On-line Learning to Recover from Thruster Failures on Autonomous Underwater Vehicles

Matteo Leonetti, Seyed Reza Ahmadzadeh, Petar Kormushev
Department of Advanced Robotics
Istituto Italiano di Tecnologia
via Morego 30, 16163 Genova
Email: {matteo.leonetti, reza.ahmadzadeh, petar.kormushev}@iit.it

***Abstract*—We propose a method for computing on-line the controller of an Autonomous Underwater Vehicle under thruster failures. The method is general and can be applied to both redundant and under-actuated AUVs, as it does not rely on the modification of the thruster control matrix. We define an optimization problem on a specific class of functions, in order to compute the optimal control law that achieves the target without using the faulty thruster. The method is framed within model-based policy search for reinforcement learning, and we study its applicability on the model of the AUV Girona500. We performed experiments with policies of increasing complexity, testing the on-line feasibility of the approach as the optimization problem becomes more complex.**

I. Introduction

Autonomous Underwater Vehicles (AUVs) have to deal with long missions in unknown environments. The risk of damage is of severe concern, owing to many factors, among which: extreme pressure, corrosive effects of sea water, the risk of damage due to waves while on the surface, and collisions during launch or later. Improving the reliability of the robot is therefore critical, especially for autonomous vehicles. The European project PANDORA (Persistent Autonomy through learNing, aDaptation, Observation, and Re-plAnning) [9], [13] is concerned with the creation of AUVs able to react to unexpected conditions, and cope with extreme uncertainty. For a PANDORA AUV, being able to reach a safe location even in case of a thruster failure is crucial.

Fault-tolerant control is the field related to identification and reaction to faults and failures [2]. A fault is defined as a temporary malfunctioning of a component, which deviates from its correct behavior. A failure is defined as the permanent loss of a component's functionality. We consider the case of thruster failures, on AUVs where thrusters are the only actuators (as opposed to AUVs where the attitude is determined by control surfaces). Methods for fault-tolerant control can be divided into passive and active ones [11]. Passive methods treat faults as sources of system uncertainty, and design a controller for the worst possible scenario. Those approaches do not rely on fault detection, as the resulting controller is passively resilient to all the faults considered at design-time. Active methods, on the other hand, are less conservative, and react to faults only when necessary. Once a fault is detected, a controller can be either selected from a set of pre-defined controllers, or computed on-line. While active methods can yield better performance than passive ones, since they are not designed for the worst possible scenario but adapt to the actual situation, their effectiveness relies heavily on the reliability of the fault detector. Our method belongs to the category of active fault-tolerant control, because we compute a controller on-line. Since we restrict the possible faults to thruster failures, the dependency on failure detection is less critical, as this problem has been extensively considered in the literature and has several effective solutions [3]. Most fault-tolerant control methods for thruster failure take advantage of redundancy and follow a desired trajectory by reallocating the force commands on the working thrusters [1], [12], [15], [19]. While the problem in case of redundancy has been extensively considered, if a broken thruster makes the AUV under-actuated the literature is lacking in unifying approaches [14].

We propose a generic method, which can be used in case of both redundant and under-actuated AUVs, for computing a controller to navigate to a target location under thruster failures. We demonstrate it on the model of Girona500 [17]. Girona500 is a reconfigurable AUV equipped with typical navigation sensors (DVL, AHRS, pressure gauge and USBL) and basic survey equipment (profiler sonar, side scan sonar, video camera and sound velocity sensor). The layout we used has 5 thrusters (Figure 1): two vertical to actuate the heave, one lateral for the sway, and two horizontal for the yaw and surge.

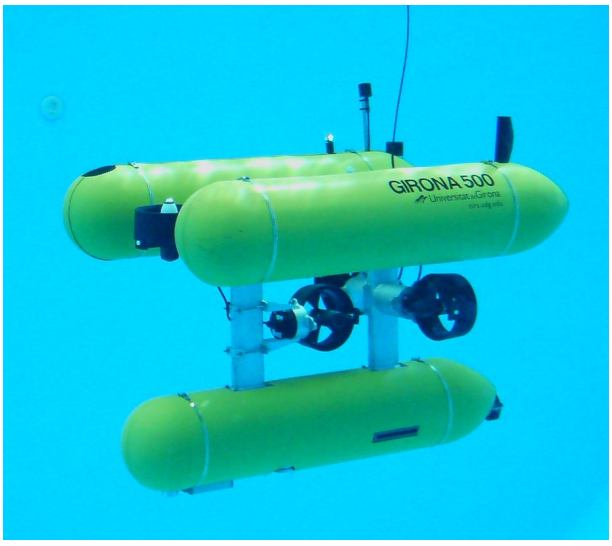

Fig. 1. The Girona500 AUV.

II. METHOD DESCRIPTION

We frame our method in the context of model-based policy search for reinforcement learning [18]. This framework comprises a dynamic model of the vehicle, a parametrized representation for the control law, the definition of a cost function, and an optimization algorithm.

A. Dynamic Model

A model of the environment is learned and continually updated through an on-line identification algorithm [5]. We employ a standard dynamic model [4], where the AUV is represented as a rigid body subject to external forces and torques:

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

in which: M is the mass matrix; C is the Coriolis matrix; D, is the drag matrix; $g(n)$ is the hydrostatic restoring force vector; $J\left(\eta\right)$ is the Jacobian matrix transforming the velocities from the body-fixed to the earth-fixed frame; $\eta = \left[x\ y\ z\ \phi\ \theta\ \psi\right]^T$ is the pose (position and orientation) vector; $\mathbf{v} = \left[u\ v\ w\ p\ q\ r\right]^T$ is the body velocity vector; τ is the force/torque vector; u is the input vector and B is the thruster control matrix. When the AUV is redundant most methods act on the matrix B, to obtain the required forces on the vehicle by reallocating the command on the functioning thrusters. The simplest way to modify B is deleting the columns correspondent to the faulty thrusters. We propose a different approach, and compute a new command function u to reach a given target without modifying B. Therefore, this approach is feasible also in under-actuated vehicles.

B. Policy representation

The control law (function u in Section II-A) is represented as a function $\pi(x|\theta)$ (called *policy*) of the observations, depending on a parameter vector θ. The policy representation is critical in reinforcement learning, and is one of the degrees of freedom of this approach, which allow for great flexibility. The policy representation determines the hypothesis set for learning, that is the search space of the learning algorithm.

In this work the policy is represented with a linear function approximator, that is a function of the form

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

where the functions $\phi_i(\mathbf{x})$ are called *basis* functions, or *features*, and may be stochastic. We use the Fourier basis functions [8] $\phi_i = cos(\pi \mathbf{c_i} \cdot \mathbf{x})$. The coefficients in $\mathbf{c_i}$ are such that $\mathbf{c_{ij}} \in \{\mathbf{0}, \ldots, \mathbf{n}\}$, and determine the order of the approximation and the correlation between the observation variables. If only one coefficient, say $c_{ik} \neq 0$, is non zero and $c_{ij} = 0$ for $j \neq k$, then the variables are considered independent. Different choices are possible for the observation vector x, and a few will be discussed in Section III

C. Cost function

The performance of the vehicle is measured through a cost function:

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

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. In policy search over a finite horizon, the particular path followed by the agent in the state space can be ignored, and the optimization treated with black-box methods over θ. It is more common, in reinforcement learning, tu assume a Markovian system, and learn a value function over the state space, to the aim of minimizing the expected cost from each state. In policy search, on the other hand, we minimize the cost by modifying the whole policy (not state by state) through the parameter vector θ. This provides a representational advantage, since value functions become quickly intractable as the number of state variables increases. On the other hand, it usually comes at the cost of a higher number of training samples necessary for learning. Therefore, we employ a model-based approach, where trials are performed on the model and not directly by the vehicle. For AUVs this is not a practical limitations, as their dynamics has been modeled accurately. The cost function is the other degree of freedom of our approach. Many different definitions of the imeddiate costs are possible. Secondary objectives can be added, such as velocity error (as will be shown in the following) or energy consumption.

The optimization algorithm we use is a global derivative-free algorithm introduced by Leonetti et al. [10], which has also been used for system identification [6]. The algorithm is a modification of the algorithm by Price [16], and combines a global controlled stochastic search and a deterministic local search. Since it is a global algorithm, it is particularly suitable for decision problems created on the fly and solved on-line, without the possibility to engineer a good initial solution. In the following, we shall refer to this algorithm as Modified Price (MP). As a baseline, we also provide results with the well-known Simulated Annealing [7]. The particular algorithm used is not critical for the method, as long as it is fast enough to be used on-line.

D. On-line procedure

When a thruster is deemed faulty, a function J is created to represents the cost of a path to the target location. The optimization algorithm is then used to compute the minimal policy, in the given policy parametrization, that takes the AUV as close as possible to the target location using only the working thrusters. The function π computed substitutes the AUV's controller that would work under normal operating conditions. It is also possible to use the target location as a waypoint, by adding to J a secondary optimization objective (appropriately weighed) to reach the point with a given velocity.

III. EXPERIMENTS

We performed our experiments on the dynamic model of Girona500, whose parameters have been identified. We limited

the trials to the horizontal plane, where we supposed the left-hand surge thruster to be broken. Therefore, the AUV can navigate only with the right-hand surge thruster and the sway one. We used the following definition of the immediate cost:

$$c_t(\langle p_t, v_t \rangle) = \begin{cases} \| p_t - g_p \| & \text{if } t < T \\ w \| v_t - g_v \| & \text{if } t = T \end{cases} \quad (1)$$

where the observation $x_t = \langle p_t, v_t \rangle$ is composed by position and velocity at time t, g_p is the goal location, g_v is the goal velocity and w weighs the velocity objective with respect to the positional one. For all our experiments we use $T = 60$s, since all the target destinations are reachable in 60 seconds. For the policy, we start with constant values (a Fourier expansion of order zero) and then increase the order of the representation. We are interesting in assessing how complex the representation can be for the method to be applicable on-line.

A. *Constant Policy*

In order to test the presented optimization algorithms, we define a desired position in the vicinity of the robot, and a target velocity of $\langle 0, 0 \rangle$. In the first experiment a constant policy vector is considered, which means applying a constant voltage function on the undamaged thrusters to move the AUV towards the desired target position. We repeated the optimizing process 50 times for each optimization algorithm. The trajectories resulting by the application of the optimum constant policies computed by the two algorithms can be seen in Figure 2. The figure shows that the solutions found by the two optimization algorithms are very similar. The acquired

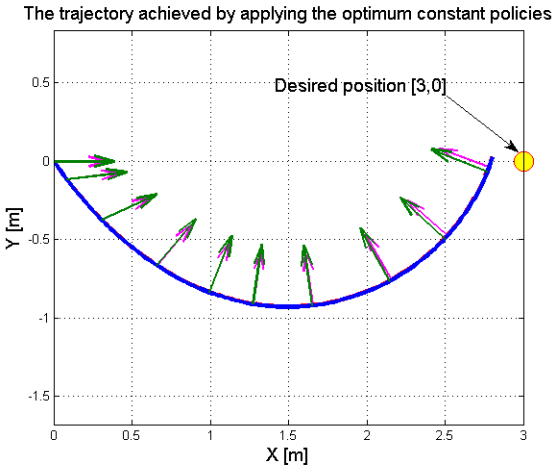

Fig. 2. Trajectories and orientation with constant policies.

velocity profiles can be seen in the Fig. 3. The resulting velocity profiles are also very similar in both cases. Since both algorithms are stochastic, the results may converge to different solutions in different runs. In Table I we report the mean, median, standard deviation, and interquartile range for the distributions of the values of the optimal policies over the 50 experiments, together with the number of iterations to compute them. In addition, the box plot of the statistical results is depicted in Figure 4. The central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers

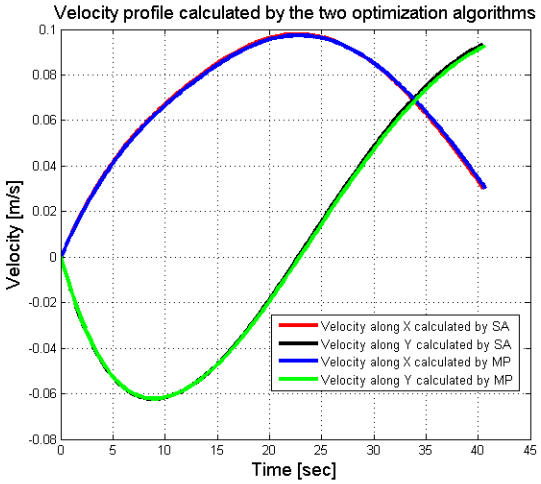

Fig. 3. Velocities with constant policies.

TABLE I. PERFORMANCE OF THE TWO OPTIMIZATION ALGORITHMS WITH CONSTANT POLICIES. FCNVAL REPRESENTS THE BEST VALUE FOUND FOR THE OBJECTIVE FUNCTION AND NUMITER IS THE NUMBER OF ITERATIONS.

	Modified Price		Simulated Annealing	
	numIter	fcnVal	numIter	fcnVal
mean	139	772.25	399	798.32
median	132	766.97	373	776.04
std	41	24.23	135	55.19
iqr	45	1.78	139	25.63

extend to the most extreme data points not considered outliers, and outliers are plotted individually. As it can be seen from the

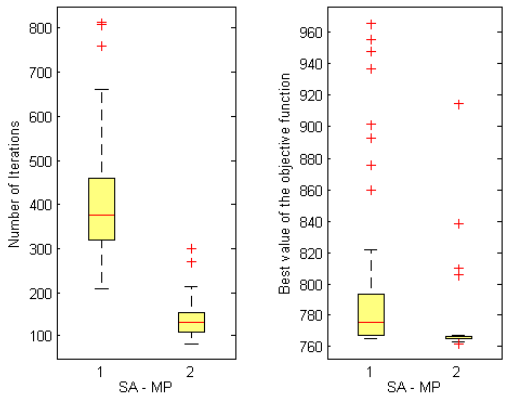

Fig. 4. Comparing the optimization algoriths with constant policies values over 50 experiments.

results, it takes longer for Simulated Annealing to achieve the same quality of the solutions as for Modified Price. Generally, a constant policy can get the AUV close enough to the target but has little control on the velocity. In order to have the velocity close to a target as a secondary objective we need a more flexible control law.

B. Time-dependent Policy

In this experiment we employ an order 3 approximator using the Fourier bases, as explained in Section II-B. The policy is dependent on time only, and has 8 parameters to optimize. On the one hand, the target velocity of $\langle 0, 0 \rangle$ becomes more relevant here, as the control law can be more flexible than the constant policy we previously used. On the other hand, while the optimization was only on 2 parameters in the previous experiments (a constant command per working thruster), it depends on four times more variables here. We repeated the optimizing process 50 times for each optimization algorithm.

As can be seen in Figure. 5 the acquired trajectories from both algorithms are similar. The velocity profiles, however, are different. As it is shown in Figure. 6, the final part of the velocity profiles acquired by the Modified Price algorithm converges to the target velocity more closely than Simulated Annealing. In Table II we report the mean, median, standard deviation, and interquartile range for the distributions of the values of the optimal policies over the 50 experiments for the time-dependent policy experiment, together with the number of iterations to compute them. In addition, the box plot of the statistical results is depicted in Figure 7.

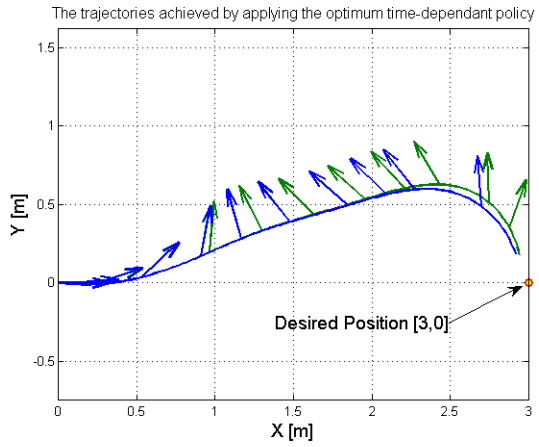

Fig. 5. Trajectories and orientations with time-dependant policies.

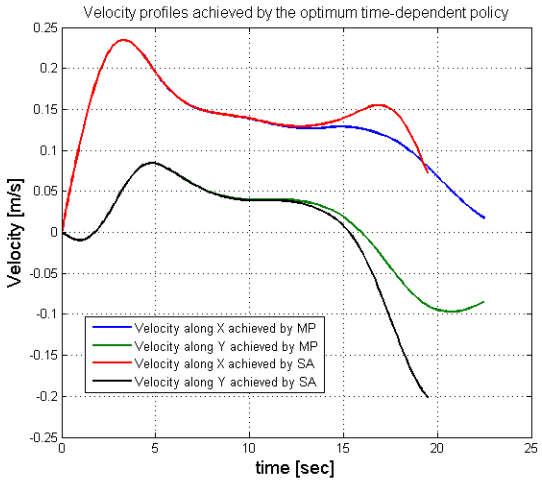

Fig. 6. Velocities with time-dependant policies.

TABLE II. PERFORMANCE OF THE OPTIMIZATION ALGORITHMS WITH TIME-DEPENDANT POLICIES. FCNVAL REPRESENTS THE BEST VALUE FOUND FOR THE OBJECTIVE FUNCTION AND NUMITER IS THE NUMBER OF ITERATIONS.

	Modified Price		Simulated Annealing	
	numIter	fcnVal	numIter	fcnVal
mean	1609	412.2	496	435.24
median	984	384.31	421	406.18
std	1822	56.1	248	86.75
iqr	841	115.48	351	70.65

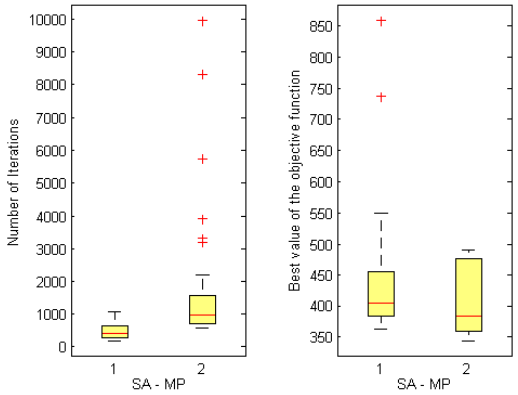

Fig. 7. Comparison of the optimization algorithms with the time-dependant policy representation over 50 experiments.

C. State-dependent Policy

In the last experiment on policy complexity, we increase the representation to include position, orientation and velocities in the observations. An order-3 Fourier policy with 5 state variables (two variables for position and velocity, plus one for orientation), produces an optimization problem with 18 variables, which is considered quite significant for policy-search methods. The robustness provided by the reactivity to more information is traded with computational complexity. We repeated the optimization process 50 times for each optimization algorithm.

As can be seen in Figure. 8 the acquired trajectories from both algorithms are similar. The velocity profiles, however, are different again. As it is shown in Figure. 9, the final part of the velocity profiles acquired by the Modified Price algorithm converges to the target velocity more closely than Simulated Annealing. In Table III we report the mean, median, standard deviation, and interquartile range for the distributions of the values of the optimal policies over the 50 experiments for the state-dependent policy experiment, together with the number of iterations to compute them. In addition, the box plot of the statistical results is depicted in Figure 10.

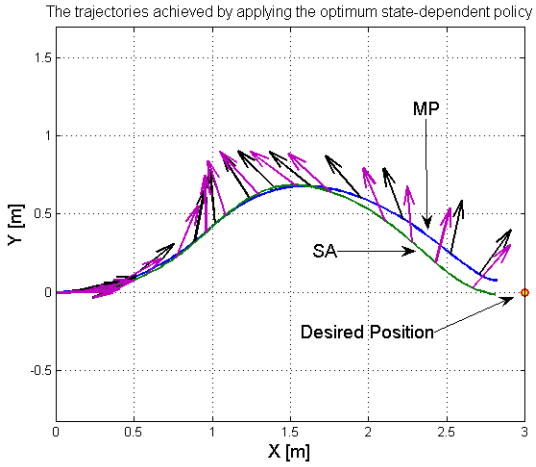

Fig. 8. Trajectories with state-dependant policies.

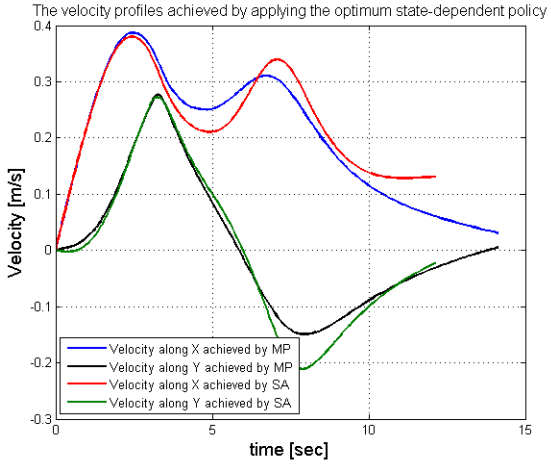

Fig. 9. Velocities with state-dependant policies.

D. Point-to-Point Navigation

In the previous experiments we focused on the complexity of the policy and the corresponding accuracy in reaching a target position and velocities. In this experiment we show that the feasibility of the approach, with a time-dependent order-3 Fourier policy, which is also valid for other target locations. We generate a grid of target points with coordinates in $[-10, 10]$ at 1 meter distance with respect to each other, and null target velocity. We show some of the generated trajectories in Figure 11. The AUV was able to reach each point at a distance of less then 0.2m and stop there, since the target velocity was the null vector.

TABLE III. PERFORMANCE OF THE TWO OPTIMIZATION ALGORITHMS USING STATE-DEPENDANT POLICIES. FCNVAL REPRESENTS THE BEST VALUE FOUND FOR THE OBJECTIVE FUNCTION AND NUMITER IS THE NUMBER OF ITERATIONS.

	Modified Price		Simulated Annealing	
	numIter	fcnVal	numIter	fcnVal
mean	5287	217.6	511	265.89
median	4214	219.5	456	254.01
std	2658	8.5	240	39.05
iqr	1947	14.7	222	16.67

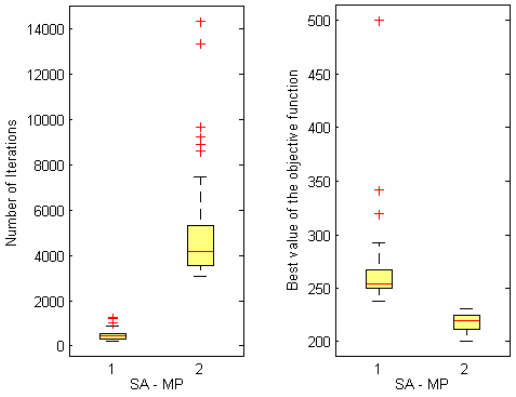

Fig. 10. Comparison of the optimization algorithms with the state-dependant policy representation over 50 experiments.

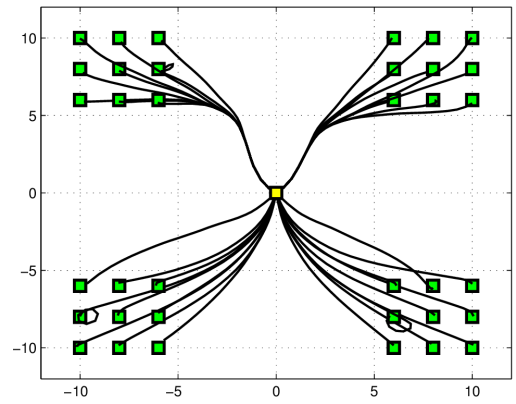

Fig. 11. The result trajectories for the point-to-point navigation experiment.

E. Navigating Through Waypoints

Point to point navigation is the most common when the objective of the AUV is just to reach a safe location, and possibly float to the surface. The presented method, however, can also be used to follow a trajectory with waypoints. We generate two trajectories to reach a point 50m far, one on a straight line and the other one on an arc of circumference. Along each trajectory, we generate a waypoint every 5m. We iteratively pose the problem of reaching the next waypoint from the current state (position and velocity), with a target velocity pointing towards the subsequent waypoint and norm equal to 0.7, the highest linear velocity for Girona500. The trajectories and the orientation of the AUV are shown in Figure 12. The AUV learns to proceed laterally, using the forward thruster to control the orientation. Sometimes the AUV happens to turn around, but it is always able to recover towards the next waypoint.

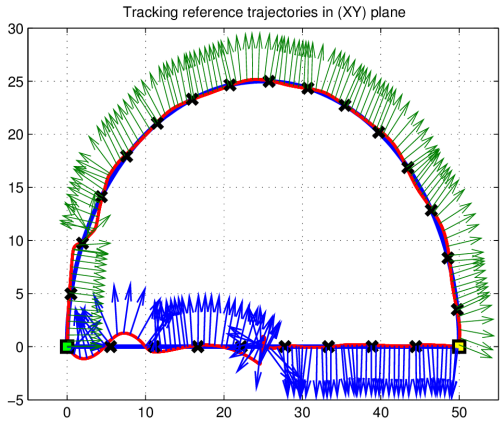

Fig. 12. The desired and result trajectories in the case that the AUV navigates through waypoints. The blue trajectory is the desired and the red is the acquired trajectory).

F. Computational Cost

For the feasibility of the presented method we need to assure that the optimization of the policy can be performed on-board in a short time. Figure 13 shows the quality of the solution (distance from the target) computed over time for a waypoint 5m away. The plot is averaged over 20 runs. We used an Intel core 2 Duo P8700 processor, where the optimization ran on a single thread. It took 12 seconds to find a solution able to take the AUV only 0.5m from the target. The autonomous vehicle can stop the optimization process at any time after this point, having guaranteed a solution good enough to reach the target.

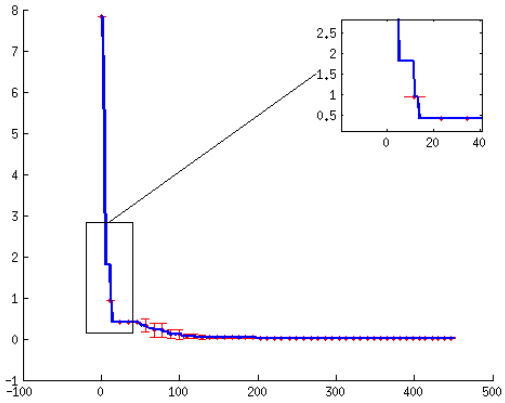

Fig. 13. Computational cost of the learning method to find the (global) optimum solution.

IV. Acknowledgments

This work was supported by the EU funded project PANDORA: Persistent Autonomy through learNing, aDaptation, Observation and ReplAnning", FP7-288273, 2012-2014.

References

[1] A. Alessandri, M. Caccia, and G. Veruggio, "Fault detection of actuator faults in unmanned underwater vehicles," *Control Engineering Practice*, vol. 7, no. 3, pp. 357–368, 1999.

[2] G. Antonelli, "A survey of fault detection/tolerance strategies for auvs and rovs," in *Fault diagnosis and fault tolerance for mechatronic systems: Recent advances*. Springer, 2003, pp. 109–127.

[3] ——, *Underwater Robots: Motion and Force Control of Vehicle-Manipulator Systems (Springer Tracts in Advanced Robotics)*. Springer-Verlag New York, Inc., 2006.

[4] T. Fossen, "Guidance and control of ocean vehicles," *Wiley, New York*, 1994.

[5] 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.

[6] G. C. Karras, C. P. Bechlioulis, M. Leonetti, N. Palomeras, P. Kormushev, K. J. Kyriakopoulos, and D. G. Caldwell, "On-line identification of autonomous underwater vehicles through global derivative-free optimization," in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[7] S. Kirkpatrick, D. G. Jr., and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[8] G. Konidaris, S. Osentoski, and P. S. Thomas, "Value function approximation in reinforcement learning using the fourier basis," in *AAAI*, 2011.

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

[10] M. Leonetti, P. Kormushev, and S. Sagratella, "Combining local and global direct derivative-free optimization for reinforcement learning," *Cybernetics and Information Technologies*, vol. 12, no. 3, pp. 53–65, 2012.

[11] M. Mahmoud, J. Jiang, and Y. Zhang, *Active fault tolerant control systems: stochastic analysis and synthesis*. Springer, 2003, vol. 287.

[12] E. Omerdic and G. Roberts, "Thruster fault diagnosis and accommodation for open-frame underwater vehicles," *Control Engineering Practice*, vol. 12, no. 12, pp. 1575–1598, 2004.

[13] PANDORA, "Persistent autonomy through learning, adaptation, observation, and re-planning," 2012. [Online]. Available: http://www.persistentautonomy.com

[14] D. Perrault and M. Nahon, "Fault-tolerant control of an autonomous underwater vehicle," in *OCEANS'98 Conference Proceedings*, vol. 2. IEEE, 1998, pp. 820–824.

[15] T. K. Podder, G. Antonelli, and N. Sarkar, "Fault tolerant control of an autonomous underwater vehicle under thruster redundancy: Simulations and experiments," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 1251–1256.

[16] W. Price, "Global optimization by controlled random search," *Journal of Optimization Theory and Applications*, vol. 40, no. 3, pp. 333–348, 1983.

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

[18] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[19] K. C. Yang, J. Yuh, and S. K. Choi, "Experimental study of fault-tolerant system design for underwater robots," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 2. IEEE, 1998, pp. 1051–1056.