2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Two steps Natural Actor Critic Learning for Underwater Cable Tracking

Andres El-Fakdi, Marc Carreras and Enric Galceran

Abstract— **This paper proposes a field application of a high-level Reinforcement Learning (RL) control system for solving the action selection problem of an autonomous robot in a cable tracking task. The underwater vehicle $ICTINEU^{AUV}$ learns to perform a visual based cable tracking task in a two step learning process. First, a policy is computed by means of simulation where a hydrodynamic model of the vehicle simulates the cable following task. Once the simulated results are accurate enough, in a second step, the learned-in-simulation policy is transferred to the vehicle where the learning procedure continues in a real environment, improving the initial policy. The natural actor-critic (NAC) algorithm has been selected to solve the problem in both steps. This algorithm aims to take advantage of policy gradient and value function techniques for fast convergence. Actor's policy gradient gives convergence guarantees under function approximation and partial observability while critic's value function reduces variance of the estimates update improving the convergence process.**

I. INTRODUCTION

Reinforcement Learning (RL) is a widely used methodology in robot learning [1]. In RL, an agent tries to maximize a scalar evaluation obtained as a result of its interaction with the environment. The goal of a RL system is to find an optimal policy to map the state of the environment to an action which in turn will maximize the accumulated future rewards.

Over the last decade, the two major classes of reinforcement learning algorithms, value-based methods and policy search methods, have offered different solutions to solve RL problems. Value function methodologies have worked well in many applications, achieving great success with discrete lookup table parameterization but giving few convergence guarantees when dealing with high dimensional domains due to the lack of generalization among continuous variables [1]. Rather than approximating a value function, policy search techniques approximate a policy using an independent function approximator with its own parameters, trying to maximize the future expected reward [2].

Policy search applications share a common drawback, gradient estimators used in these algorithms may have a large variance [3], learning much more slower than RL algorithms using a value function (see [4]) and they can converge to local optima of the expected reward, making them less suitable for on-line learning in real applications. In order to decrease convergence times and avoid local optima, newest applications combine policy gradient search with value function techniques, adding the best features of both methodologies within actor-critic algorithms [3]. In actor-critic algorithms, the critic component maintains a value function, and the actor component maintains a separate parameterized stochastic policy from which the actions are drawn. Actor's policy gradient gives convergence guarantees while critic's value function reduces variance of the policy update improving the convergence rate of the algorithm [3]. It is worth to mention the work done in [5] and [6], where a biped robot is trained to walk by means of a "hybrid" RL algorithm that combines policy search with value function methods.

Recent work suggests that even with the aid of a critic's value function, policy gradients converge quite slow, in part caused by the small gradients around plateau landscapes of the expected return [7]. In order to avoid these situations, studies presented in [8] pointed that the natural policy gradient may be a good alternative to the policy gradient. A natural gradient is the one that looks for the steepest ascent with respect to the Fisher information matrix [8] instead of the steepest direction in the parameter space. Being easier to estimate than regular policy gradients, they are expected to be more efficient and therefore accelerate the convergence process. Natural gradient algorithms have found a variety of applications in the last years, as in [9] with traffic-light system optimization and in [10] with gait optimization for robot locomotion.

All those improvements mentioned before push up gradient techniques, but speeding up gradient methods in real, continuous, high dimensional domains represents a key factor. The idea of providing initial high-level information to the learner, like example policies or human imitation techniques has achieved great success in several applications [11] [12] [13]. Also, to learn an initial policy from a model simulation for, in a second step, transfer it to the real agent can greatly increase converge rates. Rude policies learned in simulation represent a good startup for the real learning process, meaning that the learner will face the challenge of the real world with some initial knowledge of the environment, avoiding initial gradient plateaus and local maxima dead ends [14].

This paper proposes a real reinforcement learning application where the ideas mentioned before are combined together. The underwater vehicle $ICTINEU^{AUV}$ learns to perform a visual based cable tracking task in a two step learning process. First, a policy is computed by means of computer simulation where a hydrodynamic model of the vehicle simulates the cable following task. Once the simulated results are accurate enough, in a second step, the learned-

A. El-Fakdi, Marc Carreras and Enric Galceran are with the Computer Vision and Robotics Group, Institute of Informatics and Applications, University of Girona, 17071 Girona, Spain. aelfakdi@eia.udg.edu

in-simulation policy is transferred to the vehicle where the learning procedure continues on-line in a real environment, improving the initial policy. The natural actor-critic (NAC) algorithm has been selected to solve the problem through online interaction with the environment in both steps, the simulation and the real learning. This paper is structured as follows. In Section II the learning procedure and the NAC algorithm are detailed. Section III describes the setup: the mathematical model of $ICTINEU^{AUV}$ used in the simulation and the vision system. Details and results of the simulation process and the real test are given in Section IV and finally, conclusions and the future work to be done are included in Section V.

II. TWO STEPS NATURAL ACTOR CRITIC LEARNING

Two steps learning takes advantage of learning by simulation as an initial startup for the real learner. Next subsections give details about the natural actor-critic algorithm used and the procedures regarding the two steps learning proposal.

A. The Natural Actor-Critic algorithm

In natural actor-critic (NAC) [13], stochastic natural policy gradients allow actor updates while the critic computes simultaneously the natural gradient and the value function parameters by linear regression. As stated in previous section, the NAC algorithm aims to take advantage of policy gradient and value function techniques for fast convergence. Actor's policy gradient gives convergence guarantees under function approximation and partial observability while critic's value function reduces variance of the estimates update improving the convergence process. Also, natural gradients add curvature statistics into the gradient ascent. The algorithm's procedure is summarized in Algorithm 1.

As initial inputs we have a parameterized, derivable policy $\pi(u|x) = p(u|x,\theta)$ for actor's representation together with a basis function $\phi(x)$ for the critic's value function. At every iteration, action u_t is drawn from current policy π_t generating a new state x_{t+1} and a reward r_t. After updating the basis function and the statistics by means of LSTD(λ) we obtain the value function parameters v and the natural gradient w. Actor's policy parameters are updated only if the angle between two consecutive natural gradients is small enough compared to an ϵ term. The learning rate of the update is controlled by the α parameter. Next, the critic has to forget part of its accumulated statistics using a forgetting factor $\beta \in [0,1]$. Current policy is directly modified by the new parameters becoming a new policy to be followed next iteration, getting closer to a final policy that represents a correct solution of the problem.

Algorithm 1: Natural Actor-Critic algorithm with LSTD-Q(λ)

Initialize:
Parameterized policy $\pi_s = \pi(u|x)$ with initial parameters $\theta = \theta_0$, its derivative $\nabla_\theta \log \pi_s = \pi(u|x)$ and basis function $\phi(x)$ for the value function parameterization $V^\pi(x)$.
Initialize $z = A = b = 0$.
Draw initial state x_0.
for $t = 0$ to n do:
  Generate control action u_t according to current policy π_t. Observe new state x_{t+1} and the reward obtained r_t.
  Critic Evaluation (LSTD-Q)
    Update basis functions
    $\tilde{\phi}_t = [\phi(x_{t+1})^T, 0^T]$
    $\hat{\phi}_t = [\phi(x_t)^T, \nabla_\theta \log \pi(u_t|x_t)^T]^T$
    Update sufficient statistics:
    $z_{t+1} = \lambda z_t + \hat{\phi}_t$
    $A_{t+1} = A_t + z_{t+1}(\hat{\phi}_t - \gamma\tilde{\phi}_t)^T$
    $b_{t+1} = b_t + z_{t+1} r_t$
    Update critic parameters:
    $[v_{t+1}^T, w_{t+1}^T] = A_{t+1}^{-1} b_{t+1}$
  Actor Update
    If $\angle(w_{t+1}, w_{t-\tau}) \leq \epsilon$, then update policy parameters:
    $\theta_{t+1} = \theta_t + \alpha w_{t+1}$
    Forget sufficient statistics:
    $z_{t+1} = \beta z_{t+1}$
    $A_{t+1} = \beta A_{t+1}$
    $b_{t+1} = \beta b_{t+1}$
end

B. Two steps learning

The proposal here presented aims to reduce the time spent in $ICTINEU^{AUV}$'s real learning iterations when dealing with a visual cable tracking task. Previous work done in [15] demonstrated slow convergence of policy gradient techniques when applied to real large state spaces as the one we are focusing. In [15], gradient techniques allowed the vehicle to learn a fairly good policy in simulation but, once it was transferred to the real vehicle, almost any policy improvement was appreciated along the different tests. Results published by scientific community about successful practical applications of the NAC algorithm pushed us to implement it. As can be seen in Fig. 1, the two step learning procedure has a initial phase of learning in simulation, Fig. 1(a), where the NAC algorithm stated in Algorithm 1 is trained in our MATLAB simulator. The model of our underwater vehicle has been identified as shown in Section III-A, and allows us to emulate a robot with three degrees of freedom (DOF): X movement (surge), Y movement (sway) and rotation respect Z axis (yaw). Z movement (heave) has not been modeled because the attitude of the vehicle with respect to the bottom of the pool will remain constant during the tests, although it can be modified as an input parameter. Dimensions of the pool and cable thickness can be modified to match the real ones. Also the model of the camera used has been introduced in the simulator to offer the same field of view than the real camera. The sampling time of the simulator is $0.2secs$, equal to the one offered by our vehicle. Actions, forces, accelerations and velocities are ranged to match the maximum ones $ICTINEU^{AUV}$ is able to achieve for each DOF. The simulated experiments begin by setting parameterized policy $\pi_s = \pi(u|x)$ with initial parameters $\theta = \theta_0$ and basis functions $\phi(x)$ for the value function parameterization $V^\pi(x)$. Also, learning parameters such as learning rate(α), forgetting factor(β), discounted reward factor(γ), eligibility rate(λ) and the actor update condition(ϵ) are set and stored in order to be transferred to the vehicle in step two. Details concerning function designs and parameter values will be

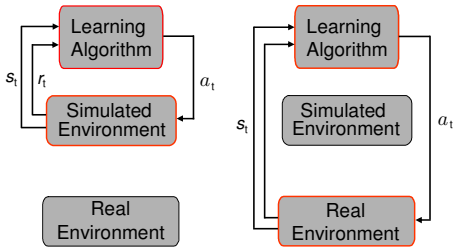

Fig. 1. Learning phases.

detailed in Section III-C.

Once it is considered that the algorithm has acquired enough knowledge from the simulation to build a "secure" policy, step two begins, Fig. 1(b). Actor and critic parameters, together with learnt values are transferred to the real vehicle to continue the learning process. In this phase, the algorithm is executed in the software architecture of the robot, a distributed object oriented architecture. The algorithm has been programmed in C++. As the computed model and the simulated environment are not accurate enough, the algorithm will have to adapt to this new situation in order to improve expected rewards. Once the algorithm converges in the real environment, the learning system will continue working forever, being able to adapt to any future change.

III. THE CABLE TRACKING TASK

This section is going to describe the different elements that take place into our problem: first, a brief description of the underwater robot $ICTINEU^{AUV}$'s model used in simulation is given. The section will also present the problem of underwater cable tracking and, finally, a description of the NAC configuration for the particular cable tracking task.

A. AUV mathematical model

As described in [16], the non-linear hydrodynamic equation of motion of an underwater vehicle can be conveniently expressed in the body fixed frame $\{B\}$ as:

$$\tau^B + G(\eta) - D(\upsilon^B)\upsilon^B + \tau_p = (M^B_{RB} + M_A)\dot{\upsilon}^B + ... \\ ... + (C^B_{RB}(\upsilon^B) + C_A(\upsilon^B))\upsilon^B \quad (1)$$

where υ^B and $\dot{\upsilon}^B$ are the velocity and acceleration vectors, η is the position and attitude vector, τ^B is the resultant force-torque vector exerted by thrusters, $G(\eta)$ is the gravity-buoyancy force-torque vector, $D(\upsilon^B)$ corresponds to the linear and quadratic damping, M^B_{RB} and M^B_A correspond to the mass-inertia and added mass mass-inertia matrixes respectively, C^B_{RB} and C^B_A represent the Coriolis and centripetal matrixes for the rigid body and the added mass effect and τ^B_p is a unmodelled perturbation. Eq. 1 represents a complex non-linear model with couplings among the different degrees of freedom being dependent to a large set of physical parameters (robot mass and inertia, the linear and nonlinear friction coefficients, thrust coefficients, etc...). Nevertheless, after some simplifications it is possible to carry out identification experiments to identify the most relevant coefficients to obtain an approximate model. Parameters corresponding to real AUVs have been reported for instance in [16]. In our work, we have identified the $ICTINEU^{AUV}$ [17] parameters using a method previously developed in our lab [18] which has been satisfactorily applied in the past to other vehicles. Table I reports the robot parameters used in the context of this work.

TABLE I
PARAMETER IDENTIFICATION RESULTS.

	Parameters			
DOF	α	β	γ	δ
Surge (X movement)	0.1347	0	0.4458	0.0040
Sway (Y movement)	0.1508	0	0.2586	-0.0004
Yaw (Z rotation)	0.2878	0	3.4628	-0.0026

B. The Cable Tracking Vision System

The vision-based algorithm used to locate the cable was formerly proposed in [19]. It uses a downward-looking B/W camera to first locate and then track the cable. The algorithm computes the polar coordinates $(\rho,\ \Theta)$ of the straight line corresponding to the detected cable in the image plane (see Figure 2). Being $(\rho,\ \Theta)$ the parameters of the cable line, the cartesian coordinates (x,y) of any point along the line must satisfy Equation (2).

$$\rho = xcos(\Theta) + ysin(\Theta) \quad (2)$$

As shown in Figure 2b, Equation (2) allows us to obtain the coordinates of the cable intersections with the image boundaries (X_u,Y_u) and (X_L,Y_L), thus the mid point of the straight line (x_g,y_g) can be easily computed $(x_g, y_g = \frac{X_L+X_u}{2}, \frac{Y_u+Y_L}{2})$. The computed parameters (ρ, Θ, x_g, y_g) together with its derivatives are sent to the control module in order to be used by the different behaviors.

C. The NAC algorithm for underwater cable tracking

As stated in previous section, the observed state is a 4 dimensional state vector $x = (x_g, \Theta, \frac{\delta x_g}{\delta t}, \frac{\delta \Theta}{\delta t})$ where $x_g \in [0,1]$ is the normalized x coordinate of the cable centroid in the image plane, $\Theta \in [\frac{-\pi}{2}, \frac{\pi}{2}]$ is the cable angle, being

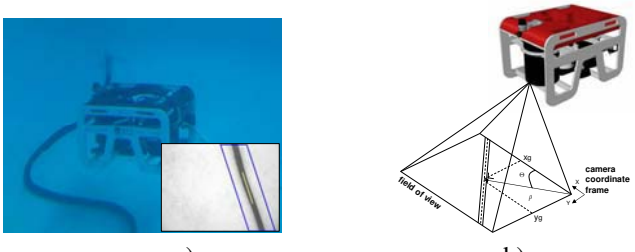

Fig. 2. (a) $ICTINEU^{AUV}$ in the test pool. Small bottom-right image: Detected cable. (b) Coordinates of the target cable with respect $ICTINEU^{AUV}$.

$\frac{\delta x_g}{\delta t} \in [-0.5, 0.5]$ and $\frac{\delta \Theta}{\delta t} \in [-1, 1] rad/s$ the respective derivative . Based on these observations, the robot decides a normalized continuous action vector $u = (u_{sway}, u_{Yaw}) \in [-1, 1]$ to guide the robot in the Sway and Yaw DOFs. The Surge motion is not learnt. Instead, a simple controller is used to move forward the robot at a constant Surge speed of $u_{Surge} = 0.3\ m/s$ when the cable is centered in the image plane, being zero otherwise.

It is worth nothing that u_{sway} actions are mainly affected by the position and velocity of x_g along the X axis of the image plane. In the same way, u_{Yaw} actions are strongly dependent on the variations of Θ. Therefore, the state vector has been split into two subspaces, $x_{sway} = (x_g, \frac{\delta x_g}{\delta t})$ and $x_{Yaw} = (\Theta, \frac{\delta \Theta}{\delta t})$, feeding two independent policies, being the NAC algorithm used for each policy the one detailed in Algorithm 1.

Actor policies for Yaw and sway are described as normal gaussian distributions of the form $\pi(u|x) = N(u|Kx, \sigma^2)$ where the standard deviation has been fixed to $\sigma = 0.1$. Basis function for Sway and Yaw value functions parameterization are chosen as $\phi(x) = [x_1^2, x_1 x_2, x_2^2, 1]$. Rewards are given by $r(x_t, u_t) = x_t^T Q x_t + u_t^T R u_t$ with $Q_{sway} = diag(1.1, 0.2)$ and $R_{sway} = 0.01$ for $sway$ DOF and $Q_{Yaw} = diag(2, 0.25)$ and $R_{Yaw} = 0.01$ for Yaw DOF. Learning rate, forgetting factor, discounted reward factor, eligibility rate and the actor update condition of both algorithms has been fixed respectively to $\alpha = 0.01$, $\beta = 0.9$, $\gamma = 0.95$, $\lambda = 0.8$ and $\epsilon = \pi/180$.

IV. RESULTS

Identical trajectories have been carried out with the simulator and the real vehicle in order to test the simulator's performance. Both, the simulated vehicle the real one have been situated in the same coordinates of the pool with the cable lying in the middle. Vehicles receive the same input to their motors and as the robots reach the center of the pool, the cable appears within the image plane. Fig. 3 and Fig. 4 show a comparison between the simulated and real values of the state variables ρ and θ. These results demonstrate a similar state transition for the simulator and the real vehicle for a given test.

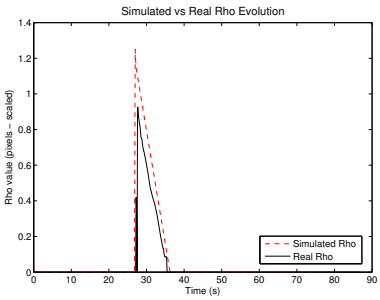

Fig. 3. For the same input, comparison between the real and the simulated rho evolution of the cable in the vehicle's image plane.

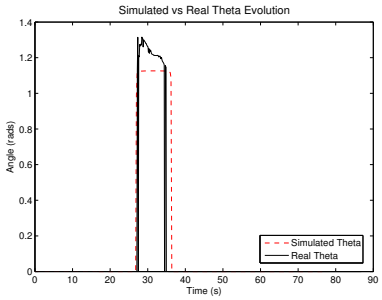

Fig. 4. For the same input, comparison between the real and the simulated theta evolution of the cable in the vehicle's image plane.

A. Step one results: Simulated Learning

The model of the underwater robot $ICTINEU^{AUV}$ navigates a two dimensional world at 1 meter height above the seafloor. The simulated cable is placed at the bottom in a fixed position. The learner has been trained in an episodic task. An episode ends either every 20 seconds (200 iterations) or when the robot misses the cable in the image plane, whatever comes first. When the episode ends, the robot position is reset to a random position and orientation around the cable's location, assuring any location of the cable within the image plane at the beginning of each episode. According to the values of the state parameters θ and x_g and actions taken u_{sway} and u_{yaw}, a scalar immediate reward is given at each iteration step. The number of episodes in simulation has been set to 100. For every episode, the total amount of reward perceived is calculated. Fig. 5 represents a comparison of the learning curves for Yaw policy between Baxter and Bartlett's algorithm used in [15] with the new results obtained with the natural actor-critic algorithm. The experiment has been repeated in 100 independent runs. The results here presented are the mean over these runs. In Fig. 6 and Fig. 7 we can observe a state/action mapping of a trained-in-simulation policy in both, yaw and sway degrees of freedom. As can be appreciated in both figures, the horizontal alignment of simulated policies mean that the trained algorithms give much more importance to the states θ and x_g rather than its derivatives $\frac{\delta x_g}{\delta t}$) and $\frac{\delta \Theta}{\delta t}$). The reason for that lies in the low speed assumption done in order to identify the model of our vehicle together with the elimination of the quadratic damping term.

B. Step two results: Real Learning

Once the learning process is considered to be finished, resultant policies with its correspondent parameters are transferred to $ICTINEU^{AUV}$. In Fig. 8 and Fig. 9 the behaviors of the learned-in-simulation policies are initially evaluated in the real vehicle. As can be seen, the results are quite good, and, although having some oscillations, the vehicle is able to follow the cable.

As stated before, once both policies are transferred to the robot, the real learning starts. Fig. 10 and Fig. 11 show the variation of the state variables θ and x_g along different learning episodes. The first trial represents the behavior of

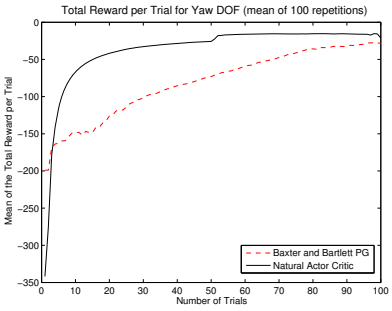

Fig. 5. Learning curves for Yaw policy comparing the natural actor-critic to Baxter and Bartlett's policy gradient algorithm. Performance estimates were generated by simulating 100 episodes. Process repeated in 100 independent runs. The results are a mean of these runs.

the vehicle after 90secs. As the learning process advances, the robot behavior improves significantly. After 20 trials (around 1800secs) we can assure that the learned policy has overcome the learned-in-simulation one.

Final policies can be seen in Fig. 12 and Fig. 13. If both policies are compared with the ones we initially obtained with the simulator (Fig. 6 and Fig. 7), two significant details can be observed. Vertical alignment of both policies show us the importance of the state derivatives. It means that the final learned policy has adapted to the lack of knowledge and limitations of our identified model in order to achieve satisfactory results with the real vehicle. Also, final policies are hard demandant on thruster requirements, as can be appreciated by the large areas in the state/action mapping where both policy outputs are -1 or 1, differing from initial simulated policies, where the response is softly distributed over all state space.

V. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

The underwater vehicle $ICTINEU^{AUV}$ has learnt to perform a real visual based cable tracking task in a two step learning process. First, a policy has been computed by means of simulation where a hydrodynamic model of the vehicle simulates the cable following task. Once the simulated results were good enough, in a second step, the learned-in-simulation policy has been transferred to the vehicle where the learning procedure continued on-line in a real environment. The natural actor-critic (NAC) algorithm

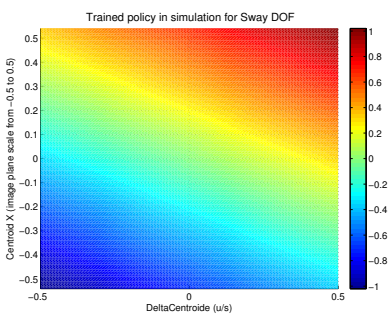

Fig. 6. Representation of sway policy after simulation learning is complete.

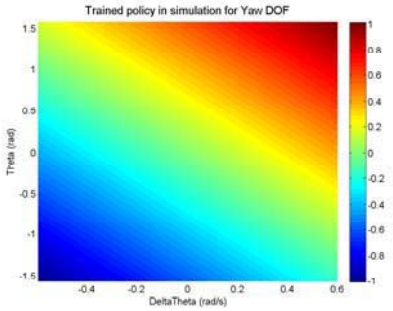

Fig. 7. Representation of yaw policy after simulation learning is complete.

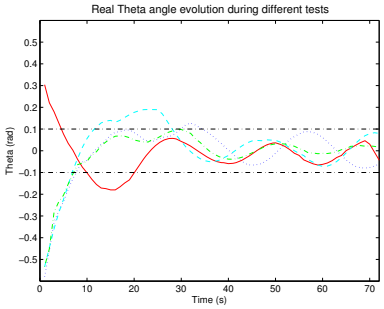

Fig. 8. Performance of the simulated policies on the real vehicle. Evolution of theta angle during several attempts to center the cable.

was selected to solve the problem through online interaction with the environment in both steps, the simulation and the real learning. Results show a good performance of the algorithm, specially compared with the slow gradient techniques used in previous work. Learning with and underwater vehicle is a challenging task. High nonlinear models, external perturbations and technical issues related to noisy sensors, navigation and delays make it difficult to achieve good results. Speed and performance demonstrated by the NAC algorithm encourages us to learn other tasks using this kind of algorithms.

B. Future Works

Our work is currently focused on testing the performance of the NAC algorithm with the cable tracking task. Future experiments point towards the variation of external elements, once the final policy has been learned, such as accidental

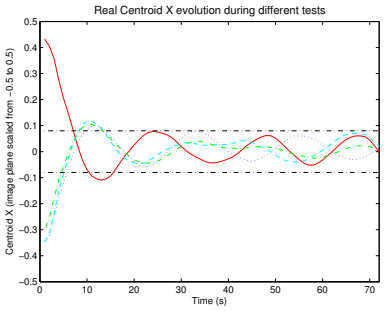

Fig. 9. Performance of the simulated policies on the real vehicle. Evolution of centroid x position during several attempts to center the cable.

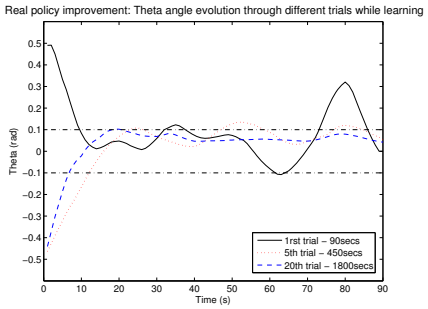

Fig. 10. Policy improvement through different trials of real learning. Evolution of theta angle during several episodes.

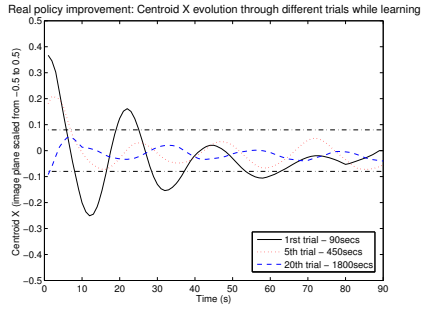

Fig. 11. Policy improvement through different trials of real learning. Evolution of centroid x position during several episodes.

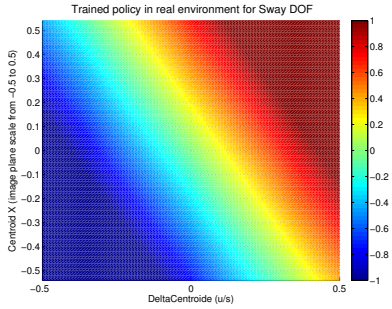

Fig. 12. Final trained policy for sway DOF.

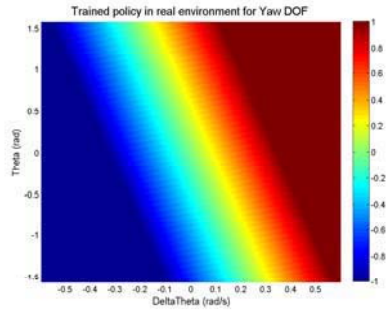

Fig. 13. Final trained policy for yaw DOF.

camera displacements or rotations, motor failures and the introduction of constant water currents. As pointed in previous section, second step of learning will never finish, so it will remain active on the vehicle, readapting it if necessary without any need of external aid.

VI. ACKNOWLEDGMENTS

We would like to give our special thanks to the group of the University of the Balearic Islands for allowing us to use their cable detection algorithm.

REFERENCES

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

[2] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS'06*, Beijing, China, October 9-15 2006.

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

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

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

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

[7] J. Peters, "Machine learning of motor skills for robotics," Ph.D. dissertation, Department of Computer Science, University of Southern California., 2007.

[8] S. Amari, "Natural gradient works efficiently in learning," *Neural Computation*, vol. 10, pp. 251–276, 1998.

[9] S. Richter, D. Aberdeen, and J. Yu, "Natural actor-critic for road traffic optimisation," in *Neural Information Processing Systems, NIPS'06*, 2006, pp. 1169–1176.

[10] T. Ueno, Y. Nakamura, T. Shibata, K. Hosoda, and S. Ishii, "Stable learning of quasi-passive dynamic walking by an unstable biped robot based on off-policy natural actor-critic," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[11] W. Smart, "Making reinforcement learning work on real robots," Ph.D. dissertation, Department of Computer Science at Brown University, Rhode Island, May 2002.

[12] B. Hammer, S. Singh, and S. Scherer, "Learning obstacle avoidance parameters from operator behavior," *Journal of Field Robotics, Special Issue on Machine Learning Based Robotics in Unstructured Environments*, vol. 23 (11/12), December 2006.

[13] J. Peters, S. Vijayakumar, and S. Schaal, "Natural actor-critic," in *ECML*, 2005, pp. 280–291.

[14] L. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching." *Machine Learning*, vol. 8(3/4), pp. 293–321, 1992.

[15] A. El-Fakdi and M. Carreras, "Policy gradient based reinforcement learning for real autonomous underwater cable tracking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008.

[16] T. I. Fossen, *Guidance and Control of Ocean Vehicles*. John Wiley and Sons, 1995.

[17] D. Ribas, N. Palomeras, P. Ridao, M. Carreras, and E. Hernandez, "Ictineu auv wins the first sauc-e competition," in *IEEE International Conference on Robotics and Automation*, 2007.

[18] P. Ridao, A. Tiano, A. El-Fakdi, M. Carreras, and A. Zirilli, "On the identification of non-linear models of unmanned underwater vehicles," *Control Engineering Practice*, vol. 12, pp. 1483–1499, 2004.

[19] J. Antich and A. Ortiz, "Underwater cable tracking by visual feedback," in *First Iberian Conference on Pattern recognition and Image Analysis (IbPRIA, LNCS 2652)*, Port d'Andratx, Spain, 2003.