A Behavior-Based Scheme Using Reinforcement Learning for Autonomous Underwater Vehicles

Marc Carreras, Junku Yuh, *Fellow, IEEE*, Joan Batlle, and Pere Ridao, *Member, IEEE*

***Abstract*—This paper presents a hybrid behavior-based scheme using reinforcement learning for high-level control of autonomous underwater vehicles (AUVs). Two main features of the presented approach are hybrid behavior coordination and semi on-line neural-Q_learning (SONQL). Hybrid behavior coordination takes advantages of robustness and modularity in the competitive approach as well as efficient trajectories in the cooperative approach. SONQL, a new continuous approach of the Q_learning algorithm with a multilayer neural network is used to learn behavior state/action mapping online. Experimental results show the feasibility of the presented approach for AUVs.**

***Index Terms*—Autonomous underwater vehicle (AUV), behavior-based control, neural networks, reinforcement learning.**

I. Introduction

WITH continuous advances in control, navigation, artificial intelligence, material science, computer, sensor and communication, autonomous underwater vehicles (AUVs) have become very attractive for various underwater tasks. The autonomy is one of the most critical issues in developing AUVs. Various control architectures have been studied to help increase the autonomy of AUVs [1]–[5]. They could be categorized into three groups: deliberative architecture, behavior-based architecture, and hybrid architecture. Deliberative architectures are based on planning using a world model [6]. A mission is specified to achieve a set of goals and each goal is executed by a control system. They allow reasoning and making predictions concerning the environment. Data flows from sensors to the world model, which is used to plan new actions to be undertaken by the actuator. When dealing with a highly dynamic environment, the delay in the response time is the main drawback. Behavioral architectures are also known as reactive architectures or heterarchies [7]. The decomposition is based on the desired behaviors for the vehicle and missions are normally described as a sequence of phases with a set of active behaviors. The behaviors continuously react to the situation sensed by the perception system. The vehicle's global behavior emerges from the combination of the elemental active behaviors. The real world acts as a model to which the vehicle reacts, based on the active behaviors. As active behaviors are based on the sense-react principle, they are suitable for dynamic environments. However, since each behavior pursues its own goal, reaction actions issued by one behavior may cause another behavior to deviate from its respective goal. As a result, the vehicle behavior is, at times, unpredictable. Hybrid architectures take advantage of the two previous architectures while minimizing their limitations [8], [9]. They usually consist of three layers: the deliberative layer, the behavior-based layer, and the control execution layer. The deliberative layer has the goal of breaking down the mission to be accomplished into a set of tasks. The behavior-based layer has the goal of carrying out each task. The deliberative layer also acts over the behavior-based layer by configuring the particular set of behaviors and the priorities among the behaviors. The deliberative layer determines if the task is being accomplished properly by monitoring sensor information and the output of the behavior-based layer. The behavior-based layer generates outputs as inputs to the low-level controller in the control execution layer.

This paper focuses on the behavior-based layer of the hybrid architecture for high-level AUV control. Various methods have been proposed to address the common characteristics of a behavior-based system, such as behavior expression, design, encoding, and coordination. As the vehicle's global behavior emerges from the combination of the elemental active behaviors, a coordinator is needed to generate the single output of the behavior-based layer from the outputs of those active behaviors. If the output of the behavior-based layer is chosen from the output of the selected single behavior, the coordinator is classified as *competitive*. If the output is the superposition of several behavior responses, the coordinator is called *cooperative*. Carreras *et al.* [10] discussed advantages and disadvantages of each approach. The competitive method could offer more robustness than the cooperative method with respect to the behavior selection and modularity when adding new behaviors. The cooperative method with properly tuned parameters could generate more efficient trajectories than the competitive method when there are changes in the dominant behaviors. Among generic behaviors for AUVs include: transit (moves the vehicle along the point-to-point local path); repulsion (avoids obstacles); tracking (follows an object of interest or a sequence of given points); stroll (wanders around); navigation (generates the global path—sequence of way-points); and hover (keeps the vehicle position). The presented behavior-based layer consists of a set of behaviors and a *hybrid coordinator* taking advantages of each of competitive and cooperative approaches [11].

In practice, the design and tune-up of behaviors may not be an easy task and require a lot of experimentation, especially in unknown and time-varying environments. To overcome this difficulty, learning techniques are introduced. *Reinforcement*

Manuscript received June 12, 2002; revised March 6, 2004; accepted July 9, 2004. This work was supported by the ONR (N00014-97-1-0961 and N00014-00-1-0629), KRISO/KORDI via MASE, and the Spanish commission MCYT (DPI2001-2311-C03-01). Associate Editor: J. Leonard.

M. Carreras, J. Batlle, and P. Ridao are with the Institute of Informatics and Applications, University of Girona, 17071 Girona, Spain (e-mail: marcc@eia.udg.es; jbatlle@eia.udg.es; pere@eia.udg.es).

J. Yuh is with the National Science Foundation, Arlington, VA 22230 USA (e-mail: jyuh@nsf.gov).

Digital Object Identifier 10.1109/JOE.2004.835805

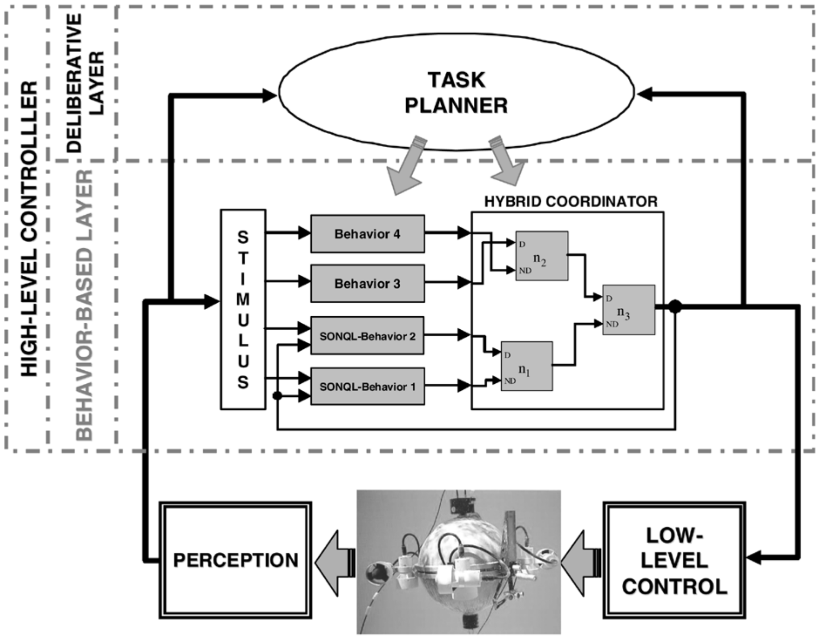


Fig. 1. Hybrid control architecture for AUVs.

learning (RL) has been used to learn the internal structure of the behaviors by mapping the perceived states to control actions while maximizing the accumulated future rewards [12]–[17]. Moreover, some researchers have used RL to adapt the behavior coordination system [18]–[22]. Most RL techniques are based on *finite Markov decision processes* (FMDP) resulting in finite state and action spaces. RL techniques are very attractive for on-line learning since they do not use any knowledge database. Classic RL algorithms based on FMDP using discrete variables may have the generalization problem when they are applied to systems with continuous variables (states or actions). It is also noted that the learning process could be damaged when the measured signals are related to the state of the environment due to noise or delay. Such environments would be considered as a partially observable MDP [23]. To overcome the difficulty due to the generalization problem, researchers have proposed to combine the RL algorithm with other techniques such as decision trees [24], CMAC function approximator [25], memory-based methods [26] and neural networks (NN) [27]. This paper describes a new learning algorithm, *semi-online neural-Q_learning* (SONQL) that is used to learn the internal state/action mapping of each behavior. SONQL uses the Q_learning algorithm with a NN function approximator. The *Q_learning* algorithm has been widely used for RL due to its good learning capabilities—on-line and off-policy features [28]. The on-line feature is the capability of learning at the same time when the learning information is being acquired by the system. The off-policy feature could be understood as the capability of learning without following any particular strategy or policy when the actions are selected. SONQL implements the Q-function directly into a NN, known as direct Q_learning [29]. While neural networks are nonlinear and offer a high generalization capability for complex tasks [30], learning in one area of the input space may not maintain the learned data of all possible areas of the input space known as the interference

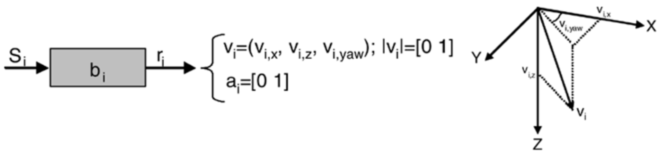


Fig. 2. The normalized vehicle control action v_i and the behavior activation level a_i constitute the behavior response r_i.

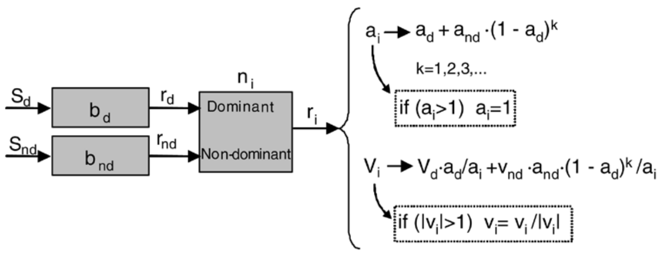

$a_i \rightarrow a_d + a_{nd} \cdot (1 - a_d)^k$
k=1,2,3,...
if $(a_i>1)$ $a_i=1$
S_d
b_d
r_d
S_{nd}
b_{nd}
r_{nd}
n_i
Dominant
Non-dominant
r_i
$V_i \rightarrow V_d \cdot a_d/a_i + v_{nd} \cdot a_{nd} \cdot (1 - a_d)^k / a_i$
if $(|v_i|>1)$ $v_i= v_i/|v_i|$

Fig. 3. HHCN.

problem [31]. A database of the most recent and representative learning samples is used to address the interference problem.

This paper presents experimental results on two AUVs, ODIN and URIS for a target following task, showing the SONQL-based behavior approach with hybrid coordination promising for AUV high-level control. The structure of the paper is as follows. Section II describes the behavior-based scheme with the hybrid coordination system. Section III describes the Semi-Online Neural-Q_learning algorithm. In Section IV, results are presented for a target following task with two different AUVs. Finally, the paper is concluded in Section V.

II. Behavior-Based Scheme

This paper considers a hybrid control architecture (Fig. 1), showing three layers: the deliberative layer, the behavior-based layer, and the control execution layer. The deliberative layer has

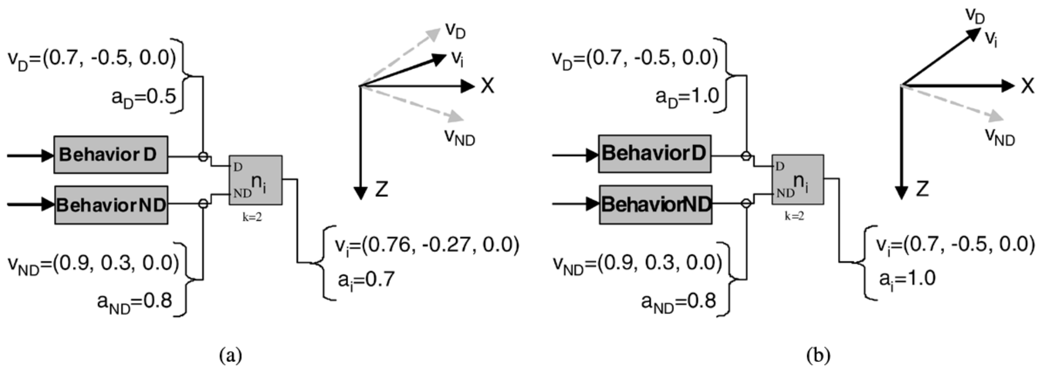


Fig. 4. Response of HHCN with 2-D l actions. (a) The node acts cooperatively when the dominant behavior is not fully activated. (b) In case the dominant behavior is completely activated, a competitive action is generated.

the goal of breaking down the mission to be accomplished into a set of tasks. Each task should be small enough to be described by the limited number of behaviors depending on various considerations such as computing power. If a task requires more than the limited number of behaviors, it should be further broken into a set of smaller tasks.

The behavior-based layer consists of a set of behaviors and a coordinator. The hybrid coordinator takes advantage of competitive and cooperative approaches [11]. The hybrid coordinator allows the coordination of a large number of behaviors without the need of a complex designing phase or tuning phase. The addition of a new behavior only implies the assignment of its priority with reference to other behaviors. The hybrid coordinator uses the priority and behavior activation level to calculate the output of the layer, which is the desired control action input to the low-level control system. Therefore, the response r_i of each behavior is composed of the activation level a_i and the control action v_i, as illustrated in Fig. 2. The activation level indicates the degree to which behavior wants to dominate in controlling the AUV. This degree is expressed by a numerical value from 0 to 1. In this paper, the control action represents the desired AUV velocity vector. The control action vector is normalized and its magnitude cannot be greater than 1.

The hybrid coordinator uses the behavior responses to compose a final action input. This process is executed at each sampling time of the high-level controller. The coordination system is composed of a set of nodes n_i. Each node has two inputs and generates a response which also has an activation level and a control action. By using these nodes, the whole coordination process is accomplished. After connecting node responses, a final response is generated and reescalated to the velocities of the vehicle as an input to the low-level AUV control system as described below.

Each node has a *dominant* and a *nondominant* input. The response connected to the dominant input will have a higher priority than the one connected to the nondominant. When the dominant behavior is completely activated $a_d = 1$, the response of the node will be equal to the dominant behavior. Therefore, in this case, the coordination node will behave competitively. However, if the dominant behavior is partially activated $0 < a_d < 1$ the two responses will be combined. The idea is that nondominant behaviors can modify the responses of dominant behaviors slightly when they are not fully activated. In this case, the node will behave cooperatively. Finally, if the dominant behavior is not activated $a_d = 0$ the response of the node will be equal to the nondominant behavior. These nodes are called *hierarchical hybrid coordination nodes* (HHCN) as its coordination approach changes depending on the activation level of the behaviors and the hierarchy or priority among them.

Fig. 3 shows the equations used to calculate the response of HHCN. The activation level will be the sum of the activation levels of the input responses, where the nondominant activation level has been multiplied by a reduction factor, $(1 - a_d)^k$ depending on the activation of the dominant behavior and on the value of the parameter k. If $k = 1$, the activation level linearly decreases as a_d increases. If more drastic reduction is desired, the value of k can be set to higher values. This parameter does not have to be tuned for each node and the same value could be used for all the coordination nodes. If the new activation level is larger than 1, the level is saturated to 1.

The control action is calculated in the same way as the activation level. Vector v_i is the sum of v_d and v_{nd}, applying the corresponding proportional factors. Therefore, each component of v_d is taken in the proportion of the activation level, a_d with respect to a_i while each component of v_{nd} is taken in the proportion of the reduced activation level with respect to a_i. If the module of v_i is larger than 1, the vector is rescaled to a magnitude equal to 1. An example of the use of HHCN is illustrated in Fig. 4. In this example, two different situations are depicted. In the first situation, the node acts cooperatively generating an action that is affected by the nondominant response. In the second situation, the dominant behavior is completely activated and the node acts competitively.

As described earlier, the hybrid coordinator is composed of a set of HHCNs which connect each pair of behaviors or nodes until a final response is generated. In order to build up this network of nodes, it is necessary to set up the priority among the behaviors. This hierarchy will depend on the task to be performed. Once the priorities are set usually by the mission designer, the hybrid coordinator is ready to use. Fig. 1 shows an example of a set of three nodes that coordinate four behaviors. A new behavior can be easily added by just setting the priority of the new behavior with respect to the others. The previous computer simulation study [32] shows that a behavior-based control architecture using the hybrid coordinator generates more efficient paths and offers more robustness and modularity in a 3-D AUV navi-

```
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) $\varepsilon$-greedy action, 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 $r_{t+1}$, and the new state be $s_{t+1}$
  (e)$\hat{Q}(s_t,a_t) = \hat{Q}(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1},a_{t+1}) - \hat{Q}(s_t,a_t)]$
```

Fig. 5. Q_learning algorithm.

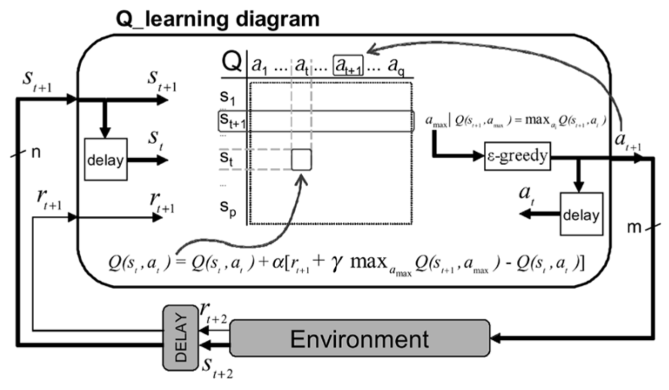

Q_learning diagram
s_{t+1}
s_{t+1}
s_t
r_{t+1}
r_{t+1}
n
delay
Q | a_1 ... a_t ... a_{t+1} ... a_q
s_1
s_{t+1}
s_t
s_p
$a_{\max} \mid Q(s_{t+1},a_{\max}) = \max_{a_t} Q(s_{t+1},a_t)$
ε-greedy
a_{t+1}
a_t
delay
m
$Q(s_t,a_t) = Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_{a_{\max}} Q(s_{t+1},a_{\max}) - Q(s_t,a_t)]$
DELAY
r_{t+2}
s_{t+2}
Environment

Fig. 6. Diagram of the Q_learning algorithm.

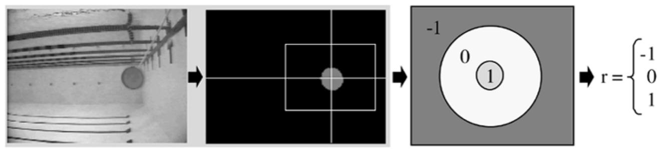


Fig. 7. Reinforcement function of the tracking behavior.

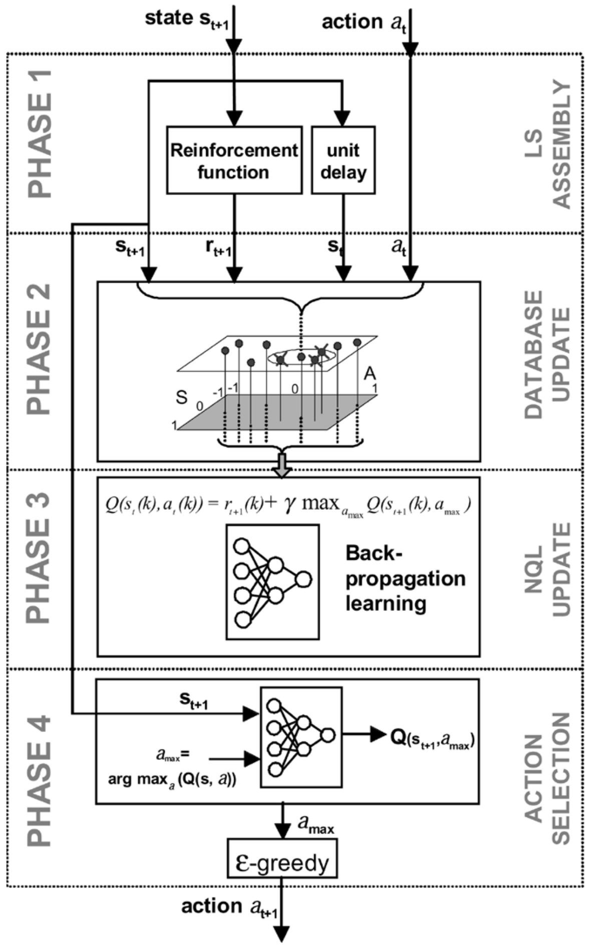


Fig. 8. Implementation of SONQL algorithm.

gation mission than four well-known behavior-based control architectures (subsumption [33], action selection dynamics [34], schema-based approach [35], and process description language [36]).

III. SONQL

SONQL could be considered as a continuous implementation of the Q_learning [15], [28]. Q_learning is a temporal difference algorithm based on the FMDP and is suitable for learning incrementally or on-line. Q_learning is also an off-policy algorithm. The off-policy feature could be understood as the capability of learning without following any particular strategy or policy when the actions are selected. This feature is very useful for AUVs. For example, if the algorithm proposes an action that would cause a collision, another behavior with higher priority will prevent it, and the learning algorithm will use the real executed action.

Q_learning uses the perceived states (s), the taken actions (a), and the received reinforcements (r) to update the values of a table, denoted as $Q(s,a)$ or Q-function. 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 *greedy action*. When the algorithm converges to the solution, the greedy policy will correspond to the *optimal* policy, in which the maximum Q value for a given state, points to the optimal action. The optimal action is the one that will maximize the accumulated future rewards, and therefore, will solve the reinforcement problem. Fig. 5 shows the Q_learning algorithm and Fig. 6 shows a diagram of its implementation.

There are several parameters that define the learning evolution:

- γ: discount rate [0 1]. Concerning the maximization of future rewards. If $\gamma = 0$, the agent maximizes only immediate rewards.
- α: learning rate [0 1].
- $\in$: random action probability [0 1]. Exploitation versus exploration dilemma. An explorative action is taken with probability $\in$ instead of the *greedy* action contained in the current Q-function. The final action is called $\in$-*greedy* action.

A neural network is used to address the generalization problem of Q_learning. The NN has the state and the action as inputs, and the one-dimensional (1-D) Q value as output with the following approximation function:

$$\hat{Q}(s_t,a_t) = r_{t+1} + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1},a_{t+1}). \tag{1}$$

To address the interference problem of the neural network, the SONQL uses a database of learning samples. The main goal of the database is to include a representative set of visited learning samples, which is repeatedly used to update the SONQL algorithm. The database also helps the learning speed.

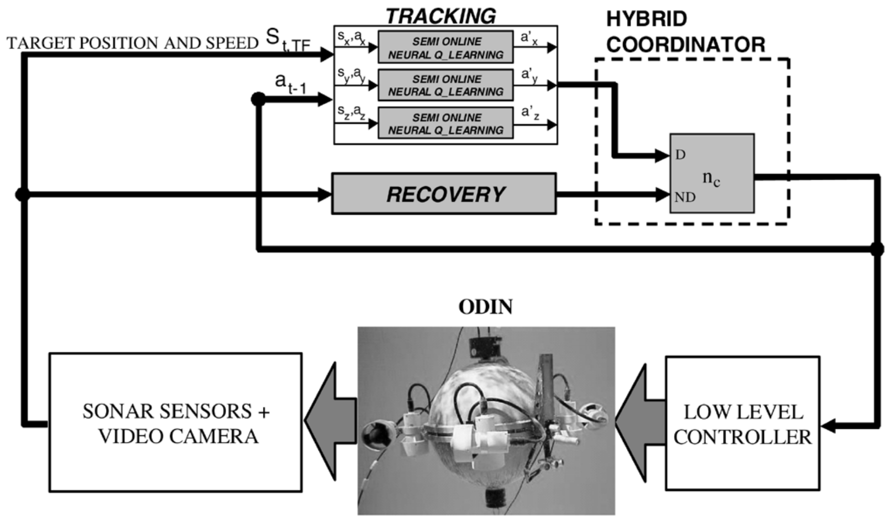


Fig. 9. ODINs target following task control system.

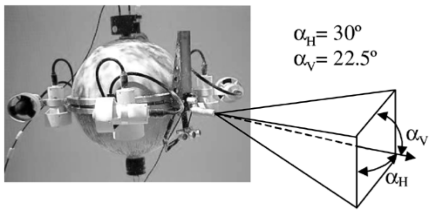


Fig. 10. Color video camera layout.

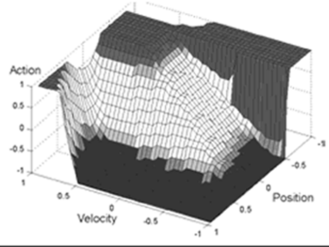
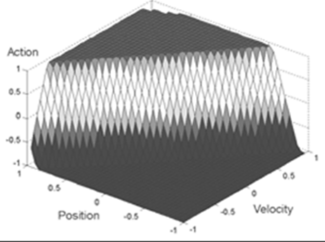
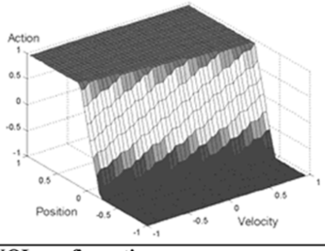
Each learning sample is composed of the initial state s_t, the action a_t, the new state s_{t+1} and the reward r_{t+1}. During the learning evolution, the learning samples are added to the database. Each new sample replaces old samples. The replacement is based on the geometrical distance between vectors $(s_t; a_t; r_{t+1})$ of the new and old samples. If the distance is less than a density parameter t for any old sample, the sample is removed from the database. The size of the database is, therefore, controlled by this parameter which has to be set by the designer. Once the algorithm has explored the reachable state/action space, a homogeneous representative set of learning samples is contained in the database.

The reinforcement function determines the policy to be followed by the behavior. Defining this function requires the knowledge of a human designer. The function associates each state with a reward "r" $= \{-1, 0, 1\}$. By associating the desired states with "$r = 1$" and the undesired with "$r = -1$," the algorithm learns how to act. Fig. 7 shows an example of the reinforcement function for the tracking behavior. In this figure in can be observed first the image acquired by the AUV, which is then used to find the horizontal and vertical position of the target in the image field. Finally, this position is taken by the reinforcement function, here represented as three 2-D regions separated by two circles, to compute the final reward r. The reinforcement function is the only internal part of each behavior that is experimentally set.

SONQL is implememted with the following steps (Fig. 8):

- **LS assembly**. The current learning sample (LS) is assembled. The state s_{t+1} and the last taken action a_t are received from the environment. The reward r_t is computed according to s_{t+1}, and, the past state s_t is extracted from a unit delay.

TABLE I
ODIN TRACKING BEHAVIOR SPECIFICATIONS

Sensors	color camera + target detection algorithm using color segmentation
Codification	$[f_x, f_y, f_z]$: target position + normalization [-1, 1]
Activation	if target detected: $a_f = 1$; else $a_f = 0$
X DOF	
State	
f_x : position	
fv_x : euler derivative	
Action	
a_{fx} : desired speed	
Reinforcement Function	
If $\|f_x\| < 0.15$: $r_{fx} = 1$	
else if $\|f_x\| < 0.4$: $r_{fx} = 0$	
else $r_{fx} = -1$	
SONQL parameters	**SONQL configuration**
$\alpha = 0.05$; $\gamma = 0.9$; $\in = 0.4$	inputs: 3 : $[f_x, fv_x, a_{fx}]$
Database size = up to 50	output: Q_value
learning samples	layer 1: 6 neurons (hyperbolic tangent a.f.)
	layer 2: 1 neuron (lineal a.f.)
Y DOF	
state	
f_y : position	
fv_y : euler derivative	
action	
a_{fy} : desired speed	
reinforcement function	
If $\|f_y\| < 0.2$: $r_{fy} = 1$	
else if $\|f_y\| < 0.5$: $r_{fy} = 0$	
else $r_{fy} = -1$	
SONQL parameters	**SONQL configuration**
$\alpha = 0.05$; $\gamma = 0.9$; $\in = 0.4$	inputs: 3 : $[f_y, fv_y, a_{fy}]$
Database size = up to 50	output: Q_value
learning samples	layer 1: 3 neurons (hyperbolic tangent a.f.)
	layer 2: 1 neuron (lineal a.f.)
Z DOF	
state	
f_z : position	
fv_z : euler derivative	
action	
a_{fz} : desired speed	
reinforcement function	
If $\|f_z\| < 0.2$: $r_{fz} = 1$	
else if $\|f_z\| < 0.4$: $r_{fz} = 0$	
else $r_{fz} = -1$	
SONQL parameters	**SONQL configuration**
$\alpha = 0.05$; $\gamma = 0.9$; $\in = 0.4$	inputs: 3 : $[f_z, fv_z, a_{fz}]$
Database size = up to 50	output: Q_value
learning samples	layer 1: 3 neurons (hyperbolic tangent a.f.)
	layer 2: 1 neuron (lineal a.f.)

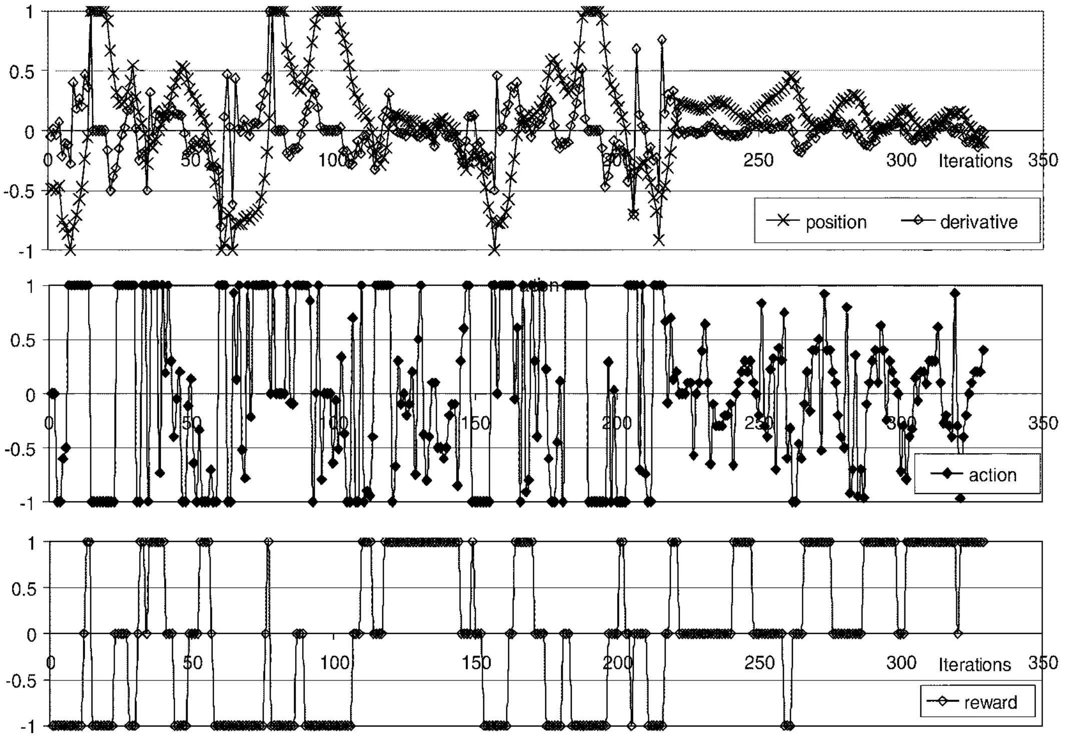


Fig. 11. Online learning evolution of the tracking behavior in the X dof. The states, actions, and rewards are shown.

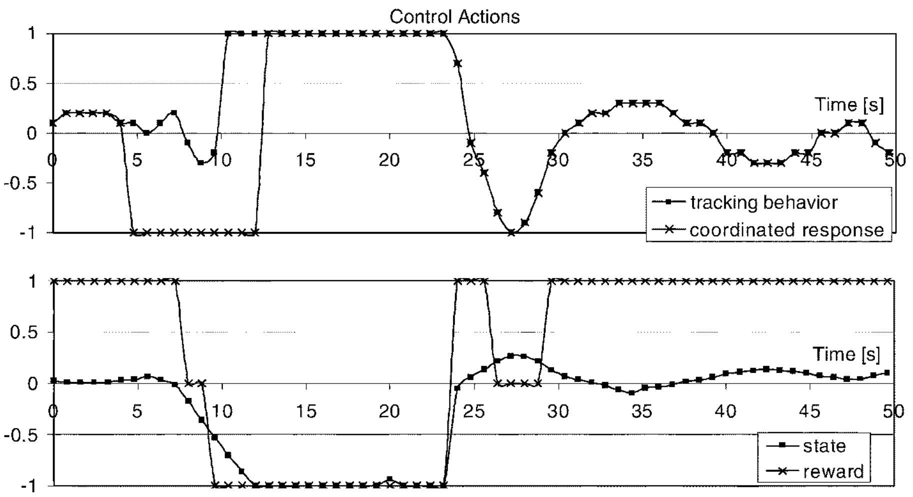


Fig. 12. Performance of the tracking behavior in the X dof. Once the coordinated action equalled the tracking action, the target was reached (state position $\cong 0$) and the maximum reward was achieved ($r = 1$).

- **Database update**. The database is updated with the new learning samples. Old samples are replaced by new samples based on the density parameter t as described earlier.
- **NQL update**. The weights of the NN in neural Q_learning (NQL) are updated according to the back-propagation algorithm and (1).
- **Action selection**. A new action a_{t+1} is computed by following the ε-*policy*. With probability $(1 - \varepsilon)$, the action taken is the one that maximizes the Q-function in the current state s_{t+1} (*greedy* action). Otherwise, a random action is generated. Since the NN is a continuous function, the greedy action is practically found by evaluating the Q values of a finite set of actions. The one with maximum Q value is the action chosen.

IV. Case Study: Target Following Task

The feasibility of the hybrid coordinator and SONQL-based behavior was investigated by experiment for a *target following* task. The first test was performed on ODIN that was developed at the Autonomous Systems Laboratory (ASL), the University of Hawaii [37], [38]. The ODIN is a sphere shaped AUV with eight thrusters (4 horizontal and 4 vertical). It is capable of maneuvering with six degrees-of-freedom (dof). The second test was performed on ODIN's sister AUV, URIS [39] that was built at the Institute of Informatics and Applications, the University of Girona after the fourth author's visit to ASL. The URIS is smaller than the ODIN but also a sphere shaped vehicle with only four thrusters (2 horizontal and 2 vertical). It is a nonholonomic system capable of moving in four dof.

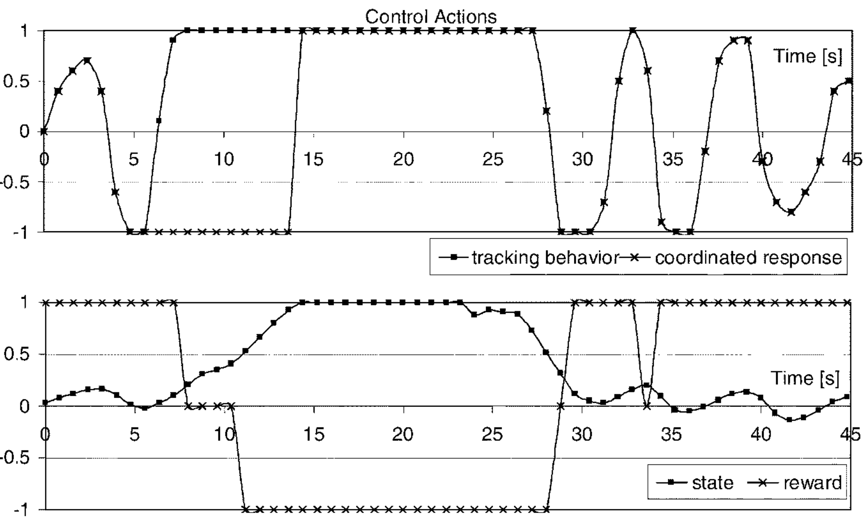


Fig. 13. Performance of the tracking behavior in the Y dof. Once the coordinated action equalled the tracking action, the target was reached (state position $\cong 0$) and the maximum reward was achieved $(r = 1)$.

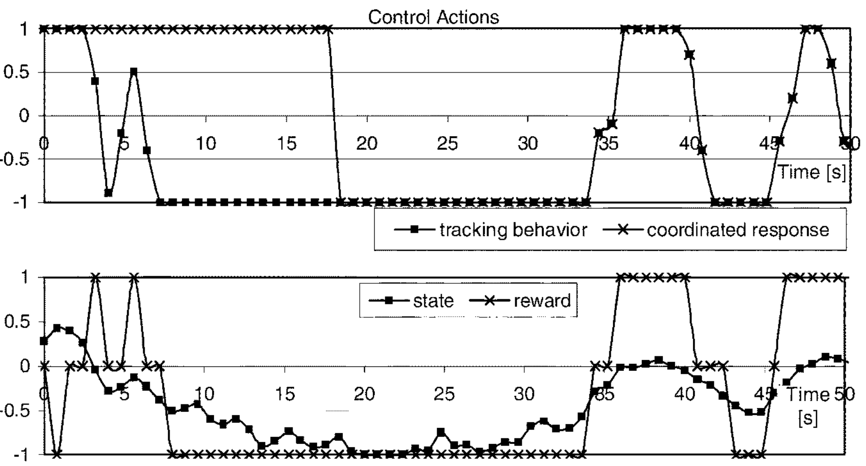


Fig. 14. Performance of the tracking behavior in the Z dof. Once the coordinated action equaled the tracking action, the target was reached (state position $\cong 0$) and the maximum reward was achieved $(r = 1)$.

A. ODIN

The first test was conducted to check the feasibility of the presented approach, especially SONQL for AUVs. The *target following* task consists of recovering and tracking an object (target) by means of a camera. Therefore this task could be accomplished with two behaviors: *tracking*, and *recovery*. For this test, the tracking behavior was implemented with the SONQL algorithm and the recovery behavior was manually programmed. The SONQL behavior receives information about the current state, usually from sensors, and it also receives the last action taken. This action is used by the SONQL algorithm to update the neural network weights. Each behavior generates a 3-D-speed vector and an activation level, which determined the final output according to the priority of behaviors. Once the coordination is done, the output is sent to the low level controller that controls the yaw angle, the depth and the surge and sway velocities. Fig. 9 shows ODINs target following task control system with two behaviors described here.

Tracking: The goal of this behavior is to follow an object using a video camera pointed toward X-axis (Fig. 10). As the mission is carried out in a pool, a simple segmentation algorithm is used to detect the position of a tinted target (Fig. 7). The behavior is

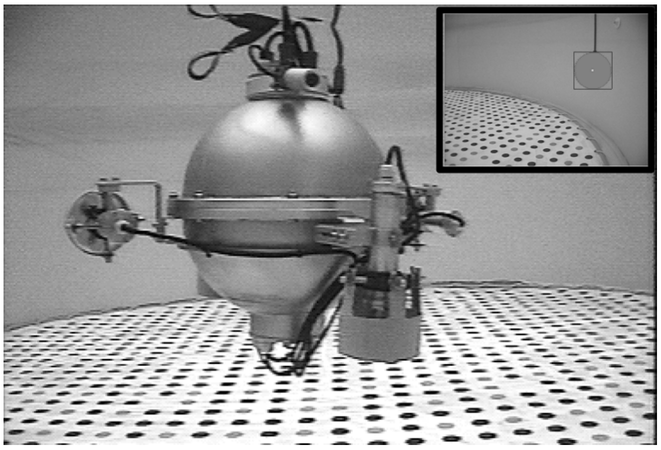


Fig. 15. URIS' underwater vehicle during the target following task in a water tank.

learned using the SONQL algorithm for each dof (x, y, z). Aside from the position of the target, its euler derivative is also used as the input. A reinforcement function was designed to give positive rewards $(r = 1)$ when the target is around the vehicle's relative position $x = 2, y = 0$ and $z = 0$. Otherwise, values like $r = 0$ or

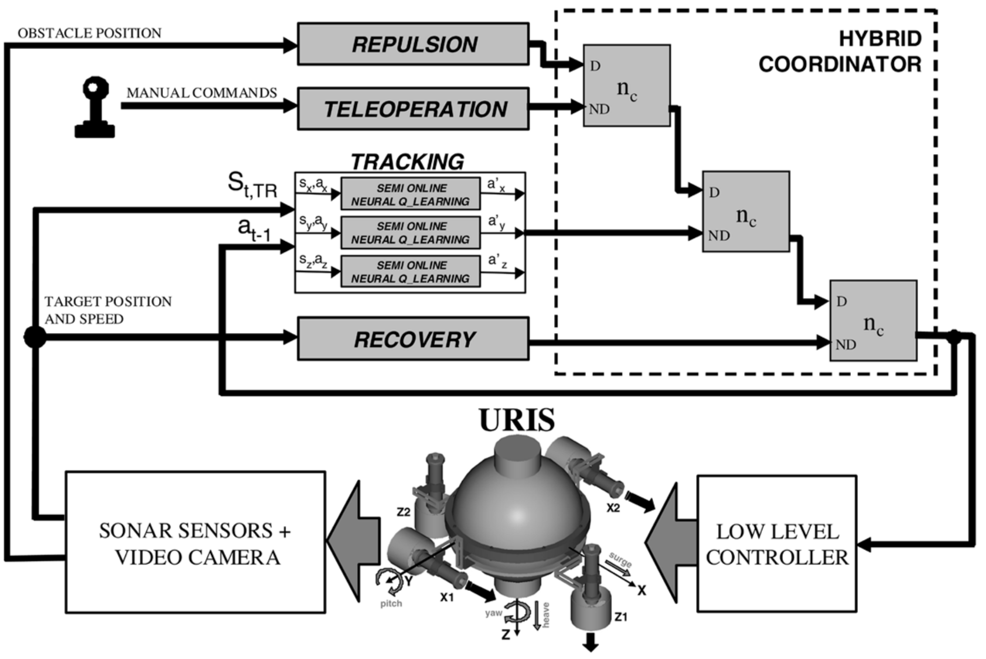


Fig. 16. URIS' target following task control system in the real experiments.

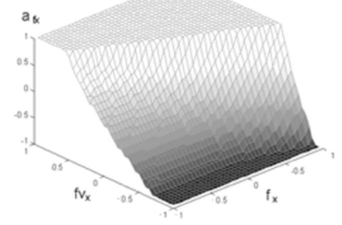
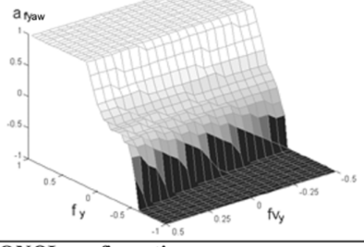
$r = -1$ are given. In this test, since this behavior is considered as a dominant behavior in the HHCN nodes, its activation level is 1 only when the target is detected. Otherwise it is 0.

Recovery: The goal of this behavior is to recover an object when it disappears from the camera view. Considering that the vehicle motion is relatively slow, a very simple policy can be used. When the tracking system is losing the target, the behavior moves the vehicle horizontally and vertically. This behavior is preprogrammed without using SONQL. The activation level is contrary to that of the tracking behavior.

The tracking behavior was implemented with three different SONQL algorithms (one for each dof). SONQL algorithms used 2-layer neural networks that were chosen after many trials. The number of layers and neurons, depends on the complexity and the number of dimensions of the Q-function to be approximated. For the hidden layers, a hyperbolic tangent function was used as the activation function. This function is antisymmetric and accelerates the learning process. The output layer had a linear activation function, which allows the NN to approximate any real value. The initialization of the NN weights were done randomly. In the experiment, 0.8 seconds were used for the sampling time of the behavior-based layer considering the overall control performance as well as the learning performance of the SONQL algorithm. The internal parameters and configuration used in the tracking behavior as well as the learned state/action mappings for each dof are shown in Table I.

Fig. 11 shows that the SONQL algorithm successfully learns the tracking behavior. The behavior was learned in less than 350 iterations (280 s). During the learning period, random actions helped the exploration of the space until the optimal policy was learned and the algorithm found the maximum reward. Once the 3 dof behaviors were learned, the target following task was tested. The vehicle was able to learn how to move in 3 dof achieving the target following task. Figs. 12–14 show ODINs performance in X, Y and Z, respectively, including the position of the target, the reward of the tracking behavior and the actions of the tracking behavior and the coordinated response (all the values are normalized from -1 to 1). It can be observed that when the coordinated action equals the tracking behavior

TABLE II
URIS TRACKING BEHAVIOR SPECIFICATIONS

Sensors	color camera + target detection algorithm using color segmentation
Codification	$[f_x, f_y]$: target position + normalization [-1, 1]
Activation	if target detected: $a_f = 1$; else $a_f = 0$
X DOF	
State	
f_x : position	
fv_x : euler derivative	
Action	
a_{fx} : desired speed	
Reinforcement Function	
If $-0.4<f_x<-0.1$: $r_{fx} = 1$	
else if $-0.1<f_x<0.0$: $r_{fx} = 0$	
else $r_{fx} = -1$	
SONQL parameters	**SONQL configuration**
$\alpha = 0.1$; $\gamma = 0.9$; $\in = 0.3$ Database size = up to 30 learning samples	inputs: 3 : $[f_x, fv_x, a_{fx}]$ output: Q_value layer 1: 6 neurons (hyperbolic tangent a.f.) layer 2: 1 neuron (lineal a.f.)
Y DOF	
state	
f_y : position	
fv_y : euler derivative	
action	
a_{fyaw} : desired speed	
reinforcement function	
If $\|f_y\| < 0.2$: $r_{fy} = 1$	
else if $\|f_y\| < 0.4$: $r_{fy} = 0$	
else $r_{fy} = -1$	
SONQL parameters	**SONQL configuration**
$\alpha = 0.1$; $\gamma = 0.9$; $\in = 0.3$ Database size = up to 30 learning samples	inputs: 3 : $[f_y, fv_y, a_{fyaw}]$ output: Q_value layer 1: 3 neurons (hyperbolic tangent a.f.) layer 2: 1 neuron (lineal a.f.)

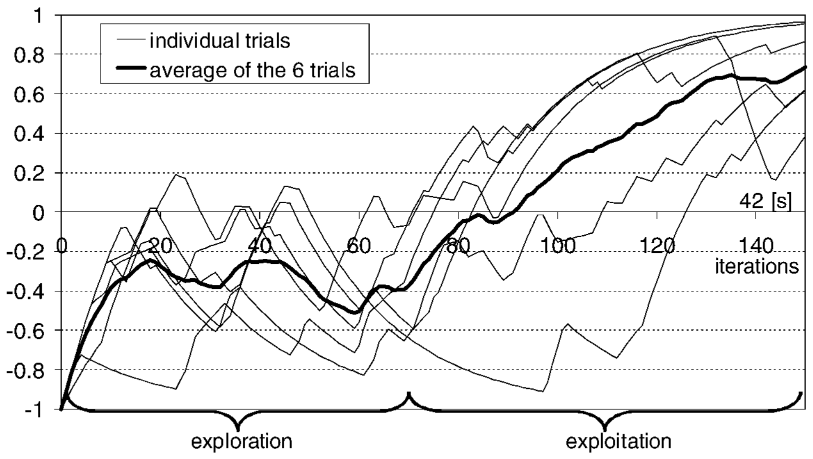


Fig. 17. Behavior convergence for six different learning trials of the Yaw dof.

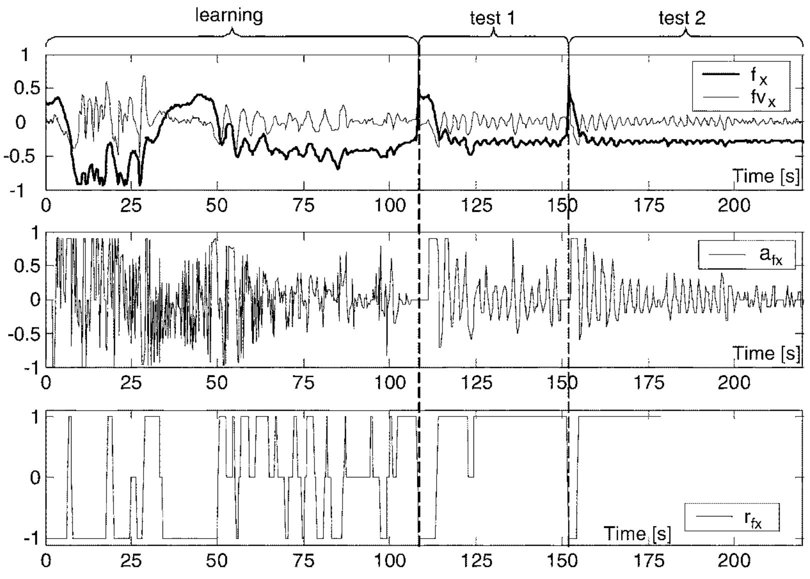


Fig. 18. Online learning evolution and behavior testing of the tracking behavior in the X dof. The states, action, and reward signals are shown.

action, the target position reaches the 0 value where the reward becomes maximum $(r = 1)$. Experimental results show that the presented approach using SONQL is feasible and promising for AUV high-level control and encouraged further investigation with additional behaviors.

B. URIS

The presented approach with four behaviors was tested on the URIS. Experiments were performed in a water tank that allowed only the movement of the vehicle in the horizontal plane. Therefore, the controlled dof were the surge and yaw velocities. The vehicle was also equipped with a forward looking camera (Fig. 15). It was used by the tracking behavior that was learned with 2 SONQL algorithms. The recovery behavior was preprogrammed using the same strategy to recover the target. For the repulsion behavior, a preprogrammed state-action mapping was used, instead of using a SONQL algorithm, to avoid any physical damage to the vehicle during the learning period. For this test, the repulsion behavior used the data of a vision-based positioning system to determine the distance to the walls of the water tank [40]. A new behavior called *teleoperation* was defined to command the vehicle from an external human machine interface (HMI) module. This behavior was used to test the performance of the approach when the vehicle was moved away from the target. Fig. 16 shows URIS' target following task control system with four behaviors.

The parameters of the learning algorithm were initially set to those used with ODIN. Several trials were then carried out to find the best parameters for URIS. The vehicle was able to learn how to move in both dof achieving the target following task. The parameters and configuration used in the tracking behavior as well as the learned state/action mapping for both dof are shown in Table II. It is noted that these parameters are very similar to those used for the ODIN.

Fig. 17 shows six consecutive learning attempts of the Yaw dof. Each curve represents the mean of the last 20 rewards. The minimum and maximum values correspond to the minimum $(r = -1)$ and maximum rewards $(r = 1)$. The figure also shows that the average reward increased as the behavior was learned. The algorithm starts exploring the state to find the maximum reward. Once the whole state is explored, the algorithm exploits the learned state-action mapping and receives the maximum reward.

Figs. 18 and 19 show a typical online learning evolution of the X and Yaw dof, respectively. The first part of each figure

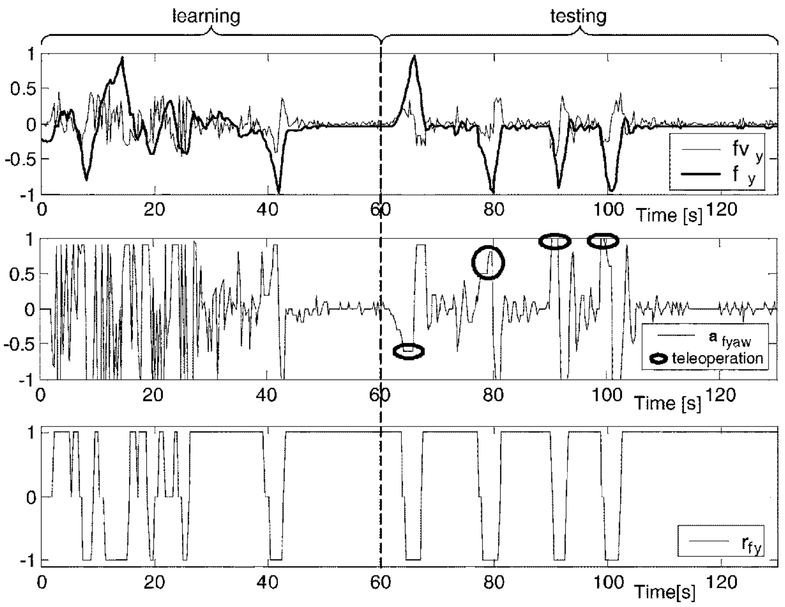


Fig. 19. Online learning evolution and behavior testing of the tracking behavior in the Yaw dof. The states, action, and reward signals are shown.

shows how the vehicle explores the state during the learning period. For the X dof, the learning time was 350 iterations (105 s) and for the Yaw dof was 200 iterations (60 s). This difference is due to the fact that the mapping for the X dof is more nonlinear than the one for the Yaw dof (see Table II). Immediately after the learning period, the vehicle was moved from the target with the teleoperation behavior. It can be seen from the figures how the target is recovered and the maximum rewards are achieved.

When the vehicle gets close to the walls of the water tank, behavior cooperation between the repulsion and tracking behaviors occurs. Whenever the vehicle gets too close to the walls of the water tank, the repulsion behavior fully dominates. To investigate the performance of the hybrid coordination system, the target was placed close to the vehicle so that the vehicle went backward until the wall was found. As shown in Fig. 20, the repulsion behavior became active and stopped the control action of the tracking behavior. The coordinated response in the surge movement was nearly zero, even though the tracking behavior generated a backward movement as shown in the X dof graph. The vehicle stopped between the target and the wall of the tank, as the hybrid coordinator effectively performed.

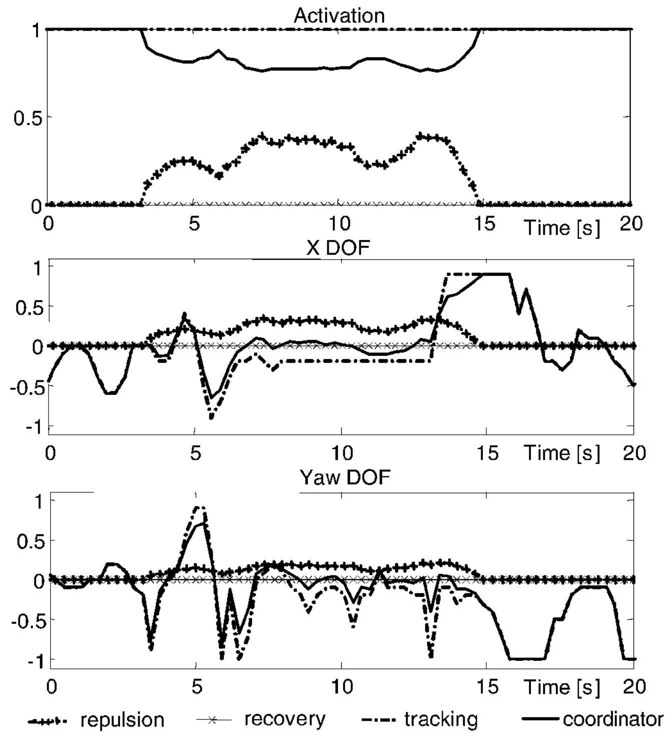


Fig. 20. Performance of the hybrid coordination system.

V. Conclusion

This paper described a new reinforcement learning algorithm, SONQL and a hybrid coordinator for a behavior-based layer of the hybrid control architecture for AUVs. SONQL is based on Q_learning with neural networks to overcome the generation problem and uses a database of learning samples to overcome the interference problem. The hybrid coordination system takes advantage of the competitive approach as well as the cooperative approach.

The presented approach was tested for a target following task. After the preliminary test on the ODIN with two behaviors (tracking and recovery) showing the feasibility of the SONQL algorithm for AUV high-level control, the additional tests on URIS were performed with four behaviors (tracking, repulsion, recovery, and teleoperation). Results show that the presented approach with SONQL algorithm and the hybrid behavior coordinator could help increase the autonomy of AUVs. The future study includes improving the learning capabilities of the SONQL algorithm and investigating effectiveness of the approach for more complex tasks.

REFERENCES

[1] E. Coste-Maniere, H. H. Wang, and A. Peuch, "Control architectures: What's going on?," in *Proc. US-Portugal Workshop on Undersea Robotics and Intelligent Control*, Lisbon, Portugal, 1995, pp. 54–60.

[2] P. Ridao, J. Batlle, J. Amat, and G. N. Roberts, "Recent trends in control architectures for autonomous underwater vehicles," *Int. J. Syst. Sci.*, vol. 30, no. 9, pp. 1033–1056, 1999.

[3] K. P. Valavanis, D. Gracanin, M. Matijasevic, R. Kolluru, and G. A. Demetriou, "Control architectures for autonomous underwater vehicles," *IEEE Contr. Syst. Mag.*, vol. 17, no. 6, pp. 48–64, 1997.

[4] P. Ridao, J. Yuh, K. Sugihara, and J. Batlle, "On AUV control architecture," in *Proc. Int. Conf. Robots and Systems*, Japan, 2000.

[5] J. G. Bellingham and J. J. Leonard, "Task configuration with layered control," in *Proc. IARP 2nd Workshop on Mobile Robots for Subsea Environments*, Monterey, CA, 1994, pp. 193–302.

[6] A. M. Meystel and J. S. Albus, *Intelligent Systems, Architecture, Design and Control*. New York: Wiley, 2002.

[7] R. Arkin, *Behavior-Based Robotics*. Cambridge, MA: MIT Press, 1998.

[8] R. C. Arkin, "Path planning for a vision-based autonomous robot," in *Proc. SPIE Conf. Mobile Robots*, Cambridge, MA, 1986, pp. 240–249.

[9] E. Gat, "Reliable goal-directed reactive control for real-world autonomous mobile robots," Ph.D., Virginia Polytech. State Univ., Blacksburg, 1991.

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

[11] M. Carreras, J. Batlle, and P. Ridao, "Hybrid coordination of reinforcement learning-based behaviors for AUV control," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, HI, 2001.

[12] S. Mahadevan and J. Connell, "Automatic programming of behavior-based robots using reinforcement learning," *Artif. Intell.*, vol. 55, pp. 311–365, 1992.

[13] M. Ryan and M. Pendrith, "RL-TOPs: An architecture for modularity and Re-Use in reinforcement learning," in *15th Int. Conf. Machine Learning*, Madison, WI, 1998.

[14] J. Shackleton and M. Gini, "Measuring the effectiveness of reinforcement learning for behavior-based robotics," *Adapt. Behav.*, vol. 5, no. 3/4, pp. 365–390, 1997.

[15] R. Sutton and A. Barto, *Reinforcement Learning, an Introduction*. Cambridge, MA: MIT Press, 1998.

[16] Y. Takahashi and M. Asada, "Vision-guided behavior acquisition of a mobile robot by multi-layered reinforcement learning," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2000.

[17] C. Touzet, "Neural reinforcement learning for behavior synthesis," *Robot. Autonomous Syst.*, vol. 22, pp. 251–281, 1997.

[18] D. Gachet, M. Salichs, L. Moreno, and J. Pimental, "Learning emergent tasks for an autonomous mobile robot," in *Proc. Int. Conf. Intelligent Robots and Systems*, Munich, Germany, 1994, pp. 290–97.

[19] Z. Kalmar, C. Szepesvari, and A. Lorincz, "Module-based reinforcement learning: Experiments with a real robot," in *Proc. 6th Eur. Workshop on Learning Robots*, 1997.

[20] P. Maes and R. Brooks, "Learning to coordinate behaviors," in *Proc. 8th AAAI*, 1990, pp. 796–802.

[21] E. Martinson, A. Stoytchev, and R. Arkin, "Robot behavioral selection using Q_learning," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Lausanne, Switzerland, 2002.

[22] B. Lee and R. C. Arkin, "Adaptive multi-robot behavior via learning momentum," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2003.

[23] B. Bakker, F. Linaker, and J. Schmidhuber, "Reinforcement learning in partially observable mobile robot domains using unsupervised event extraction," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Lausanne, Switzerland, 2002.

[24] W. T. B. Uther and M. M. Veloso, "Tree based discretization for continuous state space reinforcement learning," in *Proc. 15th National Conf. Artif. Intell.*, 1998.

[25] J. C. Santamaria, R. S. Sutton, and A. Ram, "Experiments with reinforcement learning in problems with continuous state and action spaces," *Adapt. Behav.*, vol. 6, pp. 163.218–163.218, 1998.

[26] W. D. Smart, "Making reinforcement learning work on real robots," Ph.D., Dept. Computer Science, Brown Univ., RI, 2002.

[27] C. Gaskett, "Q_learning for robot control," Ph.D., Australian National Univ., 2002.

[28] C. J. C. H. Watkins and P. Dayan, *Q_Learning Machine Learning*, 1992, vol. 8, pp. 279–292.

[29] K. Baird, "Residual algorithms: Reinforcement learning with function approximation," in *Machine Learning: 12th Int. Conf.*, San Francisco, CA, 1995.

[30] G. J. Tesauro, "Practical issues in temporal difference learning," *Machine Learning*, vol. 8, no. 3/4, pp. 257.277–257.277, 1992.

[31] S. Weaver, L. Baird, and M. Polycarpou, "An analytical framework for local feedforward networks," *IEEE Trans. Neural Netw.*, vol. 9, 1998.

[32] M. Carreras, "An overview of behavior-based robotics with simulated implementations on an underwater vehicle," Univ. Girona, Spain. Informatics and Applications Institute, Res. Rep.: IIiA 00-14-RR, 2000.

[33] R. A. Brooks, "Robust layered control system for a mobile robot," *IEEE J. Robot. Automat.*, vol. RA-2, no. 1, pp. 14–23, 1986.

[34] P. Maes, "Situated agents can have goals," *Robot. Automat. Syst.*, vol. 6, pp. 49–70, 1990.

[35] R. C. Arkin, "Motor schema-based mobile robot navigation," *Int. J. Robotica Res.*, vol. 8, no. 4, pp. 92–112, 1989.

[36] L. Steels, "Building agents with autonomous behavior systems," in *The Artificial Route to Artificial Intelligence. Building Situated Embodied Agents*. New Haven, CT: Lawrence Erlbaum, 1993.

[37] S. K. Choi, J. Yuh, and G. Y. Takashige, "Development of the omni-directional intelligent navigator," *IEEE Robot. Automation Mag.*, pp. 44–53, 1995.

[38] J. Nie, J. Yuh, E. Kardash, and T. I. Fossen, "Onboard sensor-based adaptive control of small UUV's in the very shallow water," in *Proc. IFAC-Control Applications in Marine Systems*, Fukuoka, Japan, 1998, pp. 201–206.

[39] M. Carreras, A. Tiano, A. El-Fakdi, A. Zirilli, and P. Ridao, "On the identification of non linear models of unmanned underwater vehicles," in *1st IFAC Workshop on Guidance and Control of Underwater Vehicles GCUV '03*, Wales, U.K., 2003.

[40] M. Carreras, P. Ridao, R. Garcia, and T. Nicosevici, "Vision-based localization of an underwater robot in a structured environment," in *IEEE Int. Conf. Robotics Automation ICRA'03*, Taipei, Taiwan, 2003.

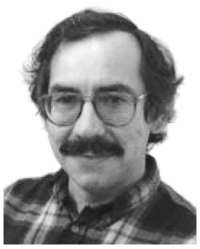
Marc Carreras was born in Spain in 1975. He received the Ms.C. degree in industrial engineering in 1998 and the Ph.D. degree in computer engineering in 2003, both from the University of Girona, Spain.

His research activity is mainly focused on robot learning and control of autonomous underwater vehicles. He joined the Institute of Informatics and Applications, University of Girona, in September 1998, and has been an Associate Lecturer with the Department of Electronics, Informatics and Automation, since October 2002. He is involved in some governmental projects about underwater robots.

Junku Yuh (F'05) received the B.S. degree from Seoul National University, Korea, in 1981 and the M.S. and Ph.D. degrees from Oregon State University in 1983 and 1986, respectively.

He joined the National Science Foundation, Arlington, VA, in 2001 as a Program Director of Robotics Program and Computer Vision Program, Division of Information and Intelligent Systems after 17 years as a Professor of mechanical engineering with the University of Hawaii, where he still directs the Autonomous Systems Laboratory. His main research interests include intelligent navigation and guidance, and underwater robotic vehicle control. He has published more than 120 technical articles and edited/coedited 10 books in the area of robotics, including *Underwater Robots* (Boston, MA: Kluwer, 1996) and *Underwater Robotic Vehicles: Design and Control* (Albuquerque, NM: TSI, 1995).

Dr. Yuh received a 1991 Presidential Young Investigator Award from the National Science Foundation and a 2004 Lifetime Achievement Award from World Automation Congress. He is listed in *Who's Who in the World, and Men of Achievement*. He has served as an associate editor for several journals including the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION and is on the Editorial Board of the *Journal of Autonomous Robots* and the *International Journal of Intelligent Automation & Soft Computing*. He has chaired several conferences including the Program Chair of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). He founded and chairs the Technical Committee on Underwater Robotics of the IEEE Robotics and Automation Society.



Joan Batlle was born in Spain in 1953. He received the Ms.C. degree in physics in 1975 from the Universitat Autònoma de Barcelona, Spain, and the Ph.D. degree in computer engineering from the Technical University of Catalonia, Barcelona.

His research activity is mainly focused on real-time vision and autonomous underwater robots. He is a Professor of computer science with the University of Girona, Spain, teaching courses on computer vision systems, and advanced technologies. He has been the Director of the Institute of Informatics and Applications, a research center with 60 full-time researchers. He has been Head of the Department of Electronics, Informatics and Automation, University of Girona. He has been a Director of many funded research projects mainly concerning real-time computer vision and underwater robotics. He is currently Rector of the University of Girona. He is involved in some governmental projects about underwater robots and technology transference to industrial environments. He has published more than 200 international contributions in journals, workshops, and conferences.

Dr. Batlle is member of the IFAC's Technical Committee on Marine Systems and has been a member of the Program Committee of more than 20 international conferences on computer vision and robotics during the last four years.

Pere Ridao (M'03) was born in Spain in 1969. He received the Ms.C. degree in computer science in 1993 from the Technical University of Catalonia, Barcelona, Spain, and the Ph.D. degree in computer engineering in 2001 from the University of Girona, Spain.

His research activity is mainly focused on underwater robotics in research topics such as control architectures, UUV modeling and identification, simulation, and real-time operating systems. He joined the Institute of Informatics and Applications, University of Girona, in September 1995. At present, he is a Lecturer with the Department of Electronics, Informatics and Automation, University of Girona. He is involved in some governmental projects about underwater robots and some technology transference projects about real-time and embedded systems.

Dr. Ridao is member of the IFAC's Technical Committee on Marine Systems and a member of some Program Committees including the IEEE/RSJ International Conference on Intelligent Robots and Systems Committee.