained as a result of its interaction with the environment. The goal of a RL system is to find an optimal policy to map the state of the environment to an action which in turn will maximize the accumulated future rewards. The agent interacts with a new, undiscovered environment selecting actions computed as the best for each state, receiving a numerical reward for every decision. The rewards are used to teach the agent and in the end the robot learns which action it must take at each state, achieving an optimal or sub-optimal policy (state-action mapping).

The dominant approach over the last decade has been to apply reinforcement learning using the *value function* approach. As a result, many RL based control systems have been applied to robotics. In [2], an instance-based learning algorithm was applied to a real robot in a corridor-following task. For the same task, in [3] a hierarchical memory-based RL method was proposed, obtaining good results as well. In [4] an underwater robot that learnt different behaviors using a modified Q-learning algorithm was presented. Although value function methodologies have worked well in many applications, they have several limitations. Function approximator methods in "value-only" RL algorithms may present converge problems, if the state-space is not completely observable, small changes in the the value function can cause big changes in the policy [5].

Over the past few years, studies have shown that approximating a policy can be easier than working with value functions, and better results can be obtained [6] [7]. Informally, it is intuitively simpler to determine *how to act* instead of *value of acting* [8]. So, rather than approximating a value function, new methodologies approximate a policy using an independent function approximator with its own parameters, trying to maximize the future expected reward. Furthermore, scientists have developed different kinds of policy search algorithms obtaining good results [5] [9] [10]. Also in [10] a study about how gradient methods can be used to search in the space of stochastic policies is presented.

Policy gradient algorithms can be used to represent the policy. For example, an ANN whose weights are the policy parameters. The state would be the input of the network and as output we would have a distribution probability function for action selection. In (1) we can see that if θ represents the vector of the policy parameters and ρ the performance of the policy (e.g., reward received), then the policy parameters are updated approximately proportional to the gradient [6]:

$$\Delta\theta \approx \alpha \frac{\delta\rho}{\delta\theta} \tag{1}$$

where α is a positive step size. In comparison with the value function approach, small changes in θ can cause only small changes in the policy.

The advantages of policy gradient methods against value-function based methods are various. The main advantage is that using a function approximator to represent the policy directly solves the generalization problem. Besides, a problem for which the policy is easier to represent should be solved using policy algorithms [7]. Furthermore, learning systems should be designed to explicitly account for the resulting violations of the Markov property. Studies have shown that stochastic policy-only methods can obtain better results when working in partially observable Markov decision processes (POMDP) than those obtained with deterministic value-function methods [11]. In [7] a comparison between a policy-only algorithm [12] and a value Q-learning method [13] is presented; both algorithms use a simple neural network as function approximator. A 13-state Markovian decision process is simulated for which the Q-learning oscillates between the optimal and a suboptimal policy while the policy-only method converges to the optimal

1-4244-0259-X/06/$20.00 ©2006 IEEE

policy. On the other hand, as disadvantage, policy gradient estimators used in these algorithms may have large variance, so these methods learn much more slower than RL algorithms using a value function [14] [15] [6] and they can converge to local optima of the expected reward [16].

The first example of an algorithm optimizing the averaged reward obtained, for stochastic policies working with gradient direction estimates, was Williams's REINFORCE algorithm [17]. This algorithm learns much more slower than other RL algorithms which work with a value function and, maybe for this reason, has received little attention. However, the ideas and mathematical concepts presented in REINFORCE were a basic platform for later algorithms.

A few years later, in [18], Williams's algorithm was extended to the infinite horizon setting. Kimura's method is, as REINFORCE, based on stochastic gradient ascent (SGA). The authors compared its algorithm with Jaakola's method [19] and Watkin's Q-learning algorithm [13] in a robot control problem achieving good results.

The Baxter and Bartlett approach [20] is the one selected in this paper to carry out the experiments. Its method calculates a parameterized policy that converges to an optimal by computing approximations of the gradient of the averaged reward from a single path of a controlled POMDP. The convergence of the method is proven with probability 1, and one of the most attractive features is that it can be implemented on-line. Baxter and Bartlett's approach is based on the fact that, given a state s, it searches for a policy that minimizes the expected reward. Moreover, in [21] and [22] an algorithm similar to Baxter and Bartlett's approach was described and its convergence demonstrated. The algorithm is only suitable for finite MDP and can be implemented to work on-line.

Close to the root of these theoretical variants of policy search methods, only a few but promising practical applications of these algorithms have appeared. Chronologically, this paper emphasizes the work presented in [23], where an autonomous helicopter learns to fly using an off-line model-based policy search method. Also important is the work presented in [24] where a simple "biologically motivated" policy gradient method is used to teach a robot in a weightlifting task. More recent is the work done in [25] where a simplified policy gradient algorithm is implemented to optimize the gait of Sony's AIBO quadrupedal robot. Finally, in [26] and [27], a biped robot is trained to walk by means of a "hybrid" RL algorithm that combines policy search with value function methods.

In this paper we apply Baxter and Bartlett's algorithm to a particular robotic task in which a neural network acts as the policy function. The task consists on reaching a target which is detected by a forward looking camera. These experiments have been designed for the Autonomous Underwater Vehicle (AUV) GARBI. In this paper the learning has been fulfilled with the hydrodynamic model of GARBI and the model of a video camera. The structure of the paper is as follows. In Section II the algorithm and the learning procedure are detailed. Section III describes all the elements considered in the target reaching camera (the simulated world, the robot model and the controller). The experimentation procedure and the results obtained are included in Section IV and finally, conclusions and the future work to be done after this work are included in Section V.

II. Learning Procedure

The objective of this work is to transfer an accurate policy, learned in a simulated environment, to a real robot and test the behavior of the policy in real conditions. So, the learning process can be divided into two main phases. First, the learning task will be performed in simulation using the model of the environment. Once the learning process is considered to be finished, the policy will be transferred to GARBI AUV in order to test it in the real world. In this paper we present only the results of the first phase.

A. The Algorithm

Baxter and Bartlett's algorithm procedure is summarized in Algorithm 1. The algorithm works as follows: having initialized the parameters vector θ_0, the initial state i_0 and the eligibility trace $z_0 = 0$, the learning procedure will be iterated T times. At every iteration, the parameters' eligibility z_t will be updated according to the policy gradient approximation. The discount factor $\beta \in [0, 1)$ increases or decreases the agent's memory of past actions. The immediate reward received $r(i_{t+1})$, and the learning rate α allows us to finally compute the new vector of parameters θ_{t+1}. The current policy is directly modified by the new parameters becoming a new policy to be followed by the next iteration, getting closer to a final policy that represents a correct solution of the problem.

Algorithm 1: Baxter and Bartlett's On-Line POMDP (OLPOMDP) algorithm

1. Initialize:
   $T > 0$
   Initial parameter values $\theta_0 \in R^K$
   Initial state i_0
2. Set $z_0 = 0$ ($z_0 \in R^K$)
3. $for\ t = 0$ to T do:
   (a) Observe state y_t
   (b) Generate control action u_t according to current policy $\mu(\theta, y_t)$
   (c) Observe the reward obtained $r(i_{t+1})$
   (d) Set $z_{t+1} = \beta z_t + \frac{\nabla \mu_{u_t}(\theta, y_t)}{\mu_{u_t}(\theta, y_t)}$
   (e) Set $\theta_{t+1} = \theta_t + \alpha_t r(i_{t+1}) z_{t+1}$
4. $end\ for$

As aforementioned, the algorithm is designed to work on-line. The function approximator adopted to define our policy is an artificial neural network (ANN) (see Figure 1).

Next lines will relate closely to the update weight process done by the algorithm. Once the ANN is initialized at random, the network will be given an observation of the state and, as a result, a stochastic control action is computed. Then the learner will be driven to another state and will receive a reward associated with this new state. The first step in the parameter update procedure is to compute the ratio:

$$\frac{\nabla \mu_{u_t}(\theta, y_t)}{\mu_{u_t}(\theta, y_t)} \tag{2}$$

for every weight of the network. In artificial neural networks like the one used in the algorithm, the expression defined in step 3.d of Algorithm 1 can be rewritten as:

$$z_{t+1} = \beta z_t + \delta_t y_t \tag{3}$$

At any step time t, the term z_t represents the estimated gradient of the reinforcement sum with respect to the network's layer weights. In addition, δ_t refers to the local gradient associated with a single neuron of the ANN and is multiplied by the input to the neuron y_t. In order to compute these gradients, we evaluate the soft-max distribution for each possible future state exponentiating the real-valued ANN outputs $\{o_1, ..., o_n\}$, being n the number of neurons of the output layer [8].

After applying the soft-max function, the outputs of the neural network give a weighting $\xi_j \in (0, 1)$ to each of the possible control actions. Finally, the probability of the ith control action is then given by:

$$Pr_i = \frac{\exp(o_i)}{\sum_{a=1}^{n} \exp(o_a)} \tag{4}$$

where n is the number of neurons at the output layer. Actions have been labeled with the associated control action and chosen at random from this probability distribution. Once we have computed the output distribution over all possible actions, the next step is to calculate the gradient for the action chosen by applying the chain rule. The whole expression is implemented similarly to error back propagation [28]. Before computing the gradient, the error on the neurons of the output layer must be calculated. This error is given by (5).

$$e_j = d_j - Pr_j \tag{5}$$

The desired output d_j will be equal to 1 if the action selected was o_j, and 0 otherwise (see Figure 2).

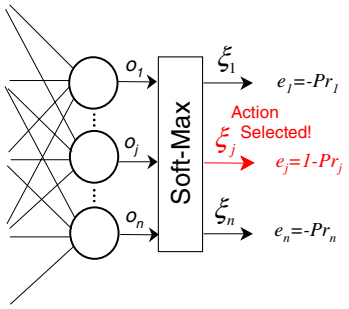

Fig. 2. Soft-Max error computation for every output.

With the soft-max output error calculation completed, the next phase consists of computing the gradient at the output of the ANN and back propagate it to the rest of the neurons of the hidden layers. For a local neuron j located in the output layer, we may express the local gradient as:

$$\delta_j^o = e_j \varphi_j'(o_j) \tag{6}$$

where e_j is the soft-max error at the output of neuron j, $\varphi_j'(o_j)$ corresponds to the derivative of the activation function associated with that neuron, and o_j is the function signal at the output for that neuron. So we do not back propagate the gradient of an error measure, but instead back propagate the soft-max gradient of this error. Therefore, for a neuron j located in a hidden layer, the local gradient is defined as follows:

$$\delta_j^h = \varphi_j'(o_j) \sum_k \delta_k w_{kj} \tag{7}$$

When computing the gradient of a hidden-layer neuron, the previously obtained gradient of the following layers must be back propagated. In (7) the term $\varphi_j'(o_j)$ represents the derivative of the activation function associated to that neuron, o_j is the function signal at the output for that neuron and finally the summation term includes the different gradients of the following neurons back propagated by multiplying each gradient to its corresponding weighting (see Figure 3).

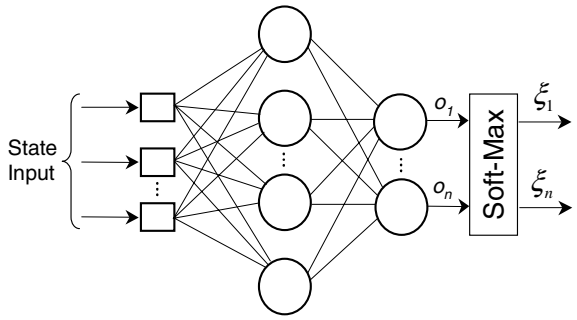

Fig. 1. Schema of the ANN architecture adopted.

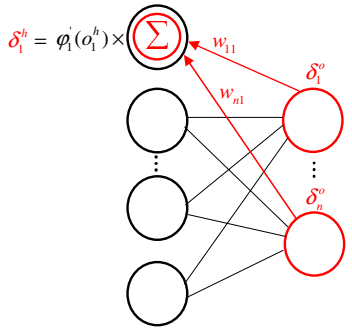

Fig. 3. Gradient computation for a hidden layer neuron.

Having all the local gradients of all the neurons calculated, the expression in (3) can be obtained. Finally, the old parameters are updated following expression 3.(e) of Algorithm 1:

$$\theta_{t+1} = \theta_t + \alpha r(i_{t+1}) z_{t+1} \tag{8}$$

The vector of parameters θ_t represents the network weights to be updated, $r(i_{t+1})$ is the reward given to the learner at every time step, z_{t+1}describes the estimated gradients mentioned before and, at last, we have α as the learning rate of the algorithm.

III. Case to Study: Target Reaching

This section is going to describe the different elements that take place in our problem: the problem of target reaching, the neural-network controller and the underwater robot GARBI.

A. Target Tracking

One of the sensory systems developed for the experimental set-up of GARBI is the *target detection and tracking* system. This vision-based application has the goal of detecting an artificial target by means of the forward looking camera. This camera provides a large underwater field of view (about 57^o in width by 43^o in height). This system was designed to provide the control architecture with a measurement of the position of an object to be tracked autonomously. Since the goal of these experiments are to test control and learning systems, a very simple target was used. The shape selected for the target was a sphere because it has the same shape from whatever angle it is viewed. The color of the target was red to contrast with the blue color of the water tank. These simplifications allowed us to use simple and fast computer vision algorithms to achieve real-time (12.5 Hz) performance. Figure 4 shows a diagram of the target being observed by GARBI.

The procedure of detecting and tracking the target is based on image segmentation. Using this simple approach, the relative position between the target and the robot is found. Also, the detection of the target in subsequent images is used to estimate its relative velocity. Once the target has been detected, its relative position with respect to the robot has to be expressed. The coordinate frame which has been used for the camera has the same orientation as the GARBI coordinate frame, but is located in the focal point of the camera. Therefore, the transformation between the two frames can be modeled as a pure translation.

The *X* coordinate of the target, represented by f_x, is related to the target size detected by the segmentation algorithm. A normalized value between *0* and *1* is linearly assigned to the range comprised between a maximum and minimum target size respectively. Similarly, the *Y* coordinate of the target is related to the horizontal position of the target in the image. However, in this case, the value represented by the f_y variable do not measure a distance, but an angle from the center of the image to the target around the *Z* axis. In this case, the angle is normalized from *-1* to *1* as it can be seen in the Figure 4.

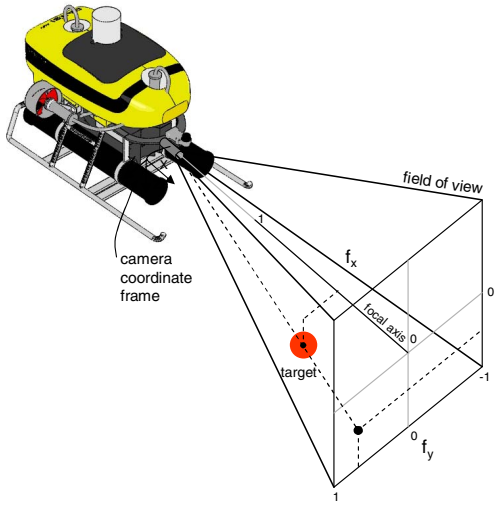

Fig. 4. Coordinates of the target in respect with GARBI.

In order to properly reach the target, the measure of its relative position is not enough. An estimation of its relative velocity is also necessary. To calculate this velocity, the f_x and f_y variables are differentiated from the sequence of images. In particular, a first order Savitzky-Golay [29] filter, with a first order derivative included, is applied to these signals.

B. The controller

A one-hidden-layer neural-network with 4 input nodes, 3 hidden nodes and 4 output nodes was used to generate a stochastic policy. As can be seen in Figure 5 the inputs to the network correspond to the normalized X and Y relative positions of the target, f_x and f_y, and the estimations of its relative velocity $\frac{df_x}{dt}$ and $\frac{df_y}{dt}$. Each hidden and output layer has the usual additional bias term. The best results have been obtained using the hyperbolic tangent function as the activation function for the neurons of the hidden layer, while the output layer nodes are linear. The four output neurons represent the possible four control actions, as can be seen in Figure 6 every action is a combined movement in surge (X movement) and yaw (rotation in Z axis). Note that in this preliminary work a very small action set has been considered. As explained in Section II-A, the outputs have been exponentiated and normalized to produce a probability distribution. In order to guarantee exploration, control actions are selected at random from this distribution.

C. GARBI AUV description

The GARBI platform was conceived as an AUV for exploration in waters up to 100 meters in depth. With a weight of 170 Kg, GARBI has a complete sensor suite including an imaging sonar, a DVL, a compass, a pressure gauge, a temperature sensor, a DGPS unit and a color camera. Hardware is enclosed into two cylindrical hulls designed to withstand pressures of 11 atmospheres. Two additional cylinders for batteries are placed at the bottom of the vehicle, ensuring the

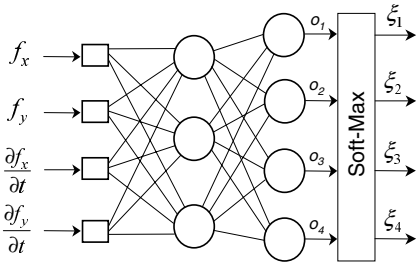

Fig. 5. The ANN used by the controller.

stability in both *pitch* and *roll* degrees of freedom. Its five thrusters will allow GARBI to be operated in the remaining degrees of freedom (*surge*, *sway*, *heave* and *yaw*) achieving maximum speeds of 3 knots (see Figure 7).

The mathematical model of GARBI was obtained using parameter identification methods [30]. The whole model has been uncoupled and reduced to emulate a robot with only two degrees of freedom (DOF), X movement and rotation respect Z axis. Also, the model of the camera has been used to simulate the vision system of GARBI AUV.

IV. Results

The controller was trained in an episodic task. According to the variables, f_x in the X DOF and f_y in the Yaw DOF, a reward value was given. Only three reward values were used: -20, -1 and 0. In order to maintain the target in front of the robot, the reward $r = 0$ is given when the position of the target is around $f_y = 0$ (between -0.2 and 0.2) and at a certain distance from the robot in X axis, around $f_x = 0.3$ (between 0.2 and 0.4). The reward value of -20 is given if the target is almost outside the image ($f_y < -0.9$ and $f_y > 0.9$) and the robot perceives a reward of -1 if it definitively loses the target. Robot an target positions are reset either every 50 seconds or when a "success"(reach reward 0) takes place, whatever comes first. The sample time was 0.1 seconds. The robot is always reset to position (0,0) meanwhile the target is reset to a random location inside the robot field of view.

Achieving a "success" or spending 50 seconds without reaching the target represents the end of an episode. The number of episodes to be done has been set to 100.000. For every episode, the total amount of reward perceived is calculated. Figure 8 represents the performance of the neural-network robot controller as a function of the number of episodes when trained using OLPOMDP. The episodes have been averaged over bins of 40 episodes. The experiment has been repeated in 100 independent runs, and the results here presented are a mean over these runs. The simulated experiments have been repeated for different values of the discount factor β.

Once the vehicle has learnt the task, it needs a few time steps to reach the goal. As it can be appreciated in Figure 8, the best performance is around -40. The best results are obtained when using a decay factor of $\beta = 0.95$. Different values of α have been tested without improving the results here presented. Figure 9 represents the behavior of a trained robot controller. Targets positions were deterministically selected to observe the robot moving to different locations.

V. Conclusions

A direct policy search algorithm for robot control based on Baxter and Bartlett's direct-gradient algorithm has been studied. The method has been applied to a simulated control system where the robot GARBI navigates a two-dimensional world learning to reach a target by means of a forward looking camera. The policy is represented by a neural network whose weights are the policy parameters. The objective of the agent was to compute a stochastic policy, which assigns a probability over each action.

The results of this preliminary work show a good performance of the algorithm. The convergence times are quite good. A future work will compare these results with a value function algorithm. A classical value method would have been affected by the generalization problem and, therefore, spent much more iterations to converge. Is is also important to note the reduced dimensions of the ANN used in the simulation.

The current work is focused on transferring the learned policy to the real robot and testing in real conditions. A future work will consist on performing the learning process on-line with the robot GARBI navigating in the real underwater environment.

Acknowledgment

This research was sponsored by the Spanish commission MCYT (CTM2004-04205/MAR). The authors would like to thank Mr. Douglas Alexander Aberdeen of the Australian National University, Mr. Russ Tedrake of the Massachusetts

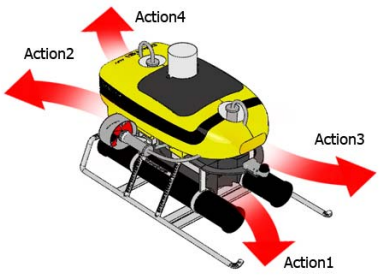

Fig. 6. GARBI schema. Control actions.

Fig. 7. GARBI AUV in experimental test.

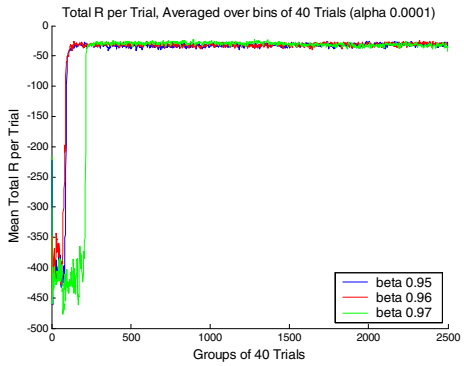

Fig. 8. Performance of the neural-network robot controller as a function of the number of episodes. Performance estimates were generated by simulating 100.000 episodes, and averaging them over bins of 40 episodes. Process repeated in 100 independent runs. The results are a mean of these runs. Fixed $\alpha = 0.0001$, and $\beta = 0.95, 0.96, 0.97$.

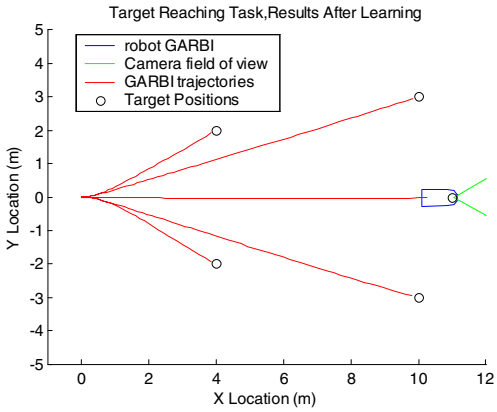

Fig. 9. Behavior of a trained robot controller, results of the target reaching task after the learning process was completed.

Institute of Technology and Mr. Jun Morimoto of the ATR Computational Neuroscience Laboratories for their help.

REFERENCES

[1] R. Sutton and A. Barto, *Reinforcement Learning, an introduction*. MIT Press, 1998.

[2] W. D. Smart and L. P. Kaelbling, "Practical reinforcement learning in continuous spaces," in *International Conference on Machine Learning*, 2000.

[3] N. Hernandez and S. Mahadevan, "Hierarchical memory-based reinforcement learning," in *Fifteenth International Conference on Neural Information Processing Systems*, Denver, USA, 2000.

[4] M. Carreras, P. Ridao, R. Garcia, and T. Nicosevici, "Vision-based localization of an underwater robot in a structured environment," in *IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, 2003.

[5] D. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[6] R. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in Neural Information Processing Systems*, vol. 12, pp. 1057–1063, 2000.

[7] C. Anderson, "Approximating a policy can be easier than approximating a value function," University of Colorado State," Computer Science Technical Report, 2000.

[8] D. A. Aberdeen, "Policy-gradient algorithms for partially observable markov decision processes," Ph.D. dissertation, Australian National University, April 2003.

[9] E. A. Hansen, "Solving POMDPs by searching in policy space," in *8th Conference on Uncertainty in Artificial Intelligence*, Madison, WI, 1998, pp. 211–219.

[10] N. Meuleau, K. E. Kim, L. P. Kaelbling, and A. R. Cassandra, "Solving POMDPs by searching the space of finite policies," in *15th Conference on Uncertainty in Artificial Intelligence*, M. Kaufmann, Ed., Computer science Dep., Brown University, July 1999, pp. 127–136.

[11] S. Singh, T. Jaakkola, and M. Jordan, "Learning without state-estimation in partially observable markovian decision processes," in *Proceedings of the Eleventh International Conference on Machine Learning*, New Jersey, USA, 1994.

[12] J. Baxter and P. Bartlett, "Direct gradient-based reinforcement learning," in *International Symposium on Circuits and Systems*, Geneva, Switzerland, May 2000.

[13] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[14] P. Marbach and J. N. Tsitsiklis, "Gradient-based optimization of Markov reward processes: Practical variants," Center for Communications Systems Research, University of Cambridge, Tech. Rep., March 2000.

[15] V. Konda and J. Tsitsiklis, "On actor-critic algorithms," *SIAM Journal on Control and Optimization*, vol. 42, number 4, pp. 1143–1166, 2003.

[16] N. Meuleau, L. Peshkin, and K. Kim, "Exploration in gradient based reinforcement learning," Massachusetts Institute of Technology, AI Memo 2001-003, Tech. Rep., April 2001.

[17] R. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.

[18] H. Kimura, K. Miyazaki, and S. Kobayashi, "Reinforcement learning in pomdps with function approximation," in *Fourteenth International Conference on Machine Learning (ICML'97)*, D. H. Fisher, Ed., 1997, pp. 152–160.

[19] T. Jaakkola, S. Singh, and M. Jordan, *Reinforcement Learning algorithms for partially observable Markov decision problems*. Morgan Kaufman, 1995, vol. 7, pp. 345–352.

[20] J. Baxter and P. Bartlett, "Direct gradient-based reinforcement learning: I. gradient estimation algorithms," Australian National University, Tech. Rep., 1999.

[21] P. Marbach and J. N. Tsitsiklis, "Simulation-based optimization of markov reward processes," Technical report LIDS-P-2411, Massachussets Institute of Technology, 1998.

[22] P. Marbach, "Simulation-based methods for markov decision processes," PhD Thesis, Laboratory for Information and Decision Systems, MIT, 1998.

[23] J. Bagnell and J. Schneider, "Autonomous helicopter control using reinforcement learning policy search methods," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Korea, 2001.

[24] M. Rosenstein and A. Barto, "Robot weightlifting by direct policy search," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2001.

[25] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[26] R. Tedrake, T. W. Zhang, and H. S. Seung, "Stochastic policy gradient reinforcement learning on a simple 3D biped," in *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS'04*, Sendai, Japan, September 28 - October 2 2004.

[27] T. Matsubara, J. Morimoto, J. Nakanishi, M. Sato, and K. Doya, "Learning sensory feedback to CPG with policy gradient for biped locomotion," in *Proceedings of the International Conference on Robotics and Automation ICRA*, Barcelona, Spain, April 2005.

[28] S. Haykin, *Neural Networks, a comprehensive foundation*, 2nd ed. Prentice Hall, 1999.

[29] A. Savitzky and M. Golay, *Analytical Chemistry*, 1964, vol. 36, pp. 1627–1639.

[30] P. Ridao, M. Carreras, and J. Batlle, "Model identification of a low-speed UUV," in *Control Applications in Marine Systems CAMS'01*, Scotland (UK), 2001.