Covariance Analysis as a Measure of Policy Robustness in Reinforcement Learning

Nawid Jamali, Petar Kormushev, Seyed Reza Ahmadzadeh, and Darwin G. Caldwell

***Abstract*—In this paper we propose covariance analysis as a metric for reinforcement learning to improve the robustness of a learned policy. The local optima found during the exploration are analyzed in terms of the total cumulative reward and the local behavior of the system in the neighborhood of the optima. The analysis is performed in the solution space to select a policy that exhibits robustness in uncertain and noisy environments. We demonstrate the utility of the method using our previously developed system where an autonomous underwater vehicle (AUV) has to recover from a thruster failure. When a failure is detected the recovery system is invoked, which uses simulations to learn a new controller that utilizes the remaining functioning thrusters to achieve the goal of the AUV, that is, to reach a target position. In this paper, we use covariance analysis to examine the performance of the top, n, policies output by the previous algorithm. We propose a scoring metric that uses the output of the covariance analysis, the time it takes the AUV to reach the target position and the distance between the target position and the AUV's final position. The top polices are simulated in a noisy environment and evaluated using the proposed scoring metric to analyze the effect of noise on their performance. The policy that exhibits more tolerance to noise is selected. We show experimental results where covariance analysis successfully selects a more robust policy that was ranked lower by the original algorithm.**

I. Introduction

Most search algorithms are designed to locate a single optimal policy, which does not match the way humans learn skills [1]. Search procedures in reinforcement learning (RL) rely on expectation-maximization procedure to iteratively update a policy. The policies produced by RL algorithms maximize the long term reward, that is, the cumulative reward of a policy, but it does not consider the behavior of the system in the presence of noise in the environment. Analyzing how sensitive a policy is to noise will allow the RL algorithm to find solutions that exhibit robustness in uncertain and noisy environments.

Learning policies that produce an accurate and robust skill is a challenging task. It is possible to learn a policy that is precise, yet at the same time a small noise in the input parameters can cause a catastrophic failure. Often, in the real world the parameter space is noisy. Hence, a method that helps in selecting a policy that finds a comprise between accuracy and robustness can be useful. In this paper we propose covariance analysis as a metric to asses the robustness of a policy. The technique is very generic and would be applicable also to a wide range of other domains, provided that there are multiple possible policies in the solution space and a covariance matrix can be computed for them.

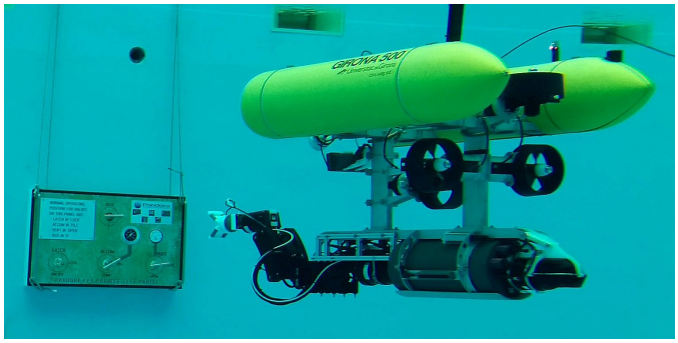


Fig. 1. The Girona 500 AUV.

As part of the PANDORA project [2], [3] we use recovery from a failure of a thruster in an autonomous underwater vehicle (AUV) to demonstrate the utility of the proposed method. AUVs need to operate in harsh underwater environments where recovery from a system failure is critical in the recovery and return of the AUV back to the base. We extend our previous work [4], in which, at the point of failure the robot uses RL to find a policy that will achieve the goal of the robot using its current assets. This allows the AUV to operate persistently. When a failure is identified, the robot uses simulations to learn a policy that uses the remaining thrusters to achieve its goal. In this paper, we modify the method by adding an extra step. Once the RL algorithm terminates, we select the top, n, policies and run them in simulation, in a noisy environment and perform covariance analysis. We also propose a score metric that uses the output of the covariance analysis to evaluate the performance of a policy. The scoring metric combines the time it takes to reach the goal, the distance from the goal and the time it takes to reach the goal. The policy with the highest score is then selected as the candidate policy to be executed by the AUV.

To test the method we used the model of the Girona 500 [5] AUV in an underwater simulator[1] (UWSim). Fig. 1 shows the Girona 500 AUV. It is a reconfigurable AUV, we used a layout that has 5 thrusters: two vertical thrusters for the heave, one lateral for the sway, and two horizontal for the yaw and surge.

This research was sponsored by the PANDORA EU FP7-Project under the Grant agreement No. ICT-288273.

The authors are with the Department of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego 30, 16163 Genoa, Italy. E-mail: {nawid.jamali, petar.kormushev, reza.ahmadzadeh, darwin.caldwell}@iit.it

[1]UWSim – an UnderWater Simulator for marine robotics research and development, http://www.irs.uji.es/uwsim/.

II. Background

Robotic systems are of high-dimensionality, having many degrees of freedom, continuous states and actions, and high noise. Consequently, traditional reinforcement learning (RL) approaches such as Markov decision process and partially observable Markov decision process do not scale up to work in robotics. They suffer severely from the curse of dimensionality. Some success was achieved with the function approximation techniques, however, it was the development of policy-search methods that allowed successful application of RL to robotics problems.

In policy-search, instead of working in the larger state/action spaces, a smaller policy space is used. Thus, reducing the dimensionality and increasing the convergence speed. Well established approaches for implementing policy-search RL [6] include policy-gradient algorithms such as episodic natural actor-critic eNAC [7]. However, these algorithms are sensitive to the learning rate and the exploratory variance parameters which must be provided by the user. An alternative approach based on expectation-maximization method has been proposed [8]. A major advantage of this approach over policy-gradient approach is that it does not require a learning rate parameter.

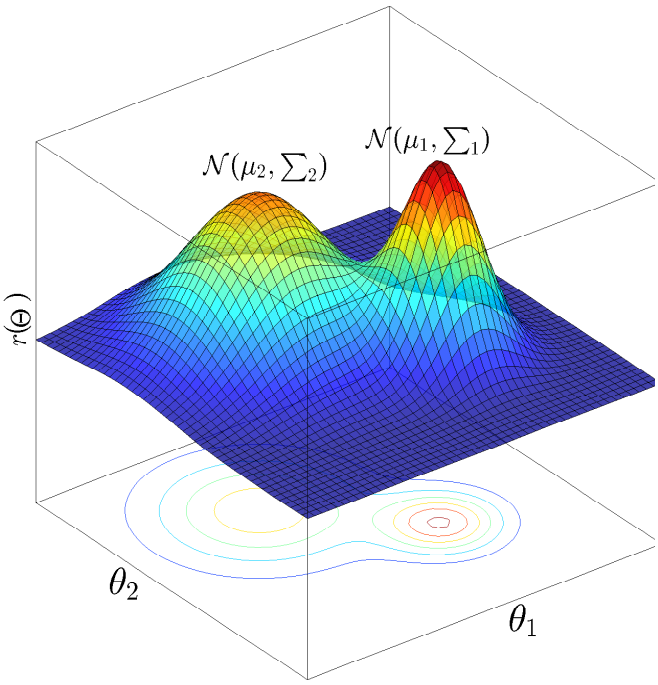
Several search algorithms from the field of stochastic optimization have recently been successfully used for iterative policy improvement. Examples of such approaches are the cross-entropy method [9] and the covariance matrix adaptation evolution strategy [10]. Although these algorithms come from a different domain and are not well-established in RL research, they seem to be a viable alternative for direct policy search RL, as some recent findings suggest [11].

Only having a good policy-search RL algorithm is not enough for solving robotic problems. Before any given RL algorithm can be applied to learn a task on a robot, an appropriate policy representation (also policy encoding) needs to be devised. The choice of policy representation determines what can be learned. In addition, the policy representation can have significant influence on the RL algorithm itself, that is, it can help or impede the convergence or influence the variance of the generated policies. Kormushev et al. [12] analyze in detail the challenges for creating a good policy representation.

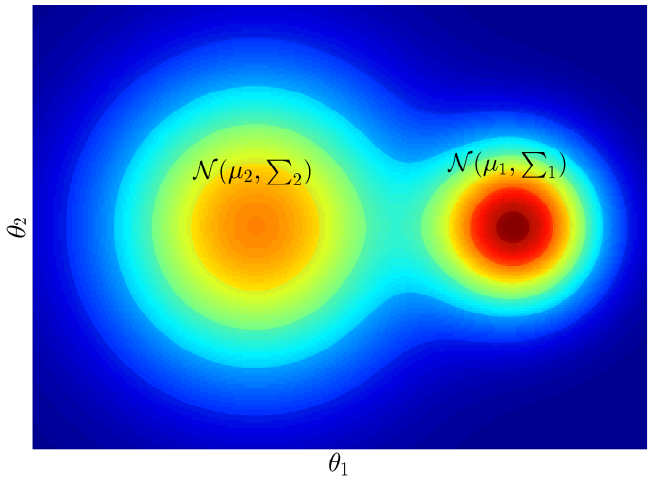
Most search algorithms discussed find one single optimal point. However, these methods do not consider the robustness of the policy to noises in the parameters.

III. Theoretical Concept

Covariance information can serve several purposes. First, it can guide the exploration by defining an adaptive exploration-exploitation trade-off. Second, it conveys important information about the neighborhood of the policy solutions, e.g., shape, total surface, principal directions, and curvature. In some tasks, the immediate neighborhood of the solution manifold has different curvature for different local optima, making some regions more tolerant to errors than others. Fig. 2 illustrates a parameter space with two local optima. A standard gradient-ascent optimization process would converge to one of the two peaks. The peak with the distribution $\mathcal{N}(\mu_1, \sum_1)$ has a higher reward value, however, compared to the peak with the distribution $\mathcal{N}(\mu_2, \sum_2)$, a small change the parameter space can reduce the reward value significantly.



(a) Side view: A plot of the surface and the contours of a bimodal reward function.



(b) Top view: contours of the reward function.

Fig. 2. An illustration of a solution space with multiple local optima, where θ_1 and θ_2 are the parameters of the policy and $r(\Theta)$ is the reward function. A standard gradient-ascent process would converge to one of the two peaks. The peak with the distribution $\mathcal{N}(\mu_1, \sum_1)$ has a higher reward value, however, compared to the peak with the distribution $\mathcal{N}(\mu_2, \sum_2)$, a small change the parameter space can reduce the reward value significantly.

Therefore, the robustness of a policy cannot be determined based on the reward information only, but it is the spread, that is, the covariance of the policy output when evaluated in the presence of noise that determines the best solution that one can reach.

IV. Covariance Analysis Applied to On-line Recovery from a Thruster Failure

The proposed method is tested using our previously developed system [4], where we use reinforcement learning (RL) to recover from a thruster failure in an AUV. In this section we give a brief description of the system and explain how covariance analysis is applied.

A. Problem Description

A failure in the thrusters of an AUV can have catastrophic consequences such as loss of the AUV in deep waters. We use RL to perform a model based policy search that helps the robot to reach a target position. When a thruster failure is detected and isolated by the system, the on-line failure recovery system is invoked. To recover from the failure, the controller system switches from the normal controller to the fault-tolerant controller. The robot uses simulations to learn a policy that produces thruster commands. These simulations use a dynamics model of the system to discover an optimal policy to overcome the thruster failure. In our previous work [4] we investigated three different policy representations, which include constant policy, time-dependent policy, and state-dependent policy. For the problem at hand, we found that time-dependent policy is more flexible than the constant policy representation. Also, compared to state-dependent policy representation it has less optimization parameters. Hence, it is a good compromise between flexibility and computational cost. Consequently, in this paper we use the time-dependent policy representation.

B. Methodology

We frame the problem as a model-based direct policy search reinforcement learning [13]. In this framework the problem is represented by a dynamic model of the vehicle, a policy representation, a cost function, and an optimization algorithm.

1) Dynamic Model: The model of the AUV is represented as a rigid body subject to external forces and torques. The 6 degree-of-freedom equations of the AUV are given by:

$$\begin{gathered} M\dot{\mathbf{v}} + C(\mathbf{v})\,\mathbf{v} + D(\mathbf{v})\,\mathbf{v} + \mathbf{g}(\eta) = \tau \\ \dot{\eta} = J(\eta)\,\mathbf{v} \\ \tau = Bu \end{gathered} \tag{1}$$

where M is the mass matrix; C is the Coriolis matrix; D, is the drag matrix; $g(n)$ is the hydrostatic restoring force vector; $J(\eta)$ is the Jacobian matrix transforming the velocities from the body-fixed to the earth-fixed frame; $\eta = [x\ y\ z\ \phi\ \theta\ \psi]^T$ is the pose (position and orientation) vector; $\mathbf{v} = [u\ v\ w\ p\ q\ r]^T$ is the body velocity vector; τ is the force/torque vector; u is the input vector and B is the thruster reconfiguration matrix. The hydrodynamics parameters of the AUV are identified using an on-line identification algorithm [14], [15].

2) Policy representation: The policy is represented with a linear function approximator that depends only on time, t:

$$u(\mathbf{t}) = \pi(\mathbf{t}|\theta) = \theta^{\mathbf{T}}\phi(\mathbf{t})$$

where u is the control input in Equation 1, the functions $\phi_i(\mathbf{t})$ are called basis functions, or features. We used a third order Fourier basis function. There are eight optimization parameters for each thruster. In the scenario considered, there are two thrusters that can be controlled to reach the target position. Hence, we have sixteen parameters in total.

3) Cost function: The cost function is represented by:

$$J(\theta) = \sum_{t=0}^{T} c_t(\eta_t)\Big|_{\pi(\mathbf{t}|\theta)}$$

where c_t is the immediate cost, and depends on the current state, which in turn is determined by the policy and its parameters. The aim of the agent is to tune the policy's parameters in order to minimize the cumulative cost J over a horizon T.

We use the following definition of the immediate cost:

$$c_t(\langle p_t, v_t\rangle) = \begin{cases} \| p_t - p_d \| & \text{if } t < T \\ w \| v_t - v_d \| & \text{if } t = T \end{cases} \tag{2}$$

where the state $\chi_t = \langle p_t, v_t\rangle$ is composed by position and velocity at time t, p_d is the desired location, v_d is the desired velocity and w weighs the velocity objective with respect to the positional one. The secondary objective is considered only at the final state ($t = T$).

4) Optimization algorithm: We use a derivative-free optimization algorithm because in this case in policy gradient approaches the estimation of the derivatives is computationally expensive. Furthermore, in our previous work [16] we studied three different derivative-free algorithms. The results show that the differential evolution [17] needs fewer number of function evaluations and can reach better objective values. Hence, we use the differential evolution algorithm for optimization.

5) On-line policy search: When a thruster fault is detected, a function J is created to represent the cost of a path to the target location. The optimization algorithm is then instantiated to compute the minimal policy that takes the AUV as close as possible to the target location using only the working thrusters. The top n policies, where in our experiments $n = 5$, are stored and later used for covariance analysis, which will be described in the next section.

C. Covariance Analysis

The policies in the previous section were learned in a noiseless environment. Often, the real world has noises that have not been accounted for in the simulation. As a result a policy must be tested for robustness against such noises. The policy with the highest reward is not necessarily the most robust policy. We use this scenario as an example to show how covariance analysis can be used to select a robust policy. In order to perform covariance analysis, the robot picks the top policies from the noiseless simulations and runs them in a noisy simulation. In our experiments the noise is introduced by adding a Gaussian noise to the input of the thrusters. Each policy is run thirty times and scored using the proposed scoring metric, which will be described in the rest of this section.

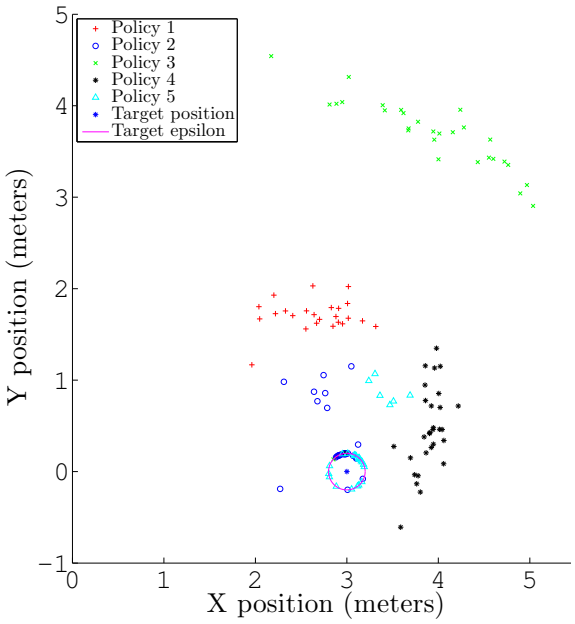


Fig. 3. The positions reached by the robot during a noisy simulation. Policy 1 had the highest reward, while Policy 5 was the fifth ranking policy.

The covariance, Q, of k variables is a $k \times k$ matrix that is calculated from N samples using (3), where x_{ni} is the n^{th} sample from the i^{th} variable, and q_{jm} is the covariance between j^{th} and m^{th} variables.

$$q_{jm} = \frac{1}{N-1}\sum_{i=1}^{N}(x_{ij}-\bar{x}_j)(x_{im}-\bar{x}_m) \quad \forall\ j,m\ \in\ 1 \leq k. \quad (3)$$

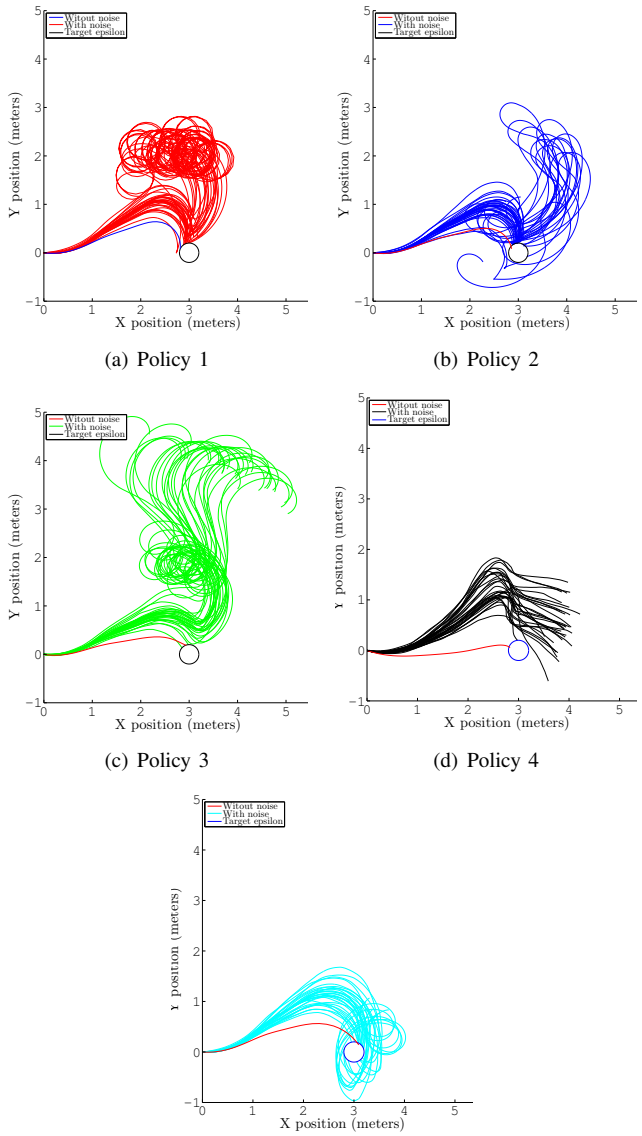
We use (3) to calculate the covariance matrix using the thirty samples with two variable: x and y position of the AUV. As illustrated in (4), for each policy, p, we extract a single value, C_P, from the covariance matrix, $Q_{k\times k}$, by taking the product of the diagonal of the matrix. Any other metric such as summation of the diagonals can also work.

$$C_p = \prod_{i=1}^{k} Q_{ii}^{p} \quad (4)$$

The reward for the scoring metric is calculated using (5). It takes into account two factors: the time, t, it takes to reach the target, and the distance, d, from the target. We want to make sure that a policy that cannot reach, or takes a long time to reach the target position is penalized. Therefore, a policy is scored using the formula in (6), where $\hat{R}_p$ and $\hat{C}_p$ are normalized reward and covariance of a policy, and α determines the compromise between accuracy and robustness.

$$R_p = \mathbb{E}[d(x_{AUV}, x_{target})] + \mathbb{E}[t_{target}] \quad (5)$$

$$score = \alpha\hat{R}_p + (1-\alpha)\hat{C}_p \quad (6)$$



(a) Policy 1 (b) Policy 2

(c) Policy 3 (d) Policy 4

(e) Policy 5

Fig. 4. Simulation results showing the effect of noise on top five policies. The policies are ranked according their reward. The figures illustrate that policies with higher ranking are not necessarily robust to noise in the environment.

V. Results

Fig. 3 shows the results of running the top five polices in a noisy environment. The figure plots the final destination of the robot and the target position. We also mark the area around the target position that is within the tolerance, ε, of acceptable error between the target position and the robot's position. In our experiments, ε was set to 2 cm. It is evident that the robustness to noise is not necessarily correlated to the ranking of a policy in the noiseless simulation. We also notice that the highest ranking policy is performing worse than lower ranking policies.

Fig. 4 shows the path taken by each policy when executed in a noisy environment. The figures display paths for both noiseless and noisy environments. We observe that the top policy exhibits an erratic behavior when it fails, taking longer

TABLE I
POLICY SELECTION AS THE VALUE OF α IS CHANGED

α	0	0.25	0.50	0.75	1.00
Policy	5	5	5	2	2

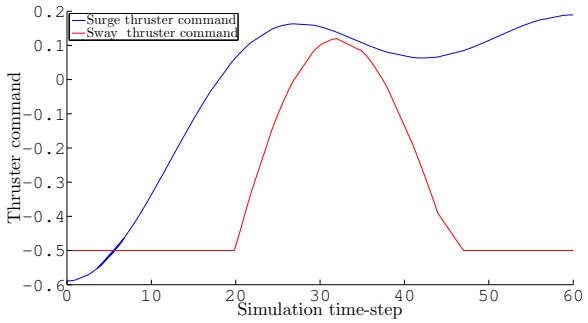


Fig. 5. The thruster commands. The blue curve is the command for the surge thruster and the red curve is the command for the sway thruster. The figure shows that in our experiments, the sway thrusters reach saturation.

paths and performing loops. Other policies are also affected by noise. However, Policy 2 and Policy 5 are still able to reach the target position. While the two policies perform similarly when only target position is considered, Policy 5 performs better on the distance measure.

We also evaluated the effect of the α on the scoring. Table I shows the results of the scoring policy. Not surprisingly, for an α value of upto 0.5, the scoring selects policy 5, which reflects our visual analysis of the policies.

We also noticed an unexpected behavior in the data. Since we add a Gaussian noise, we expected the paths in the noisy environment to have a Gaussian distribution around the path in the noiseless environment. Our investigations revealed that this is due to the hardware limitations. Fig. 5 shows the commands sent to the thrusters. The blue curve is the command for the surge thruster and the red curve is the command for the sway thruster. The thruster commands are produced by the learned policy. However, the thrusters only operate in the range [-1,1] and [-0.5,0.5] for surge and sway thrusters, respectively. As shown in Fig. 5, in our experiments, the sway thrusters sometimes reach saturation. Therefore, in the robustness test simulations, adding noise to the already saturated thruster has no effect unless the addition of the noise moves the thruster away from saturation. We also observe that the curve does not have a zero mean, hence, the saturation affects the negative thruster values more. Thereby, the AUV has a tendency to sway more towards the positive direction, which explains the bias noticed in our data.

VI. CONCLUSIONS

In this paper we have presented a theoretical framework for covariance analysis as a metric to evaluate the performance of a policy learned using reinforcement learning. We have also proposed a scoring metric that uses the output of the covariance analysis to evaluate the performance of a policy. We have shown a real-world application by applying the method to the problem of a thruster failure in an AUV. We show that using the proposed analysis, we have discovered a policy with better performance in the presence of environmental noise.

The method is not limited to underwater robotic problems. It can be applied to other problems where there are multiple possible policies in the solution space and a covariance matrix can be computed for them.

VII. ACKNOWLEDGMENTS

We would like to thank Arnau Carrera, Narcís Palomeras, and Marc Carreras from the Computer Vision and Robotics Group, University of Girona, Spain, for providing the specialized underwater simulator UWSim and detailed dynamics model of the Girona 500 AUV. We are also grateful to Charalampos Bechlioulis, Kostas Kyriakopoulos, and George Karras from the Department of Mechanical Engineering, National Technical University of Athens, Greece, for providing the dynamics model of Girona 500 obtained through least-squares system identification.

REFERENCES

[1] D. Sternad, M. O. Abe, X. Hu, and H. Mueller, "Neuromotor noise, error tolerance and velocity-dependent costs in skilled performance," *PLoS Comput. Biol.*, vol. 7, no. 9, 2011.

[2] D. M. Lane, F. Maurelli, P. Kormushev, M. Carreras, M. Fox, and K. Kyriakopoulos, "Persistent autonomy: the challenges of the PANDORA project," in *Proc of IFAC MCMC*, 2012.

[3] PANDORA, "Persistent Autonomy through learNing, aDaptation, Observation and Re-plAnning," http://persistentautonomy.com, 2013.

[4] M. Leonetti, S. R. Ahmadzadeh, and P. Kormushev, "On-line learning to recover from thruster failures on autonomous underwater vehicles," in *Proc. MTS/IEEE OCEANS*, San Diego, USA, 2013.

[5] D. Ribas, N. Palomeras, P. Ridao, M. Carreras, and A. Mallios, "Girona 500 AUV: From survey to intervention," in *IEEE/ASME Trans. on Mech.*, vol. 17, no. 1, pp. 46–53, 2012.

[6] J.Peters and S.Schaal, "Policy gradient methods for robotics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2006.

[7] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomput.*, vol. 71, no. 7-9, pp. 1180–1190, 2008.

[8] J. Kober and J. Peters, "Learning motor primitives for robotics," in *Proc. IEEE Int. Conf. Robot. Auto.*, 2009, pp. 2112–2118.

[9] R. Rubinstein and D. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer, 2004.

[10] N. Hansen, "The CMA evolution strategy: A comparing review," in *Towards a new evolutionary computation.* Springer, 2006, pp. 75–102.

[11] F. Stulp and O. Sigaud, "Path integral policy improvement with covariance matrix adaptation," in *Proc. Int. Conf. on Machine Learning*, 2012.

[12] P. Kormushev, S. Calinon, D. G. Caldwell, and B. Ugurlu, "Challenges for the policy representation when applying reinforcement learning in robotics," in *IEEE Int. Joint Conf. on Neural Networks,* 2012, pp. 1–8.

[13] R. S. Sutton and A. G. Barto, *"Reinforcement learning : an introduction"*, Adaptive computation and machine learning. Cambridge, MA, USA: MIT Press, 1998.

[14] G. Karras, S. Loizou, and K. Kyriakopoulos, "Towards semi-autonomous operation of under-actuated underwater vehicles: sensor fusion, on-line identification and visual servo control," *Autonomous Robots*, pp. 67–86, 2011.

[15] G. Karras, C. Bechlioulis, M. Leonetti, N. Palomeras, P. Kormushev, K. Kyriakopoulos, and D. G. Caldwell, "On-line identification of autonomous underwater vehicles through global derivative-free optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013.

[16] S. R. Ahmadzadeh, M. Leonetti, A. Carrera, M. Carreras, P. Kormushev, and D. G. Caldwell, "Online discovery of AUV control policies to overcome thruster failures," in *Proc. IEEE Int. Conf. Robot. Auto,* 2014.

[17] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.