Proceedings of the 2001 IEEE/RSJ
International Conference on Intelligent Robots and Systems
Maui, Hawaii, USA, Oct. 29 – Nov. 03, 2001

Hybrid Coordination of Reinforcement Learning-based Behaviors for AUV Control

M. Carreras, J. Batlle and P. Ridao
Institute of Informatics and Applications, University of Girona
Edifici Politècnica II, Campus Montilivi, 17071 Girona, Spain
{ marcc,jbatlle,pere }@eia.udg.es

Abstract

This paper proposes a Hybrid Coordination method for Behavior-based Control Architectures. The hybrid method takes in advantages of the robustness and modularity in competitive approaches as well as optimized trajectories in cooperative ones. This paper shows the feasibility of this hybrid method with a 3D-navigation application to an Autonomous Underwater Vehicle (AUV). The behaviors were learnt online by means of Reinforcement Learning. Continuous Q-learning implemented with a feed-forward neural network was applied. Realistic simulations were carried out. Results show the good performance of the hybrid method on behavior coordination as well as the convergence of the behaviors.

1 Introduction

Behavior-based Robotics [1] is a methodology for designing autonomous agents and robots. Since its appearance, in the middle of 1980s, a huge amount of robotic applications have used this methodology. An endless quantity of methods have been proposed to solve the common characteristics of a Behavior-based system: behavior expression, design, encoding and coordination. Behaviors are implemented as a control law using inputs and outputs. The basic structure consists of all behaviors taking inputs from the robot's sensors and sending outputs to the robot's actuators. Behavior coordination is the phase in which a coordinator module receives the responses of all the behaviors and generates a single output to be applied to the robots. If the output is the selection of a single behavior, the coordinator is classified as *competitive*. On the other hand, if the output is the superposition of several behavior responses, the coordinator is called *cooperative*.

According to the coordination system, some advantages and disadvantages appear in the control performance of an autonomous vehicle. After testing 4 well-known behavior-based architectures (Subsumption [2], Action Selection Dynamics [3], Schema-based approach [4] and Process Description Language [5]) in a simulated 3D-navigation mission with an AUV some conclusions were extracted [6,7]. Competitive methods (subsumption and action selection dynamics) show good robustness in the behavior selection and modularity when adding new behaviors. However, a bad trajectory is found when there is a continuous change of the dominant behavior. As far as cooperative methods are concerned, they have an optimal trajectory when parameters are properly tuned. However, they lack of robustness. A small change on the parameters can lead to control failures. In some circumstances, a set of behaviors can cancel the action of behaviors with a higher priority (i.e. obstacle avoidance behavior).

In this paper we propose a *hybrid* approach between competitive and cooperative coordination systems with the aim of taking advantage of both. Coordination is done through hierarchical hybrid nodes. These nodes act as cooperative or competitive coordinators depending on an activation level associated to each behavior. To test the feasibility of the hybrid coordination method, a behavior-based control architecture was designed and tested.

Making use of the high capability of Reinforcement Learning [8] for robot learning, behaviors were implemented using this technique. RL has been applied to various Behavior-based systems, most of them using Q_learning [9]. In some cases, the RL algorithm was used to adapt the coordination system [10, 11]. On the other hand, some researchers have used RL to learn the internal structure of a behavior, mapping the perceived states to robot actions [12, 13, 14]. The work presented by Mahadevan [12] demonstrated that the decomposition of the whole agent learning policy in a set of behaviors, as Behavior-based robotics proposes, simplified and increased the learning speed. The approach taken in this paper is a continuous implementation of the Q_learning algorithm. Generalization between states and actions was achieved by a feed-forward neural network which approximates the Q_function. Direct Q_learning [15] (backpropagation) was used to train the network.

As stated above, the field of application is underwater robotics. The work presented in this paper corresponds to a research project on Behavior-based Robotics and

0-7803-6612-3/01/$10.00©2001 IEEE

Reinforcement Learning experimenting with AUVs. In this paper, the theoretical assumptions are presented together with results based on realistic simulations of our AUV URIS. We use the term "realistic" due to the use of an accurate hydrodynamic model of the vehicle, the simulation of sensor noise and the use of onboard control software. Further work will be based on real experiments.

The structure of this paper is as follows. Section 2 describes the proposed hybrid coordination system. Section 3 introduces the continuous Q_learning algorithm used for behavior learning. In section 4, the application to test the hybrid coordination method is detailed. In section 5, simulation results are given. And finally, conclusions and future work are presented in section 6.

2 Hybrid Coordinator

Due to the disadvantages of competitive and cooperative methodologies and with the aim of making use of their advantages, a *hybrid coordination* method is proposed. In the proposed method, the coordination of the responses is done through a hybrid approach that keeps the robustness and modularity of competitive approaches as well as the good performance of the cooperative ones.

The coordinator is based on normalized behavior outputs. The outputs contain a three-dimensional vector "v_i" which represents the velocity proposed by the behavior. Associated with this vector is an activation level "a_i" which indicates how important it is for the behavior to take control of the robot. This value is between 0 and 1, see figure 1. This codification sharply defines the control action from the activation of the behavior.

The proposed coordination system is composed of a set of *hierarchical hybrid nodes*, see figure 2. The nodes have two inputs and generate a merged normalized control response. The nodes compose a hierarchical and cooperative coordination system. The idea is to use the good performance of cooperation when the predominant behavior is not completely active. The nodes have a dominant behavior which suppresses the responses of the non-dominant behavior when the first one is completely activated (a_i=1). However, when the dominant behavior is partially activated ($0<a_i<1$), the final response will be a combination of both inputs. Non-dominant behaviors can slightly modify the responses of dominant behaviors when they aren't completely activated. For example, if the dominant behavior is "obstacle avoidance" and the non-dominant is "go to point", when "obstacle avoidance" is only slightly activated (the obstacles are still far), a mixed response will be obtained. When non-decisive situations occur, cooperation between behaviors is allowed. Nevertheless, robustness is present when dealing with critical situations.

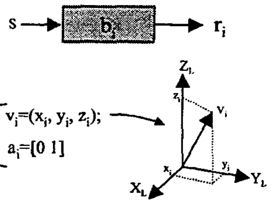

Figure 1. Normalized output of a behavior.

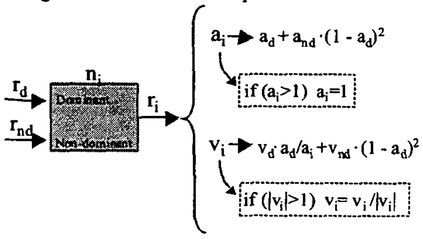
n_i
r_d
r_{nd}
Dominant
Non-dominant
r_i
$a_i \rightarrow a_d + a_{nd}\cdot(1 - a_d)^2$
if (a_i>1) a_i=1
$v_i \rightarrow v_d \cdot a_d/a_i + v_{nd}\cdot(1 - a_d)^2$
if ($|v_i|$>1) $v_i = v_i/|v_i|$

Figure 2. Hierarchical hybrid node.

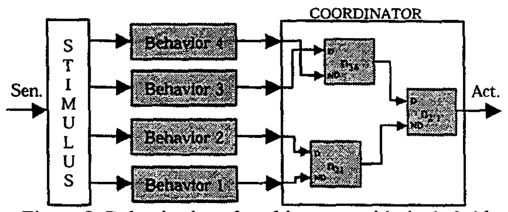

Figure 3. Behavior-based architecture with the hybrid coordination system.

The node "n_i" has the ability to generate a normalized response like the one generated by behaviors. The effect of the non-dominant behavior depends on the squared activation of the dominant to assure that in a critical situation between both, the dominant will always take control. Depending on the situation, the control response could be produced by all the behaviors or by only one. The hybrid nodes do not need a tuning phase. The coordination of a set of behaviors is defined hierarchically classifying each behavior depending on its priority. A disposition of the whole coordination system using hierarchical hybrid nodes can be seen in figure 3.

The coordination method can be classified as a *hybrid* approach because the response is the one generated by the dominant behavior affected by non-dominant behaviors according to the level of activation of the first. Although to the authors best knowledge there is not any sort of hybrid coordination system presented in the literature, this method offers good properties and can be successfully implemented in an autonomous robot. The proposed method has been implemented in simulation [7] showing its good path performance, robustness and modularity controlling an AUV in a 3D-navigation mission.

3 Reinforcement Learning-based Behaviors

When programming a Behavior-based system, there are some unknown parameters which cannot be identified without experimentation. To solve this difficulty, many robotic systems have included learning techniques. Adaptation is also needed in order to be able to perform in different and changing environments. Reinforcement Learning (RL) [8] is a class of learning suitable for robotics when online learning without information about the environment is required. In RL an agent tries to maximize a scalar evaluation (reward or punishment) of its interaction with the environment. The evaluation is generated by the *critic* using an utility function. A RL system tries to map the states of the environment to actions (policy) in order to obtain the maximum reward. In our case, the state is the sensor information perceived by the robot and the action is the behavior output (the velocity set-points). RL does not use any knowledge database as in most forms of machine learning. Most of the theories are based on Finite Markov Decision Processes (FMDPs).

The approach taken in this paper was a continuous implementation of the Q_learning algorithm [9]. Q_learning is a temporal difference [8] algorithm, which means that the transition probabilities between the states of the FMDPs are not required, and therefore, the dynamics of the environment does not have to be known. Temporal difference methods are also suitable to learn in an incremental way, required in online robot learning. An important characteristic of Q_learning is that is an off-policy algorithm. The optimal action values are leant independently of the policy being followed. This is very important in our behavior-based architecture because all the behaviors can be learnt even if they are not controlling the vehicle.

The original Q_learning algorithm is based on FMDPs. It uses the states perceived (s), the actions taken (a) and the reinforcements received (r) to update the values of a table, denoted as $Q(s,a)$. If state/action pairs are continually visited, the Q values converge to a *greedy policy*, in which the maximum Q value for a given state points to the optimal action. Figure 4 shows the Q_learning algorithm.

There are several parameters which define the learning evolution:

- γ: discount rate [0 1]. Concerning the maximization of future rewards. If γ=0, the agent is "myopic" in being concerned only with maximizing immediate rewards.
- α: learning rate [0 1].
- ϵ: random action probability [0 1]. (Exploitation / Exploration dilemma) The agent needs to explore new actions in order to find the optimal ones.

```
1. Initialize $\hat{Q}(s,a)$ arbitrarily
2. Repeat:
  (a) $s_t \leftarrow$ the current state
  (b) choose an action $a_t$ that maximizes $\hat{Q}(s_t,a)$ over all $a$
  (c) carry out action $a_t$ in the world with probability $(1-\varepsilon)$,
      otherwise apply a random action (exploration)
  (d) Let the short term reward be $r_t$, and the new state be $s_{t+1}$
  (e) $\hat{Q}(s_t,a_t) = \hat{Q}(s_t,a_t) + \alpha[r_t + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1},a_{t+1}) - \hat{Q}(s_t,a_t)]$
```

Figure 4. Q_learning algorithm.

Due to its use of finite spaces, Q-learning has a considerably large learning time and memory requirement. More sophisticated methods [16,17] implement a parameterized Q-function which enables generalization between states and actions. In this paper a continuous implementation of the algorithm was used. A neural network approximates the Q_function and its weights are updated according to the backpropagation algorithm or direct Q_learning [15]. There is no convergence proof of this continuous implementation. However, with suitable network configuration and parameter selection, the algorithm demonstrated to converge. The continuous Q_learning algorithm structure is showed in figure 5.

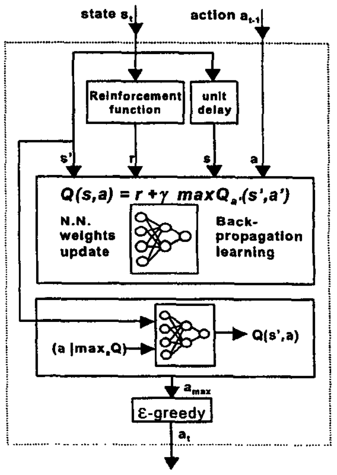

Figure 5. Continuous Q_learning algorithm structure.

The neural network approximates the Q_function:

$$Q(s,a) = r + \gamma \max Q_{a'}(s',a')$$

therefore, its inputs are the continuous states and actions, and the output is the Q_value. A reinforcement function associates each state with a reward "r" (-1, 0 and 1). In order to find the action that maximizes the Q_value, the network evaluates all the possible actions that could be applied. Although actions are continuous, a finite set, which guarantee enough resolution, is used.

4 Experimentation with an AUV

The kind of robot which we are working on is an Autonomous Underwater Vehicle (AUV) called URIS (Underwater Robotic Intelligent System). The proposed application consists of following a target by means of a camera and avoiding obstacles using a set of sonar sensors. The AUV must act as an autonomous camera recording all the movements of the target without colliding or losing the target. This application was designed to be carried out in a swimming pool where light absorption does not apply. In this paper, the application is fulfilled using realistic simulations. Further work will be based on real experiments.

4.1 The URIS Vehicle

URIS is a small-sized non-holonomic AUV designed and built at the University of Girona. The hull is composed of a Ø350mm stainless steel sphere, designed to withstand pressures of 4 atmospheres (40 meters depth). The spherical shape simplifies the construction of a dynamic model of the vehicle which is very useful for simulation of missions in the laboratory. On the outside of the sphere there are 4 thrusters (2 in X direction and 2 in Z direction). Due to the stability of the vehicle in *pitch* and *roll*, there are four degrees of freedom; X, Y, Z and Yaw.

4.2 The Behavior-based Architecture

To accomplish this mission a Behavior-based architecture with three behaviors was designed. Each behavior has its own input from sensors and generates a 3D-speed vector defined by (*u, w, r*). In association with this response, the behavior generates the activation level which determines the final robot movement. Figure 6 shows the schema of the architecture. The three behaviors are:

- *Obstacle avoidance.* The goal is to avoid any obstacles perceived by means of 7 sonar sensors, see figure 7. The behavior is learnt using a continuous Q_learning algorithm for each DOF (x,y,z). A reinforcement function gives negative rewards depending on the distance at which obstacles are detected. The activation level is also proportional to the proximity of obstacles.
- *Target following.* The behavior follows the target using a video camera pointed towards X-axis, see figure 7. A real-time tracking board based on chromatic characteristics gives the relative position of the target. The behavior is learnt using a continuous Q_learning algorithm for each DOF (x,y,z). The reinforcement function gives negative rewards when the target moves away from the position X=5, Y=0 and Z=0, relative to the on-board coordinate system. The activation level is 1 when the target is detected, alternatively, it is 0.

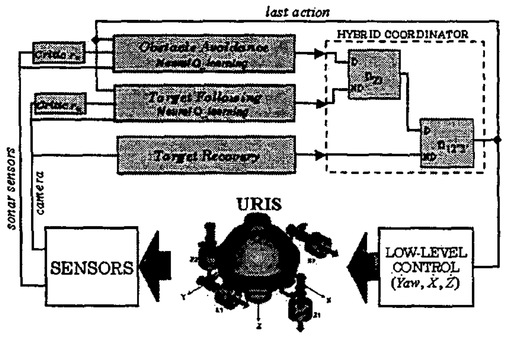

Figure 6. Schema of the architecture.

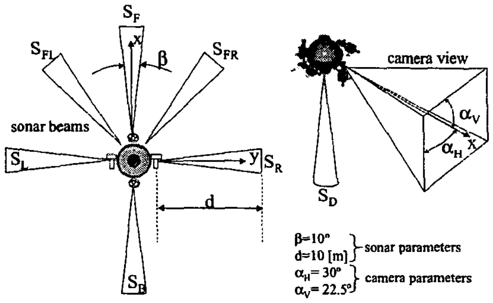

Figure 7. Sonar transducer and video camera layout.

- *Target recovery. The goal of this behavior is to recover* the target when it disappears from the camera view. Considering that the dynamics of the vehicle are relatively slow, we have adopted a very simple policy. When the tracking system loses the target, the behavior spins and moves the vehicle vertically in the direction last seen. This behavior is not learned but preprogrammed. The activation level is contrary to that of target following behavior.

5 Simulated Results

An environment called DEVRE [18] (Distributed Environment for Virtual and/or Real Experimentation) was developed to control, design and implement missions. DEVRE is an integrated software platform composed of three modules: (1) the *Object Oriented Control Architecture for Autonomy* (OOCAA) [19], which is in charge of controlling the vehicle at high and low levels; (2) the *Human Machine Interface* (HMI); and (3) the *Mathematical Model of the Vehicle and Virtual Environment (MMVVE).* The latter module simulates the vehicle according to the actions sent by the OOCAA module and provides a virtual representation of an underwater environment, see figure 8. It also simulates sensors (sonar transducers, video camera, etc.) according

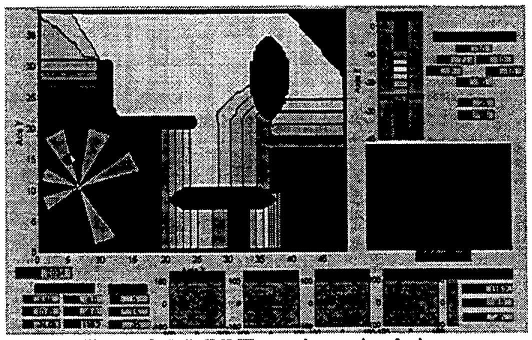
Figure 8, MMVVE running a simulation.

to the position of the vehicle in the environment and it uses a hydrodynamic model of an AUV [20] with the identified parameters of URIS.

The architecture proposed above was implemented in the OOCAA. A structured 3-dimensional environment was designed and used by the MMVVE, see figure 8. A moving target was introduced carrying out a 3D closed path repeatedly. The velocity of the target changed between 0 and 0.17 m/s (60% of the maximum velocity of URIS).

"Target following" and "Obstacle avoidance" behaviors were implemented using the neural Q_learning algorithm. Each degree of freedom was implemented independently with its inputs/outputs and rewards. At the beginning, each behavior was learnt alone, without the influence of the other. The number of iterations required to learn each DOF was approximately 2000 (Sample time=1s). Figure 9 shows the evolution of the "x" DOF of the "target following" behavior during its training. It can be seen how the algorithm explores the action space and learns how to track the target. Figure 10 shows the state/action mapping of two behaviors after the learning phase.

Once the 3 DOFs of both behaviors were completely learnt, the mission was tested. Figure 11 shows the tracking error evolution during the mission. When the vehicle was close to an obstacle, the obstacle avoidance behavior took partial control of the vehicle, and therefore, the tracking error increased. However, the hybrid coordination system generated a cooperative response between both behaviors, and the target was not lost.

Many simulation episodes were done in order to find the optimal neural network configuration. Finally a 3 layer neural network was used. The parameters and specifications of the *"obstacle avoidance"* and *"target following"* behaviors can be seen in tables 1 and 2 respectively.

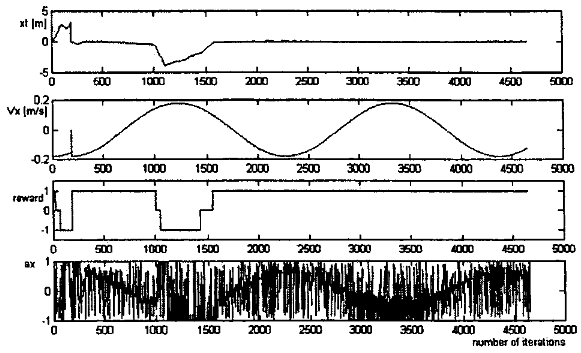

Figure 9. Learning evolution of the "target following" behavior (x axis).

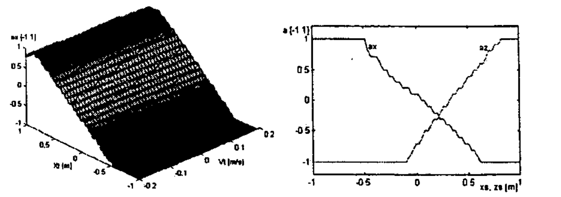

Figure 10. State/action mapping of the "target following" (x axis) and "obstacle avoidance" (x and z axis) behaviors.

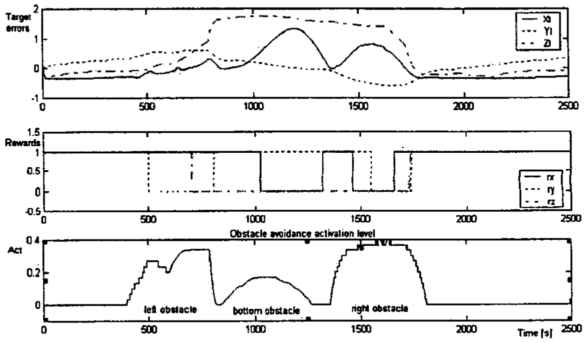

Figure 11. Tracking error evolution during a mission.

OBSTACLE AVOIDANCE BEHAVIOR	
Input variables	8 sonar transducers values
Codification	x_o, y_o, z_o (gravity center of the perceived obstacles, [-4 4] m)
Output variables	a_x, a_y, a_z (desired vehicle speed in x,y,z [0 1])

Critic function	If dist > 3.2 m : $r_t = 1$
	else if dist > 1.6 m : $r_t = 0$
	else $r_t = -1$ (dist = abs(x_o) or abs(y_o) or abs(z_o))

Behavior activation	act=dist/3 (if act>1, act =1)
Q_learning param.	$\alpha = 0.1; \gamma = 0.9; \epsilon = 0.2$
Neural Network	*inputs : 2 (i.e.: x_o, a_x) outputs : 1 (Q_val)* *layers*: 1- 5 neurons; sigmoidal act. function 2- 3 neurons; sigmoidal act. function 3- 1 neuron; lineal act. function

Table 1. Obstacle avoidance behavior specifications.

TARGET FOLLOWING BEHAVIOR	
Input variables	- x_t, y_t, z_t (errors between the target position and the target desired location, [-3 3]m) - v_{xt}, v_{yt}, v_{zt} (target velocity estim. [-0.3 0.3] m/s)
Output variables	- a_x, a_y, a_z (desired vehicle speed in x,y,z [0 1])

Critic function	If dist > 2 m : $r_t = -1$
	else if dist > 0.5 m : $r_t = 0$
	else $r_t = 1$ (dist = abs(x_t) or abs(y_t) or abs(z_t))

Behavior activation	act=1 if the target is visible, alternatively 0.
Q_learning param.	$\alpha = 0.1; \gamma = 0.9; \epsilon = 0.2$
Neural Network	*inputs : 3 (i.e.: x_t v_{xt} a_x) outputs : 1 (Q_val)* *layers:* 1- 4 neurons; sigmoidal act. function 2- 2 neurons; sigmoidal act. function 3- 1 neurons; lineal act. function

Table 2. Target following behavior specifications.

6 Conclusions and Future Work

This paper has proposed a hybrid coordination method for Behavior-based control architectures. The method has been tested in a simulated experiment. The architecture has been implemented using a continuous implementation of the Q-learning algorithm. The simulated results showed the feasibility of the hybrid approach as well as the convergence of the learning algorithm. The proposed hybrid coordination demonstrated as behaving with the robustness of competitive coordinators and with the optimized paths of cooperative ones. The neural network implementation of the Q_learning algorithm also demonstrated to converge to the optimal policy, obtaining maximum rewards.

Future work will concentrate on the realization of real experiments and on the improvement of the RL-based behaviors in order to learn simultaneously all the behaviors and to use only one RL function for each behavior.

Acknowledgements

The authors wish to acknowledge the Spanish Government for the financial project CICYT: MAR99-1062-C03-02 and the Autonomous Systems Laboratory of the University of Hawaii for the research collaboration.

References

[1] Arkin, R. Behavior-based Robotics. MIT Press, 1998.

[2] Brooks, R. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, vol. RA-2, is.1, pp.14-23. 1986.

[3] Maes, P. Situated Agents Can Have Goals. *Robotics and Automation Systems*, vol. 6, pp. 49-70, 1990.

[4] Arkin, R. C. Motor schema-based mobile robot navigation. *International Journal of Robotica Research*, vol. 8, is. 4, pp. 92-112, 1989.

[5] Steels, L. Building agents with autonomous behaviour systems. The artificial route to artificial intelligence. *Building situated embodied agents*. Lawrence Erlbaum Associates, New Haven, 1993.

[6] Carreras, M., Batlle, J., Ridao, P. and Roberts, G.N.. An overview on behaviour-based methods for AUV control. *MCMC2000, 5th IFAC Conference on Manoeuvring and Control of Marine Crafts.* Aalborg, Denmark, August 2000.

[7] Carreras, M.. An Overview of Behaviour-based Robotics with simulated implementations on an Underwater Vehicle. *University of Girona, Spain. Informatics and Applications Institute.* Research report: IIiA 00-14-RR. October, 2000.

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

[9] Watkins, C.J.C.H., and Dayan, P. Q-learning. *Machine Learning*, 8:279-292, 1992.

[10] Maes, P. and Brooks, R. Learning to coordinate behaviors. In *Proceedings of the Eighth AAAI*, pages 796-802. Morgan Kaufmann, 1990.

[11] Gachet, D., Salichs, M., Moreno, L. and Pimental, J. Learning Emergent tasks for an Autonomous Mobile Robot, *Proceedings of the International Conference on Intelligent Robots and Systems (IROS '94)*, Munich, Germany, September, pp. 290-97, 1994.

[12] Mahadevan, S. and Connell, J. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55:311-365, 1992.

[13] Shackleton, J. and Gini, M. Measuring the Effectiveness of Reinforcement Learning for Behavior-based Robots. *Adaptive Behavior*, 1997.

[14] Touzet, C. Neural reinforcement learning for behaviour synthesis. In*: Robotics and Autonomous Systems*, 22, 251-281, 1997.

[15] Baird, K. Residual Algorithms: Reinforcement Learning with Function Approximation. *Machine Learning: Proceedings of the Twelfth International Conference*, San Francisco, USA, 1995.

[16] Gaskett, C., Wettergreen, D. and Zelinsky, A. Q-learning in continuous state and action spaces. In *Proc. of the 12th Australian Joint Conference on Artificial Intelligence*, Sydney, Australia, 1999.

[17] Takahashi, Y., Takada, M. and Asada, M. Continuous Valued Q-learning for Vision-Guided Behavior Acquisition. In *International Conference on Multisenso Fusion and Integration for Intelligent Systems*, pages 716-721, 1999.

[18] Ridao, P., Batlle, J., Amat, J. and Carreras, M. A distributed environment for virtual and/or Real Experiments for Underwater Robots. *International Conference on Robotics and Automation ICRA 2001*, Seoul, Korea, 2001.

[19] Ridao, P., Carreras, M., Batlle,, J. and Amat, J. O2CA2: A New Hybrid Control Architecture for A Low Cost AUV. *Proc. of the Control Application in Marine Systems*, Scotland, 2001.

[20] Fossen, T. I. *Guidance and Control of Ocean vehicles.* John Wiley & Sons, 1995.