Online Discovery of AUV Control Policies to Overcome Thruster Failures

Seyed Reza Ahmadzadeh[1], Matteo Leonetti[2], Arnau Carrera[3], Marc Carreras[3], Petar Kormushev[1], and Darwin G. Caldwell[1]

***Abstract*— We investigate methods to improve fault-tolerance of Autonomous Underwater Vehicles (AUVs) to increase their reliability and persistent autonomy. We propose a learning-based approach that is able to discover new control policies to overcome thruster failures as they happen. The proposed approach is a model-based direct policy search that learns on an on-board simulated model of the AUV. The model is adapted to a new condition when a fault is detected and isolated. Since the approach generates an optimal trajectory, the learned fault-tolerant policy is able to navigate the AUV towards a specified target with minimum cost. Finally, the learned policy is executed on the real robot in a closed-loop using the state feedback of the AUV. Unlike most existing methods which rely on the redundancy of thrusters, our approach is also applicable when the AUV becomes under-actuated in the presence of a fault. To validate the feasibility and efficiency of the presented approach, we evaluate it with three learning algorithms and three policy representations with increasing complexity. The proposed method is tested on a real AUV, Girona500.**

I. INTRODUCTION

Nowadays Autonomous Underwater Vehicles (AUVs) are required to operate over longer missions while dealing with extreme uncertainties in unstructured environments. In such conditions an undetected failure can lead to the loss of the vehicle, which is a dramatic event. Even in the case that the failure is detected, in order to terminate the mission and rescue the AUV safely, a fault-tolerant strategy must be considered. In Remotely Operated Vehicles (ROVs), a failure detection strategy helps a skilled human operator to make a proper decision. The human expert then decides whether to terminate the mission or to take a proportionate fault-tolerant strategy, e.g., turning off a thruster and use the others instead.

Although the failure can happen in all subsystems of the ROV or AUV, in this paper we focus on the case of thruster failure. Thruster blocking, rotor failure, and flooded thrusters are some of the factors that can lead to a thruster failure in real missions [2].

By definition, fault detection is the process of monitoring a system in order to recognize the presence of a failure; fault isolation or diagnosis is the capability to determine which specific subsystem (thruster in our case), is subject to failure. The fault detection and isolation scheme for thruster failures has been extensively investigated in the literature and has several effective solutions [2], [3], [4], [5].

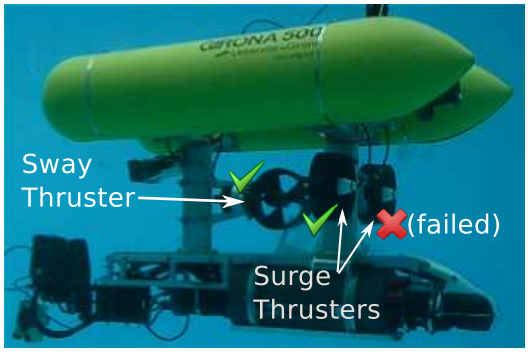


Fig. 1. The Girona500 AUV equipped with 5 thrusters (3 are visible in the photo). In our experiment, one of the surge thrusters is broken.

Fault tolerance is the capability to complete the mission despite the failure of one or more subsystems. It is referred to also as fault control, fault accommodation or control reconfiguration. Most of the existing fault-tolerant schemes consider some actuator redundancies, so that the vehicle remains actuated in the Degree of Freedom (DOF) of interest, even if a fault occurs in one of the thrusters. For this category of problems a general solution has been found: reallocating the desired forces on the vehicle over the working thrusters [3], [2], [6], [7]. While the problem has been extensively considered in the case of actuator-redundant vehicles, the literature is still lacking a unifying approach if a broken thruster makes the AUV under-actuated [8]. A few works are targeted at AUV controlled with surfaces [9], [10], [11]. Those methods are specific to the kinematics and dynamics models of the considered AUV. Our method, on the other hand, makes use of the model for simulation, but not in the derivation of the controller, which is of a pre-defined form. We use a linear function approximator to represent the policy, whose parameters are learned depending on the AUV model and the particular task at hand. M. Andonian et al. [12], design trajectories for an AUV to overcome an actuator failure and accomplishing the mission using geometric control theory. They model the AUV as a forced affine connection control system, and develop the control strategies through the use of integral curves. They present a scenario and compute the control signals in

This research was supported by the PANDORA EU FP7 project [1] under the grant agreement No. ICT-288273. http://persistentautonomy.com/

[1] Department of Advanced Robotics, Istituto Italiano di Tecnologia, Via Morego, 30 16163, Genova, Italy. {reza.ahmadzadeh, petar.kormushev, darwin.caldwell} @iit.it

[2] Department of Computer Science, The University of Texas at Austin, 2317 Speedway, Austin, TX 78712, matteo@cs.utexas.edu

[3] Computer Vision and Robotics Group (VICOROB), University of Girona, 17071, Girona, Spain. {marc.carreras, arnau.carrera}@udg.edu

simulation. The presented geometric control is an open-loop control strategy, therefore its validity is mainly theoretical, because of the inevitable presence of unmodeled dynamics and external disturbances. Finally, Choi and Kondo [13], provide an analysis of the thruster failure combinations for a vehicle similar to ours. Their method is simple and in some circumstances applicable to our problem. However, as most of the other papers, it addresses the problem of tracking a given trajectory, and does not take into account the trajectory generation. While this is a common and relevant control problem, in the case of a thruster failure the pre-defined trajectory for a functional AUV may not be optimal for the faulty AUV anymore. Our method generates an optimal trajectory, that is a trajectory that accomplishes the given task achieving the lowest cost in the presence of the fault. Nonetheless, their method could be combined with our open-loop policy, providing an alternative way of having feedback control on a trajectory designed for the faulty AUV.

We consider the problem of using the functional thrusters to bring the vehicle safely to a station where it can be rescued, when the thruster failure reduces the mobility of the vehicle, and hence it cannot maneuver as previously prescribed. We build on the work by Leonetti et al. [14] which introduced a method to compute a fault-tolerant policy, and can be utilized in both the cases of redundant and under-actuated AUVs. The original work is limited in its applicability in open water by the fact that the only policy that can be computed online is an open-loop function of time. In this paper, we performed extensive simulated experiments and individuated an optimization algorithm that can compute online a state-dependent policy, closing the loop with perceptions. Consequently, we are able to evaluate the resulting controller on the real AUV. The AUV we use for our experiments is Girona500 [15] which is used in PANDORA [1]. Girona500 is a reconfigurable AUV equipped with typical navigation sensors (e.g. Doppler Velocity Log, etc.), basic survey equipments (e.g. side scan sonar, video camera, etc.), and various thruster layouts. In the layout we selected, the AUV is equipped with 5 thrusters: 2 heave, 2 surge, and 1 sway thrusters.

II. Methodology

We frame our approach in the context of model-based direct policy search for reinforcement learning. This framework comprises a dynamic model of the vehicle (Equation 1), a parameterized representation for the control policy, a cost function, and an optimization algorithm.

A. AUV Model

According to the standard modeling procedure for underwater vehicles [16], an AUV can be modeled as a rigid body subject to external forces and torques while moving in a fluid environment. The 6 DOF equations of motion for the AUV are given in a compact form in (1).

$$\begin{aligned} \dot{\eta} &= \mathbf{J}(\eta)\,\nu \\ \mathbf{M}_{RB}\dot{\nu} + \mathbf{C}_{RB}(\nu)\,\nu &= -\mathbf{M}_A\dot{\nu} - \mathbf{C}_A(\nu)\,\nu \\ &\quad - \mathbf{D}(\nu)\,\nu - \mathbf{g}(\eta) + \mathbf{B}\mathbf{u} \end{aligned} \tag{1}$$

where $\eta \triangleq [x\ y\ z\ \phi\ \theta\ \psi]^T$ is the pose (position and orientation) vector with respect to the inertial frame and $\nu \triangleq [u\ v\ w\ p\ q\ r]^T$ is the body velocity vector defined in the body-fixed frame. $\mathbf{J}(\eta)$ is the velocity transformation matrix, $\mathbf{M}_{RB}$ is the rigid body inertia matrix, $\mathbf{C}_{RB}$ is the rigid body Coriolis and centripetal matrix, $\mathbf{M}_A$ is the hydrodynamic added mass matrix, $\mathbf{C}_A$ is the added mass Coriolis and centripetal matrix, $\mathbf{D}$ is the hydrodynamic damping matrix, $\mathbf{g}(\eta)$ is the hydrostatic restoring force vector, $\mathbf{B}$ is the actuator configuration matrix, and the vector $\mathbf{u}$ the control input vector or command vector. More details about the equations of motion can be found in [16]. In order to complete the dynamic model of the AUV, we use the hydrodynamic parameters of Girona500, which have been extracted using an online identification method and reported in [17]. In addition, it should be considered in the modeling of the system that the Girona500 AUV was designed to have passive stability in roll and pitch.

For over-actuated systems, since the vehicle remains redundant even after thruster failure, most approaches operate on the matrix $\mathbf{B}$, to obtain the required forces/torques by reallocating the command on the functional thrusters. The simplest way to modify $\mathbf{B}$ is ignoring the columns correspondent to the faulty thrusters. In this paper, on the other hand, we propose a different approach by computing a new command function $\mathbf{u}$ to reach a given target without modifying the matrix $\mathbf{B}$. Our approach is applicable in both cases of over-actuated and under-actuated vehicles.

B. Policy Representation

In this work we consider the control input vector $\mathbf{u}$ as a function $\Pi(\chi|\theta)$ of observation vector χ depending on a parameter vector θ. The policy is represented with a linear function approximator, that is a function of the form $\mathbf{u} = \Pi(\chi|\theta) = \theta^T\Phi(\chi)$, where $\Phi(\chi)$ is a matrix of basis functions or feature vectors ($\phi_i(\chi)$). Here we use Fourier basis functions because they are easy to compute accurately even for high orders, and their arguments are formed by multiplication and summation rather than exponentiation. In addition, the Fourier basis seems like a natural choice for value function approximation [18]. For each Fourier basis function $\phi_i = \cos(\pi\mathbf{c}_i \cdot \chi)$, the coefficient $\mathbf{c}_i$ determines the order of the approximation and the correlation between the observation variables. There are different choices for the observation vector χ, a number of which will be discussed in section IV.

C. Cost Function

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

$$\mathbb{J}(\theta) = \sum_{t=0}^{T} c_t(\eta_t)\Big|_{\Pi(\chi|\theta)} \tag{2}$$

where c_t is the immediate cost, and depends on the current state η_t, which in turn is determined by the policy and its parameters. Therefore, the aim of the agent is to tune the policy's parameters in order to minimize the cumulative cost $\mathbb{J}$ over a horizon T. We employ a model-based policy search approach where trials are performed on the model and not directly by the vehicle. For AUVs this is not a practical limitation, as their dynamics has been modeled accurately. The cost function is the other degree of freedom of our approach. Many different definitions of the immediate costs are possible. 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.

D. Optimization Algorithms

We implement three optimization algorithms to compare the quality and the computational feasibility of the solution for online discovery of the fault-tolerant policy. We use a derivative-free optimization algorithm introduced by Leonetti et al. [19], the well-known Simulated Annealing [20], and the powerful stochastic evolutionary algorithm, Differential Evolution [21]. The first algorithm was used for online identification of Girona500 as well [17]. Policy gradient approaches can be used as an alternative solution, because they estimate the derivative of the policy with respect to the parameters of the model. The main issue is that the estimation procedure of these approaches is expensive, so derivative-free methods are chosen to be applied in this particular case.

1) Modified Price's Algorithm: Modified Price's (MP) [19] is a global, derivative-free, and iterative black-box optimization algorithm with a great potential in its application to policy search for robotic reinforcement learning tasks. MP is a combination of a global and a local derivative-free method, designed for optimization of the non-linear, multi-modal, and multivariate functions. The global part of the MP algorithm which has been introduced by Brachetti et al. [22] is a population-based method. Recently, Leonetti et al. [19] combined this global search with a deterministic local search. So, the global phase is used to find a neighbourhood of the global minimum, and then the local search explores the neighbourhood to find the global minimum. In this work, the initial population size for the algorithm is set to 20 times the number of the parameters and other parameters are set according to [19].

2) Simulated Annealing: Simulated Annealing (SA) is a probabilistic meta-heuristic that mimics the physical process of annealing, in which a material is heated and then the temperature is slowly lowered to decrease defects, thus minimizing the system energy [20]. The choice of the temperature or cooling scheduling and the next candidate distribution are the most important decisions in the definition of the SA algorithm [23]. Although there are different variants of this algorithm [24], in this paper we use the standard SA algorithm. The initial temperature, re-annealing interval, and temperature function options are set to 100, 100, and *'fast annealing'* scheme respectively.

3) Differential Evolution: The Differential evolution (DE) algorithm, proposed by Storn and Price [21] is simple yet powerful population-based, stochastic, heuristic evolutionary algorithm, which is an efficient and effective global optimizer in the continuous search domain. DE uses a similar crossover, and selection strategies to the genetic algorithm but a stronger mutation strategies. The standard DE algorithm includes 10 different options to define the algorithmic structure (2 crossover schemes and 5 mutation strategies) [21]. According to the classic notation $DE/x/y/z$, the particular variant of DE that is used in this work can be classified as $DE/rand/1/bin$ (where: *rand* specifies the vector to be mutated, 1 is the number of difference vectors used, and *bin* represents the binomial crossover scheme). In this paper, we use the standard implementation of the DE algorithm which can be found in [25]. We set the number of parents Np to 10 times the number of the parameters, and consider weighting factor $F \in [0.8, 0.9]$, and crossover constant $CR \in [0.9, 1]$ according to [26].

E. Online Procedure

In our scenario, when a thruster is deemed faulty, a function $\mathbb{J}$ is created to represent the cost of a path to the target location. The on-board model of the AUV is adapted to the failure conditions (i.e. the isolated thrusters are specified and ignored in the model). The optimization algorithm is then used to compute the optimal policy, in the given policy representation, that takes the AUV as close as possible to the target location using only the functional thrusters. The optimization algorithm computes the optimal policy based on the on-board model of the AUV. The discovered policy Π substitutes the AUV's controller that would work under normal operating conditions. Finally, the learned policy is executed on the real robot in a closed-loop using the state feedback of the AUV. It is also possible to use the target location as a waypoint, by adding a secondary optimization objective (appropriately weighed) to $\mathbb{J}$. As will be seen subsequently, the secondary objective enforces the robot to reach the desired point with a given velocity.

III. Experimental Setup

We performed our experiments on the dynamic model of Girona500 presented in (1), whose parameters have been identified in [17]. All of the experiments, are designed so that the thruster failure occurs in the horizontal plane, while the heave movement of the AUV is always controlled by its original controller. We assume the right surge thruster to be broken, so we turn it off during the failure recovery experiments. In such a case, the Girona500 AUV can only navigate using the left surge and the sway thrusters (the lateral one). Thus the vehicle becomes under-actuated and any attempt to change the allocation matrix $\mathbf{B}$ would be ineffective. We use the following definition of the immediate

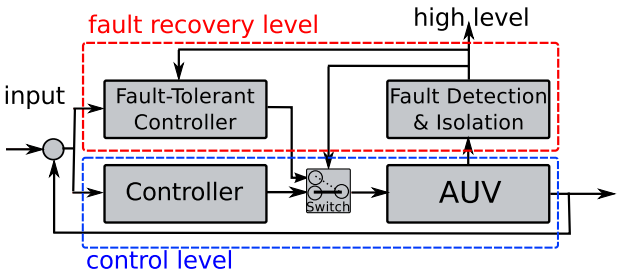


Fig. 2. Control architecture of the AUV including the controller level and the fault recovery level. The green line shows the state feedback used in the state-dependent policy representation (see Section.IV-A and IV-D for more details).

cost:

$$c_t(\langle p_t, v_t \rangle) = \begin{cases} \| p_t - p_d \| & \text{if } t < T \\ w \| v_t - v_d \| & \text{if } t = T \end{cases} \quad (3)$$

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 ($t = T$). For all our experiments we use $T = 60$s, since all the target destinations are reachable in 60 seconds. We also designed the cost function so that when the AUV reaches to an area close enough to the desired position, $\| p_t - p_d \| < 0.2m$, the optimization algorithm is terminated.

IV. Experimental Result

A. Controller Test

A classical control architecture of an AUV includes a position/velocity controller that utilizes the desired inputs and the sensory feedbacks of the vehicle to control the position or velocity of the system. This architecture is illustrated in Fig. 2 and is called the controller level. In order to evaluate the capability of the original controller of Girona500 for thruster failure recovery, a real-world experiment is designed. Firstly, we command the AUV to move $3m$ in the surge direction (x-axis) and record the thruster commands for all 5 thrusters of the robot. Secondly, we turn off the right surge thruster and repeat the same test. The video accompanying this paper includes both experiments and is available online at [27]. In addition, the recorded data is depicted in Fig. 3. It can be seen that in the second test the governing controller of the system tries to use the same configuration of the thrusters was used in the normal situation. And although the right surge thruster is broken, the lateral thruster still remains unused. This experiment shows that the original controller of the system cannot recover the robot from thruster failure, and a failure recovery level (the dashed blue box in Fig. 2) needs to be concatenated to the control level architecture of the AUV (the dashed red box in Fig. 2). Therefore, when the fault detection and isolation module identifies a failure, it sends a message to the higher-level supervisor and, eventually, modifies the fault-tolerant controller and triggers the switch.

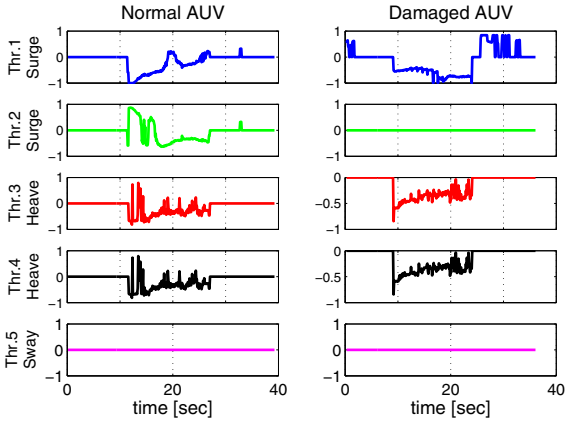


Fig. 3. The recorded thruster commands for a normal AUV (left column) and a damaged AUV (right column - the right surge thruster is broken) while using the original controller scheme without failure recovery level (see Section. IV-A for more details).

B. Constant Policy

In the first experiment, a constant policy (a Fourier expansion of order zero), $\Pi(\chi|\theta) = \theta_{cte}$, is considered, which operates as a set of constant commands, (i.e. voltage), given to the undamaged thrusters. In this case the number of optimization parameters is equal to the number of undamaged thrusters which is 2 in our experiment. Applying the optimum constant policies computed by the three optimization algorithms, the trajectories and velocity profiles can be seen in Fig. 4(a), 4(b). Since all algorithms are stochastic, the results may converge to various solutions in different runs. So we repeated the optimization process 50 times for each algorithm. The box-plots of the statistical results are depicted in Fig. 4(c). The central mark presents the median, the edges of the box are the 25^{th} and 75^{th} percentiles, and the whiskers extend to the most extreme data-points that are not considered as outliers. The result suggests that in this case the Modified Price's algorithm performs better than the other two methods, because it needs less number of function evaluations and converges to better solutions.

C. Time-dependent Policy

In the next experiment, the policy is represented as a linear function approximator which depends only on time t, $\Pi(t|\theta) = \theta^T \Phi(t)$. In this representation θ is the parameter vector and to represent $\Phi(t)$ we employ a 3^{rd} order Fourier basis [18]. In this case the control policy can be more flexible than the constant policy representation in the previous experiment. Also the desired velocity of $\langle 0, 0 \rangle$ becomes more relevant. The number of optimization parameters, which was only 2 in the previous experiment, equals to 8 in this case. As it can be seen in Fig. 4(d), 4(e), the obtained velocity profiles are varied; however, the acquired trajectories are similar. Once again, the optimization process was repeated 50 times for each optimization algorithm. The box-plots of the statistical results are depicted in Fig. 4(f). Increasing the number of optimization parameters, in this case the

differential Evolution algorithm shows better results.

D. State-dependent Policy

In the last policy representation experiment, we close the loop by including feedback from the state variables (i.e. position, orientation, together with linear and angular velocities). In this case, the policy depends on the state variables χ, $\pi(\chi|\theta) = \theta^T\Phi(\chi)$, where θ is the parameter vector. Employing a 3^{rd} order Fourier basis to represent $\Phi(\chi)$, the number of optimization parameters becomes 16 for each thruster. So, for the experiment in 2D plane including 2 undamaged and one broken thrusters, the total number of optimization parameters equals to 32. As it can be seen in Fig. 4(g), 4(h) the acquired velocity profiles are varying but converged towards $\langle 0,0 \rangle$ more smoothly; however, the acquired trajectories are similar. Once again, the optimization process was repeated 50 times for each optimization algorithm. The box-plots of the statistical results are depicted in Fig. 4(i). Also in this case, the DE algorithm shows a better performance in terms of the number of function evaluation and the best value of the objective function.

E. Navigation through Waypoints

In this experiment, two trajectories are generated to reach a point $50m$ far from the current position of the robot. One of the trajectories is a straight line and the other is an arc of circumference. Along each trajectory a waypoint is generated 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 of Girona500. The trajectories and the orientation of the AUV are shown in Fig. 5. 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. This experiment is feasible in the real-world using the state-dependent policy representation and closing the loop. But in fact, either a larger test facility or an onshore tank is required to validate the experiment.

F. Learning Distributions

All the applied optimization algorithms generate the initial population randomly distributed over the upper and lower bounds of the parameter vector. We collect all the different solutions from the 50 runs of the experiments in sections IV-B, IV-C, and IV-D, then fit a normal distribution to each parameter. Then, the uniform distribution in the algorithm are replaced by the estimated means and standard deviations for each parameter. Afterwards, the algorithm generates the initial population from the learned distributions expecting better fitness or better objective values. Using this method, we can collect the best solutions in each experiment and add them to the set of distributions as a new point. The final set of distributions represents the internal dynamics behavior of the model in the presence of failures. In our experiment, as it is depicted Fig. 6, the optimization process

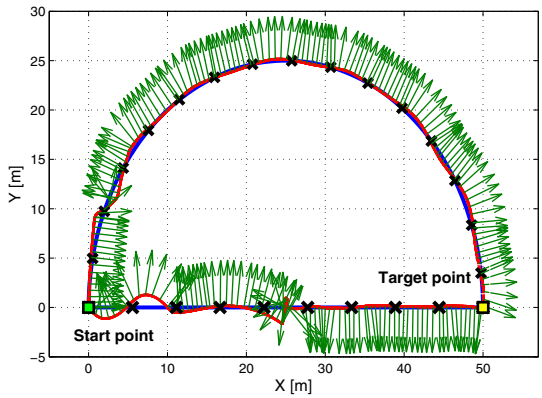


Fig. 5. The desired and result trajectories in the case that the AUV navigates through waypoints. The blue trajectory is the desired and the red is the acquired trajectory. The arrows show the orientation of the AUV (see Section.IV-E for more details).

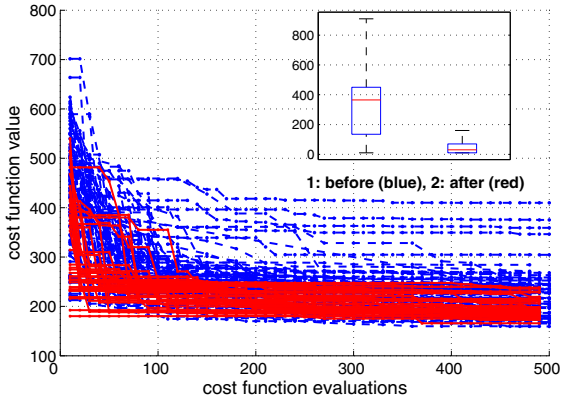
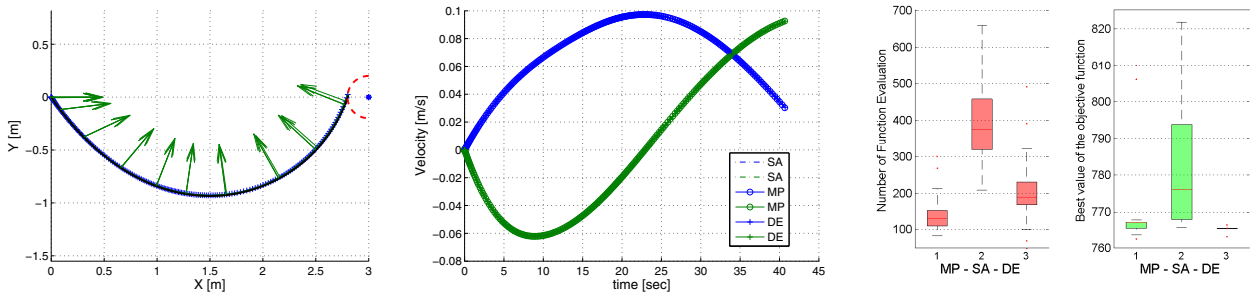


Fig. 6. The comparison before and after learning the parameter distributions. The learning algorithm needs less number of function evaluations to reach the same results after learning the parameter distributions (see Section. IV-F for more details).

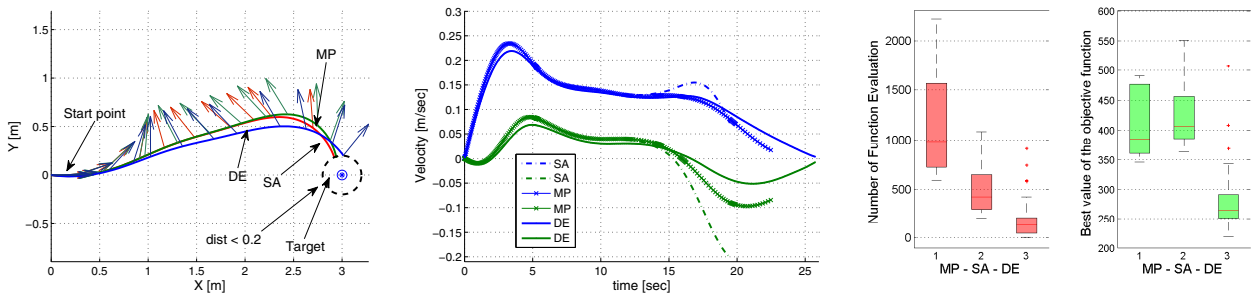
shows better performance while using learned distribution, because it starts from better solutions. Measuring the number of function evaluations in both cases, the algorithm needs 90% less function evaluations when it utilizes the learned distributions. This method not only decreases the online computational cost and makes the approach practically faster, but also employs the dynamical behavior of the system using previously experienced knowledge. This method can be considered as lifelong learning, that will improve the efficiency of the learning algorithm in the long-term, so we do not consider it in the discussion about computational cost of the presented method in the next section.

G. Computational Cost

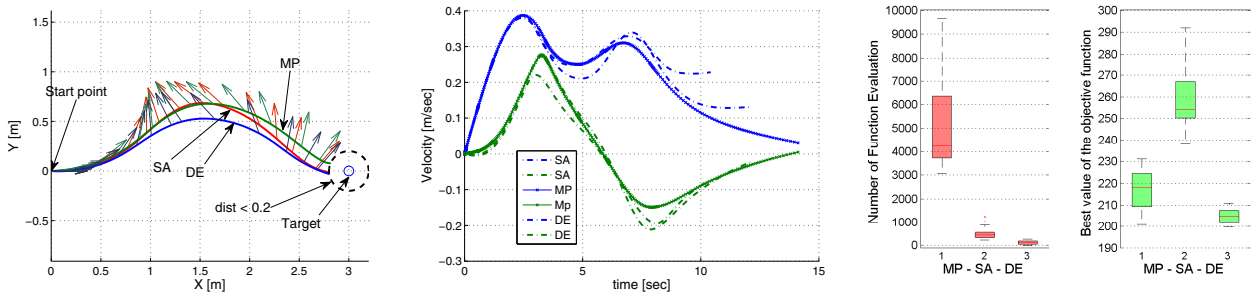
In order to check the feasibility of the presented approach we need to assure that the policy optimization can be performed on-board in a short time. So, we compute the quality of the solution (i.e. distance from the target) over time for a waypoint $5m$ away from the initial position of the AUV. The optimization process was repeated 20 times



(a) Trajectories in 2D plane. (b) Velocity profiles along X and Y axes. (c) Statistical result over 50 runs.



(d) Trajectories in 2D plane. (e) Velocity profiles along X and Y axes. (f) Statistical result over 50 runs.



(g) Trajectories in 2D plane. (h) Velocity profiles along X and Y axes. (i) Statistical result over 50 runs.

Fig. 4. Acquired results for the 1^{st} experiment with constant policy representation (a)-(c), the 2^{nd} experiment with time-dependent policy representation (d)-(f), and the 3^{rd} experiment with state-dependent policy representation (g)-(i) (see section IV-B,IV-C, and IV-D for more details.)

and the average result shows that, it took 12 seconds to find a solution able to take the AUV only $0.5m$ from the target and it can find a good solution in less than 2 minutes. All experiments took place on a single thread on an Intel® Core™ i3-2350M CPU 2.30GHz.

H. Real-world Experiment

In this real-world experiment, we test our approach on Girona500. As it is depicted in Fig. 7, firstly we command the robot to move $3m$ along the surge direction while the original controller of the system is navigating the AUV; the blue trajectory in Fig. 7 shows the result. Secondly, we turn off the right surge thruster and repeat the same experiment. The behavior of the controller is plotted as the red trajectory in Fig. 7. The result shows that the original controller of the system cannot recover the AUV from the failure, and the position error is increasing gradually. Furthermore, we run the simulation using the state-dependent policy representation to find an optimal policy for this thruster failure situation. The simulation result is plotted as the green trajectory in Fig. 7. Finally, the same optimal solution is applied to the real robot and the recorded trajectory is plotted as the black trajectory in Fig. 7. The behavior of the robot is very similar to the simulation. Although the presented approach is using the model of the AUV, the main factors that make the real and simulated data slightly different can be enumerated as: 1) a manipulator arm was attached to the robot during the real-world experiment (for some other purpose), which was not considered neither in the model of the AUV nor in the identification process of the hydrodynamic parameters, 2) unmodeled disturbances from the dynamic environment (e.g. currents, eddies and other sources of noise.) The video accompanying this paper shows the real-world experiments

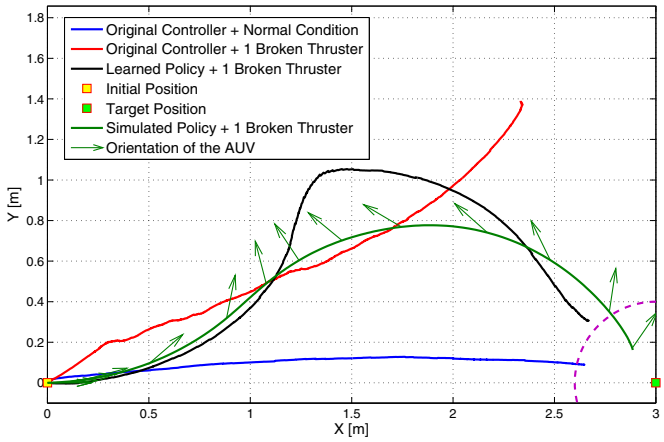


Fig. 7. The trajectories recorded in different scenarios during the real-world experiments (see Section.IV-H for more details).

and is available online at [27].

V. CONCLUSIONS

In this paper a learning-based approach for discovery of new control policies to overcome thruster failures is proposed. One of the advantages of this approach is that it is applicable in both cases which the vehicle becomes under-actuated or remains over-actuated in the presence of the failure. In addition, the approach generates an optimal trajectory that can take the AUV to the target with minimum cost. In most other methods, on the other hand, trajectory generation is ignored in the fault-tolerant control problem. Furthermore, in this work a state-dependent policy is computed online and the fault-tolerant control loop is closed with state feedbacks. So, contrary to many existing methods, we are able to evaluate the resulting controller on the real AUV. Finally, the presented approach can be implemented in multiple types of underwater vehicles, because the theoretical aspect is independent of the choice of the vehicle. As far as a dynamic model of the vehicle and related hydrodynamic parameters are available the approach is applicable.

REFERENCES

[1] 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.

[2] M. Caccia, R. Bono, G. Bruzzone, G. Bruzzone, E. Spirandelli, and G. Veruggio, "Experiences on actuator fault detection, diagnosis and accomodation for rovs."

[3] A. Alessandri, M. Caccia, and G. Veruggio, "A model-based approach to fault diagnosis in unmanned underwater vehicles," in *OCEANS'98 Conference Proceedings*, vol. 2. IEEE, 1998, pp. 825–829.

[4] K. Hamilton, D. Lane, N. Taylor, and K. Brown, "Fault diagnosis on autonomous robotic vehicles with recovery: an integrated heterogeneous-knowledge approach," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 4. IEEE, 2001, pp. 3232–3237.

[5] 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.

[6] T. 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.

[7] T. K. Podder and N. Sarkar, "Fault-tolerant control of an autonomous underwater vehicle under thruster redundancy," *Robotics and Autonomous Systems*, vol. 34, no. 1, pp. 39–52, 2001.

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

[9] 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.

[10] A. S.-f. Cheng and N. E. Leonard, "Fin failure compensation for an unmanned underwater vehicle," in *Proceedings of the 11th International Symposium on Unmanned Untethered Submersible Technology*. Citeseer, 1999.

[11] M. L. Seto, "An agent to optimally re-distribute control in an underactuated auv," *International Journal of Intelligent Defence Support Systems*, vol. 4, no. 1, pp. 3–19, 2011.

[12] M. Andonian, D. Cazzaro, L. Invernizzi, M. Chyba, and S. Grammatico, "Geometric control for autonomous underwater vehicles: overcoming a thruster failure," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 7051–7056.

[13] J.-K. Choi and H. Kondo, "On fault-tolerant control of a hovering auv with four horizontal and two vertical thrusters," in *OCEANS 2010 IEEE-Sydney*. IEEE, 2010, pp. 1–6.

[14] M. Leonetti, S. R. Ahmadzadeh, and P. Kormushev, "On-line learning to recover from thruster failures on autonomous underwater vehicles," in *OCEANS 2013*. IEEE, 2013.

[15] 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.

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

[17] 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.

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

[19] 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.

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

[21] 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.

[22] P. Brachetti, M. D. F. Ciccoli, G. Di Pillo, and S. Lucidi, "A new version of the price's algorithm for global optimization," *Journal of Global Optimization*, vol. 10, no. 2, pp. 165–184, 1997.

[23] M. Miki, T. Hiroyasu, and K. Ono, "Simulated annealing with advanced adaptive neighborhood," in *Second international workshop on Intelligent systems design and application*. Citeseer, 2002, pp. 113–118.

[24] B. Suman and P. Kumar, "A survey of simulated annealing as a tool for single and multiobjective optimization," *Journal of the operational research society*, vol. 57, no. 10, pp. 1143–1160, 2005.

[25] K. V. Price, R. M. Storn, and J. A. Lampinen, "Differential evolution a practical approach to global optimization," 2005.

[26] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," *Advances in intelligent systems, fuzzy systems, evolutionary computation*, vol. 10, pp. 293–298, 2002.

[27] Video, "Video accompanying this paper, available online," http://kormushev.com/goto/ICRA-2014_Reza, 2013.