Hybrid Coordination of Reinforcement Learning-based Behaviors for AUV Control

M. Carreras, J. Batlle, P. Ridao and J. Pagès

Informatics and Applications Institute. University of Girona
{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 will demonstrate 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. Q(λ)-learning was used extending the one-step learning of the popular Q-learning to n-steps. Realistic simulations were carried out. Results showed the good performance of the hybrid method on behavior coordination as well as on increasing and improving behavior learning.

Keywords: Behavior-based Robotics, Reinforcement Learning, Autonomous Underwater Vehicles.

1 Introduction

Since the appearance of Behavior-based Robotics [1], 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. 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 robot. 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*.

After programming an Autonomous Underwater Vehicle (AUV) for a 3D-navigation mission using these methodologies some disadvantages (section 2) were found [4,5]. In this paper we propose a *hybrid* approach between competitive and cooperative coordination systems with the aim of taking advantage of both. 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 [9] for robot learning, behaviors were implemented using this technique. Specifically, the Q(λ)-learning [13] algorithm was applied, which is an extension of the popular one-step Q-learning [23] to n-steps. As a second purpose, this paper explores the influence of the hybrid coordination method on the learning algorithm, considering that reinforcements must be distributed proportionally to behavior influences.

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 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 introduces the basics of Behavior-based Robotics. Section 3 describes the proposed hybrid method. Section 4 summarizes the fundamentals of Reinforcement Learning and its relation with Behavior-based Robotics. Section 5 describes Q(λ)-learning. In section 6, the application to test the hybrid coordination method is detailed. In section 7, simulation results are given. And finally, conclusions and future work are presented in section 8.

2 Coordination in Behavior-based Robotics

Behavior-based [1] control architectures are a bottom-up approach inspired by biology where a collection of behaviors act in parallel, achieving goals. There are a few basic principles which provide the keys to success with this methodology [14]: parallelism, modularity, agent situatedness/embeddedness and behavior emergence. 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, see figure 1. A coordinator is needed in order to send only one command at a time to the motors. There are two primary coordination mechanisms to assemble behaviors:

- *Competitive methods.* The output is the selection of a single behavior. The coordinator chooses only one behavior to control the robot. Principal methods are suppression networks such as Subsumption architecture, action-selection and voting-based coordination .
- *Cooperative methods.* The output is the superposition of the force gradients given by all the behaviors. Behaviors which generate a stronger output will impose a greater influence on the final behavior of the robot. Principal methods are vector summation (potential fields) and behavioral blending.

After experimentation with several architectures [4,5], it has been noted that, depending on the coordination methodology, some advantages and disadvantages appear in the control performance of an autonomous vehicle. Competitive methods showed good robustness and modularity when adding new behaviors. However, an bad trajectory is found when there is a continuous change of the active 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).

3 Hybrid Coordination for a Behavior-based Control Architecture

Due to the disadvantages of both 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 2. 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*. 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

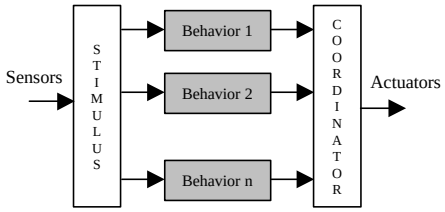

Figure 1. Structure of a Behavior-based architecture.

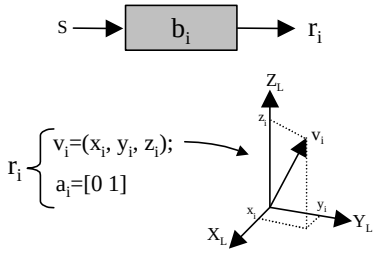

Figure 2. Normalized output of a behavior.

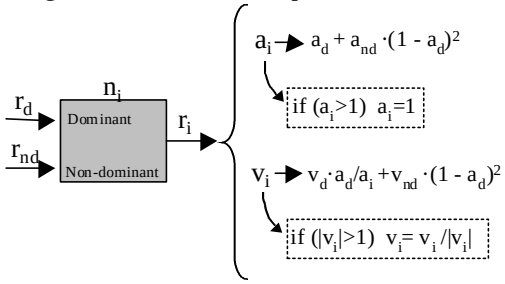

Figure 3. Hierarchical hybrid node.

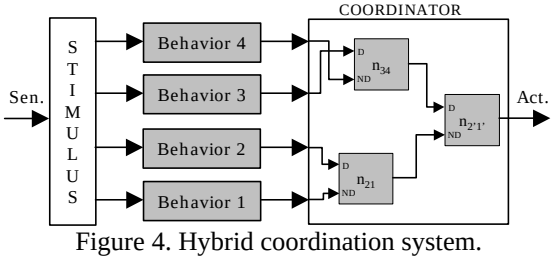

Figure 4. Hybrid coordination system.

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. The proposed node to coordinate behaviors can be seen in figure 3.

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. Using these nodes all behaviors can be coordinated. 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 by hierarchically classifying each behavior depending on its priority. A disposition of the whole coordination system using hierarchical hybrid nodes can be seen in figure 4.

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 [4] showing its good path performance, robustness and modularity controlling an AUV in a 3D-navigation mission. The method has been compared with 4 well-known behavior-based architectures: Subsumption [3], Action Selection Dynamics [10], Schema-based approach [2] and Process Description Language [20].

4 Reinforcement Learning and Behavior-based control architectures

When programming a Behavior-based system, there are some unknown parameters which cannot be identified without experimentation. This requires a great number of experiments until the architecture can work autonomously. 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. There isn't yet an established methodology to develop adaptive behavior-based systems. A commonly used approach is *Reinforcement Learning* [24].

Reinforcement Learning (RL) [9,19] is a class of learning algorithm in which 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. RL does not use any knowledge database as in most forms of machine learning. For this reason, this class of learning is suitable for robotics when online learning without information about the environment is required. Aspects such as delayed rewards and the trade-off between exploration and exploitation characterize RL. Most of the techniques are based on *Markov Decision Processes* (MDPs).

The most popular RL technique applied in robotics for its simplicity as well as effectiveness is Q-learning [23]. This technique is model-free (transitions between states and reinforcements are not learned) and an off-policy method (the optimal policy is learned in the long run independently of the policy followed). Q-learning, as well as most RL techniques, is based on *finite* MDP. This requires that state and action spaces be finite. The algorithm uses the states perceived (s), the actions taken (a) and the reinforcements received (r) to update the values of a table, denoted as $Q(s,a)$. Under appropriate conditions, the Q values converge to a *greedy policy*, in which the maximum Q value for a given state points to the optimal action. Q-learning is a one-step learning algorithm, so that reinforcements are only applied to the last state-action pair. Due to its use of finite spaces, Q-learning has a considerably large learning time and memory requirement. More sophisticated methods [8,21] implement a parameterized Q-function which enables generalization between states and actions.

Reinforcement learning has been applied to various Behavior-based systems, most of them using Q-learning. In some cases, the RL algorithm was used to adapt the coordination system [7, 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, 17, 22]. 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.

5 Reinforcement Learning-based Behaviors

For its feasibility in online adaptation of robotic systems, a Reinforcement Learning algorithm was adopted to learn the independent behaviors of our control architecture. In the work presented in this paper, our main goal was to test the viability of the use of RL-based behaviors with the proposed hybrid coordination system. Unlike the works of [12,17,22], in which a competitive coordination system was used, our coordination method uses the hybrid proposal. This means that a proportional learning rate must be taken depending on the activation level of each behavior.

The RL algorithm taken was Q(λ)-learning, proposed by Peng and Williams [13]. The algorithm is an incremental multi-step Q-learning which extends the original one-step learning, allowing the spreading of reinforcements not only in the last state-action pair but also in the precedent pairs. This is very important in a real robot application, where the dynamics of the robot is slow and the time-response history define the whole behavior. Hence, the reinforcements must be applied to a set of past actions. Figure 5 summarizes the Q(λ)-learning algorithm.

The algorithm repeatedly updates the table Q(x,a), which in the long run will contain the optimal action for a given state. There is also the table Tr(x,a), which is the "activity" trace of the state-action pair (x,a) used to spread reinforcements to precedent actions. The inputs of the algorithms are r_t (reinforcement of action a_t) and x_{t+1} (state $t+1$). The output is the action a_{t+1} to be taken in $t+1$. There are several parameters which define the learning evolution:

- l: eligibility trace [0 1]. Used to determine the credit assignment distribution. A Q(0)-learning acts as Q-learning. A Q(1)-learning spreads credit to all past actions.
- g: 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.
- a: learning rate [0 1].

In order to adapt our hybrid coordination system to

1. $\hat{Q}(x,a) = 0$ and $Tr(x,a) = 0$ for all x and a
2. Do forever:
   (a) $x_t \leftarrow$ the current state
   (b) choose an action a_t that maximizes $\hat{Q}(x_t,a)$ over all a
   (c) carry out action a_t in the world. Let the short term reward be r_t, and the new state be x_{t+1}
   (d) $e'_t = r_t + g\hat{V}_t(x_{t+1}) - \hat{Q}_t(x_t,a_t)$
   $\hat{V}(x) = argmax_a Q(x,a)$
   (e) $e_t = r_t + g\hat{V}_t(x_{t+1}) - \hat{V}_t(x_t)$
   (f) for each state-action pair (x,a) do
   - $Tr(x,a) = g\,l\,Tr(x,a)$
   - $\hat{Q}_{t+1}(x,a) = \hat{Q}_t(x,a) + a\,Tr(x,a)e_t$

   (g) $\hat{Q}_{t+1}(x_t,a_t) = \hat{Q}_{t+1}(x_t,a_t) + a\,e'_t$
   (h) $Tr(x_t,a_t) = Tr(x_t,a_t) + 1$

Figure 5. The Q(λ)-learning algorithm

the Q(λ)-learning algorithm, the learning rate a was modified depending on the influence of each behavior on the robot. This influence was summarized by a *control level* parameter c_{it} [0 1]. Therefore, the learning rate of behavior i at the time t was $a\ c_{it}$. Summarizing, each behavior is represented as a function that improves actions in the course of time:

$$a_{t+1} = \text{Q}(\lambda)\text{-learning}(x_{t+1}, r_t, c_{it}, \lambda_i, \gamma_i, \alpha_i)$$

where,

a_{t+1} : proposed robot action
x_{t+1} : current state (from the sensors)
r_t : past state-action pair reward
c_{it} : control level of behavior i
$\lambda_i, \gamma_i, \alpha_i$: learning evolution of behavior i

6 Experimentation with an AUV

An application to test the proposed Behavior-based architecture with Reinforcement Learning techniques was designed. The kind of robot which we are working on is an Autonomous Underwater Vehicle (AUV) called URIS (Underwater Robotic Intelligent System). The design of the experimentation task considered the difficulties found in underwater environments, such as positioning. 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.

6.1 The URIS vehicle

URIS is a small-sized non-holonomic AUV designed and built at the University of Girona. It was conceived as a low-cost underwater vehicle for research experimentation in control architectures and underwater computer vision. The hull is composed of a ∅350mm stainless steel sphere, designed to withstand pressures of 4 atmospheres (40 meters depth), see figure 6. 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 7 sonar sensors, a video camera and 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. The vehicle has an onboard Pentium PC-104 with the real-time operative system VxWorks.

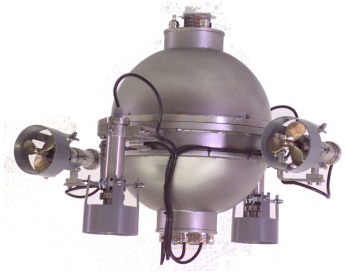

Figure 6. The underwater vehicle URIS.

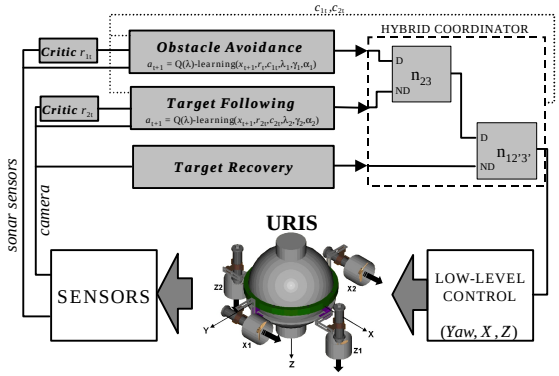

Figure 7. Schema of the architecture

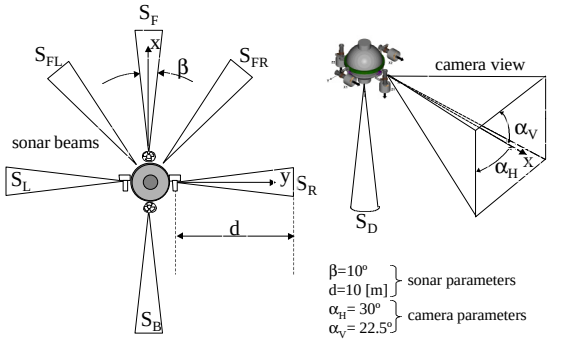

Figure 8. Sonar transducer and video camera layout.

6.2 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*). These velocities are relative to the on-board coordinate system (+X: prow and +Z: down). To fulfill the mission an absolute position is not necessary. In association with this response, the behavior generates the activation level which determines the final robot movement. Figure 7 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 8. The behavior is learnt using Q(λ)-learning. 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 8. A tracking algorithm based on chromatic characteristics gives the relative position of the target. The behavior is learnt using Q(λ)-learning. The reinforcement function gives negative rewards when the target moves away from the position X=4, 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.
- *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.

7 Simulated Results

The proposed hybrid coordination method with reinforcement learning-based behavior was simulated to test its feasibility. This section presents the realistic simulation platform used and the obtained results.

7.1 Realistic simulation platform

An environment called DEVRE [15] (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:

- *Object Oriented Control Architecture for Autonomy* (OOCAA) [16]. Software in charge of controlling the vehicle at high and low levels. It runs on the on-board PC.
- *Human Machine Interface* (HMI). This is an interface with a human operator executed on an external PC. It allows vehicle monitoring as well as tele-operation and is used for testing. An umbilical cable must be plugged into the robot.
- *Mathematical Model of the Vehicle and Virtual Environment (MMVVE).* This module simulates the vehicle according to the actions sent by the OOCAA module, see figure 9. It simulates sensors (sonar transducers, video camera, etc.) according to the position of the vehicle in the environment. Gaussian sensor noise is taken into account. It uses a hydrodynamic model of an AUV [6] with the identified parameters of URIS.

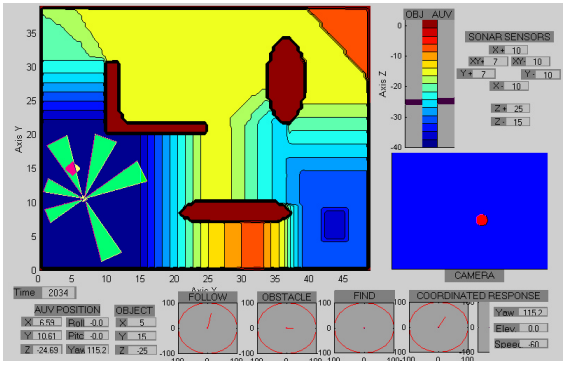

Figure 9, MMVVE running a simulation

7.2 Results

The architecture proposed above was implemented in the OOCAA. A structured 3-dimensional environment was designed and used by the MMVVE, see figure 9. A moving target was introduced carrying out a 3D closed path repeatedly. The velocity of the target changed between 0 and 0.2 m/s (40% of the maximum velocity of URIS).

Many simulation episodes were done in order to adjust the state and action dimensions of the two behaviors that were learnt. Due to the enormous quantity of entries of each learning table (more that a million for the *target tracking* behavior), a simplification was made. Each behavior was split in its horizontal and vertical movements, and two Q(λ)-learning algorithms were applied independently. Using this strategy, the behaviors slowly converged carrying out the mission. The definitions of the *obstacle avoidance* and *target following* behaviors can be found in tables 1 and 2 respectively.

In table 3 the parameters λ, γ and α, are showed as well as some indexes about the performance of the learning. Each simulation episode consisted of 100.000 iterations of 1second, in which behaviors converged. The indexes shown in table 3 are based on the averaged results of 4 episodes. The *average reward* index indicates the average of the reinforcements that receives the behavior when the algorithm converged. This is used to evaluate the performance of the learning algorithm. Figures 10 and 11 show the average reward of the last 500 iterations in the course of an episode for the two behaviors.

The *Null Q value* percentage indicates the entries of the learning table that were not updated. Figure 12, shows the averaged learned table for the obstacle avoidance behavior. In this figure, the null state-action pairs can be seen (Q=0). Finally, the n*umber of iterations* percentage measures the activity that had the behavior during an episode. Using the proposed hybrid coordinator, this activity increases and consequently, the learned speed. For this reason, using the hybrid coordination method a behavior learns more continuously and faster than using a competitive learning method [12,17,22]. We would like also to emphasize the good performance showed by the hybrid coordinator in assembling behaviors, providing robustness as well as behavior fusion.

OBSTACLE AVOIDANCE BEHAVIOR	
Horizontal movements	
Input variables	6 horizontal sonar transducers
Codification	X and Y of the perceived obstacles
Critic function	If $\sqrt{X^2+Y^2} > 4$m : $r_t = 0$ else if $\sqrt{X^2+Y^2} > 2.5$ m : $r_t = -1$ else $r_t = -3$
Behavior activation	act=(6-$\sqrt{X^2+Y^2}$)/3; act:=[0 1]
Input states	13 states for X, range [-6 6]m 11 states for Y, range [-5 5]m Total = 143 states
Output variable	*u* and *r* normalized set-points
Output states	5 states for *u*, [-1 –0.5 0 0.5 1] 5 states for *r*, [-1 –0.5 0 0.5 1] Total = 25 states
Learning table dim.	143·25 = 3575 entries
Vertical movement	
Input variables	1 vertical sonar transducers
Codification	Z position of the perceived obstacle
Critic function	If Z> 4m : $r_t = 0$ else if Z> 2.5 m : $r_t = -1$; else $r_t = -3$
Behavior activation	act=Z/3; act:=[0 1]
Input states	13 states, range [0 6]m
Output variable	*w* speed normalized set-point
Output states	7 states, [-1 –0.6 -0.3 0 0.3 0.6 1]
Learning table dim.	13·7 = 91 entries

Table 1. Obstacle avoidance behavior definition.

TARGET FOLLOWING BEHAVIOR	
Horizontal movements	
Input variables	X, Y target positions from tracking board respect the target desired location (4,0,0)m
Critic function	If $\sqrt{X^2+Y^2} > 9$m : $r_t = -7$ else if $\sqrt{X^2+Y^2} > 5$m : $r_t = -5$ else if $\sqrt{X^2+Y^2} > 1$m : $r_t = -3$ else $r_t = 0$
Behavior activation	act=1 if the target is visible, alternatively 0.
Input states	20 states for X, range [0 20]m 11 states for yaw=atan2(Y,X), range ±30º Total = 220 states
Output variable	*u* and *r* normalized set-points
Output states	5 states for *u*, [-1 –0.5 0 0.5 1] 9 states for *r*, from [-1 1] Total = 45 states
Learning table dim.	220·45 = 9900 entries
Vertical movement	
Input variables	X, Z target positions from tracking board respect the target desired location (4,0,0)m
Critic function	If Z> 9m : $r_t = -7$ else if Z> 5m : $r_t = -5$ else if Z> 1m : $r_t = -3$; else $r_t = 0$
Behavior activation	act=1 if the target is visible, alternatively 0.
Input states	11 states for pitch=atan2(Z,X), range [-22.5 22.5]º
Output variable	*w* speed normalized set-point
Output states	11 states, [-1 1]
Learning table dim.	11·11 = 121 entries

Table 2. Target following behavior definition

	Obstacle Avoidance		Target Following	
	Horizon.	Vertical	Horizon.	Vertical
λ	0.8	0.8	0.4	0.4
γ	0.7	0.7	0.3	0.3
α	0.2	0.2	0.1	0.1
Average Reward	-0.35	-0.06	-3.43	-0.97
Null Q values	32%	17%	27%	22%
Number of iter.	42%	36%	31%	31%

Table 3. Q(λ)-learning parameters and performance indexes averaged from 4 episodes of 100.000 iterations.

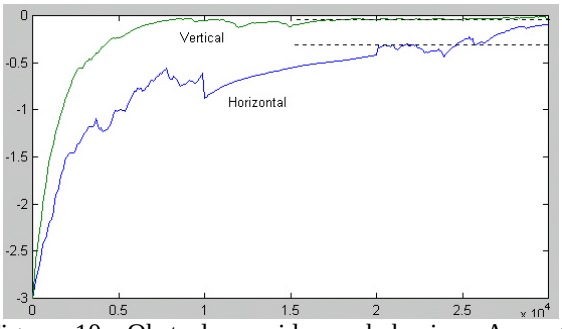

Figure 10. Obstacle avoidance behavior. Average reward of the last 500 iterations during an episode of 100.000 iterations. The horizontal and vertical learning modules are showed.

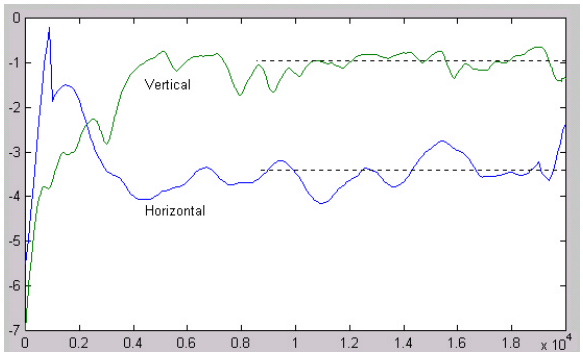

Figure 11. Target Following behavior. Average reward of the last 500 iterations during an episode of 100.000 iterations. The horizontal and vertical learning modules are showed.

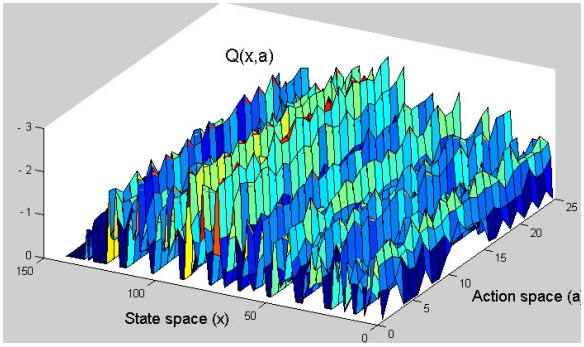

Figure 12. Learned table for the horizontal obstacle avoidance behavior.

Concluding this section, it has to be noted that the use of a finite MDP Reinforcement Learning, such as Q(λ)-learning, caused a very slow learning speed (more than a day) demonstrating its impracticability in a real experiment.

8 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 Q(λ)-learning, a Reinforcement Learning algorithm.

The simulated results showed the feasibility of the hybrid approach, as well as the convergence of the learning algorithm to optimal behaviors. The proposed hybrid coordination demonstrated as behaving with the robustness of competitive coordinators and with the optimized paths of cooperative ones. Another interesting advantage is the increase of the learning speed of each behavior. The proportional influence of

each behavior on the robot caused a proportional learning which doesn't appear in competitive architectures. Finally, the non-practical convergence time of Q(λ)-learning for real experiments should be noted. As stated in the literature [19], generalization between states and actions, as well as faster algorithms are needed. The use of continuous spaces and parameterized learning functions together with real experimentation in a swimming pool constitutes our current and future work.

References

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

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

[3] 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.

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

[5] 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.

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

[7] 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.

[8] 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.

[9] Kaelbling, L. P. Reinforcement Learning A Survey. *Journal of Artificial Intelligence Research*, vol. 4, pp. 237-285, 1996.

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

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

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

[13] Peng, J. and Williams, R.J. Incremental multi-step Q-learning. *Machine Learning*, 22:283-290, 1996.

[14] Pfeifer, R. and Scheier, C. *Understanding Intelligence*. MIT Press, 1999.

[15] 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.

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

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

[18] Sutton, R. S. Reinforcement Learning. *The MIT encyclopedia of the cognitive sciences*, pp. 715-717. MIT Press, 1999.

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

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

[21] 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.

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

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

[24] Ziemke, T. Adaptive Behavior in Autonomous Agents. Presence, vol. 7, is. 6, pp. 564-587, 1998.