Robot Learning for Persistent Autonomy

Petar Kormushev and Seyed Reza Ahmadzadeh

Abstract Autonomous robots are not very good at being autonomous. They work well in structured environments, but fail quickly in the real world facing uncertainty and dynamically changing conditions. In this chapter, we describe robot learning approaches that help to elevate robot autonomy to the next level, the so-called 'persistent autonomy'. For a robot to be 'persistently autonomous' means to be able to perform missions over extended time periods (e.g. days or months) in dynamic, uncertain environments without need for human assistance. In particular, persistent autonomy is extremely important for robots in difficult-to-reach environments such as underwater, rescue, and space robotics. There are many facets of persistent autonomy, such as: coping with uncertainty, reacting to changing conditions, disturbance rejection, fault tolerance, energy efficiency and so on. This chapter presents a collection of robot learning approaches that address many of these facets. Experiments with robot manipulators and autonomous underwater vehicles demonstrate the usefulness of these learning approaches in real world scenarios.

1 Persistent Autonomy

While humans and animals can perform effortlessly complicated tasks in unknown environments, our human-built robots are not very good at being similarly independent. Operating in real environments, they easily get stuck, often ask for help, and generally succeed only when attempting simple tasks in well-known situations. We

Petar Kormushev
Dyson School of Design Engineering, Imperial College London, London SW7 2AZ, UK
e-mail: p.kormushev@imperial.ac.uk

S. Reza Ahmadzadeh
iCub Facility, Istituto Italiano di Tecnologia, via Morego 30, 16163, Genoa, Italy
e-mail: reza.ahmadzadeh@iit.it

would like autonomous robots to be much better at being autonomous for longer stretches of time (persistent autonomy), and to be able to carry out more complicated tasks without getting stuck, lost or confused.

Real environments are hard to operate in because they are not completely known, because they change, and because they are complicated. In addition, sensors used to perceive real environments and to self-locate often produce data that are noisy and incomplete. As a result, the effects of actions taken by the robot are not deterministic, but uncertain.

From the moment of birth, humans and animals are good at dealing with such uncertainties. They operate persistently and successfully because they continually observe the effects of their actions, and learn from the outcomes of their attempts to do things. They use these observations to continually change what they know about the world, and then to adapt the ways they move, and to evaluate and perhaps change the strategies, plans and purpose that direct their being.

In this chapter, we describe and evaluate new computational methods that can equip human-built autonomous robots with some of these fundamental capabilities essential for persistent and successful autonomy.

2 Robot Learning Architecture

Before going into specific details about learning methods, it is useful to have a more abstract computational architecture for autonomous robots. Fig. 1 outlines such an architecture designed for development and study of persistent autonomy. A key notion is that the robot's response to changes in the environment takes place at one or a number of hierarchical levels.

Four levels are recognized, each of them taking place at a different time scale. Starting from the smallest timescale (0.001 – 0.1 seconds) to the biggest one (hours–days), these levels are the following: Execution, Operational, Tactical, and Strategic.

The lowest level is the *Execution level*, in which the robot hardware (e.g. actuators) physically execute the commands from the upper levels. The embedded controllers, usually programmed in DSP chips, have the highest control frequency on the order of 1 kHz.

At the *Operational level*, sensor data is processed in Perception to remove noise, extract and track features, localize the robot, in turn providing measurement values for Robust Control of body axes, contact forces/torques and relative positions.

At the *Tactical Level*, a status assessment is being performed using information from around the robot in combination with expectations of planned actions, world model and observed features to determine if actions are proceeding satisfactorily, or have failed. Alongside this, reinforcement and imitation learning techniques are used to train the robot to execute set pre-determined tasks, providing reference values to controllers. Fed by measurement values from Perception, they update controller reference values when disturbance or poor control causes action failure.

Finally, at the *Strategic level*, sensor features and state information are matched with geometric data about the environment to update a geometric world model. These updates include making semantic assertions about the task, and the world geometry, and using reasoning to propagate the implications of these through the world description. The Planning uses both semantic and geometric information as pre-conditions on possible actions or action sequences that can be executed. When State Estimation detects failure of an action, the Planner instigates possibilities for a plan repair.

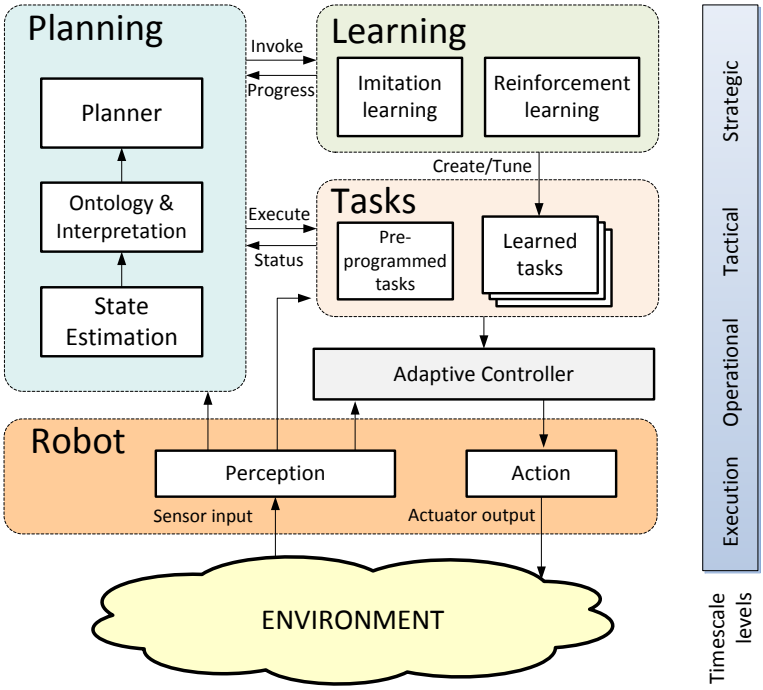


Fig. 1: A high-level diagram illustrating an example architecture for robot learning. It shows how the learning can be integrated with the robot controller and planner. The purpose of this computational architecture is to develop and study persistent autonomy.

3 Learning of Reactive Behavior

One of the most challenging tasks for autonomous robots is the autonomous manipulation of objects in unstructured environment. This is difficult due to the presence of active disturbances and uncertainties at many levels. A typical example for such a task is the autonomous robotics valve turning task. It is not surprising that this task is also included in the DARPA Robotics Challenge – an event where the best humanoid robots in the world compete for successful completion of a series of challenging tasks. In this section we explain how robot learning can be used to learn reactive behaviors, and in particular how this can be applied to the valve turning task.

3.1 Autonomous Robotic Valve Turning

Autonomous robotic valve turning is a challenging task specially in unstructured environments with increased level of uncertainty (e.g. in disaster response setting, or in underwater or aerial applications). The existing disturbances in the environment or the noise in the sensors can endanger both the robot and the valve during the operation. For instance, the vision system may be occluded and thus introduce a delay in updating the data, or even providing the system with wrong information. Exerting huge forces/torques on the valve by the robot, is another hazardous and highly probable situation. In such cases an autonomous system that is capable of observing the current state of the system and reacting accordingly, can help to accomplish the mission successfully even in the presence of noise.

The learning approach described here is very helpful for coping with the challenges of autonomous robotic valve turning in the presence of active disturbances and uncertainties. The valve turning task comprises two phases: reaching and turning. For the reaching phase the manipulator learns how to generate trajectories to reach or retract from the target. The learning is based on a set of trajectories demonstrated in advance by the operator. The turning phase is accomplished using a hybrid force/motion control strategy. Furthermore, a reactive decision making system is devised to react to the disturbances and uncertainties arising during the valve turning process. The reactive controller monitors the changes in force, movement of the arm with respect to the valve, and changes in the distance to the target. Observing the uncertainties, the reactive system modulates the valve turning task by changing the direction and rate of the movement. A real-world experiment with a robot manipulator mounted on a movable base shows the efficiency and validity of this learning approach. The experimental setup for these experiments is shown in Fig. 2. One of the most interesting applications of the described learning methods is for accomplishing the autonomous robotic valve manipulation in underwater environment which is one of the goals of the PANDORA project (Lane et al, 2012; PANDORA, 2012).

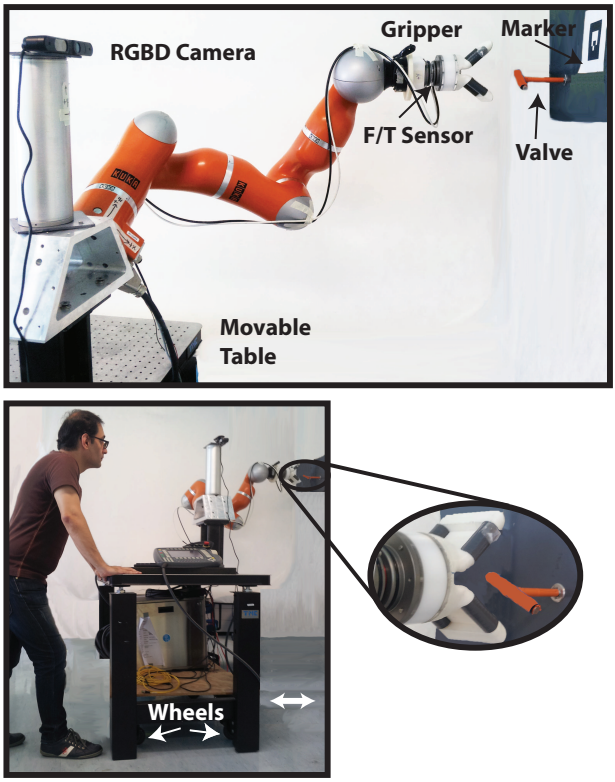


Fig. 2: The experimental set-up for the valve turning task. The valve is detected and localized using an RGB-D sensor through an AR-marker. The manipulator is equipped with a gripper and is mounted on a movable (wheeled) table. During the execution of the task, a human can create random disturbances by moving the base of the robot.

3.2 Related Work

Robotic valve manipulation contains a number of complex and challenging subtasks. There seem to be few published description of attempts directly related to this task. Prior works in industrial robotic valve operation, generally use nonadaptive classical control and basic trajectory planning methods. In (Abidi et al, 1991), Abidi *et al.,* tried to achieve inspection and manipulation capabilities in the semi-autonomous operation of a control panel in a nuclear power plant. A 6-DoF industrial robot equipped with a number of sensors (e.g., vision, range, sound, proximity, force/torque, and touch) was used. The main drawback is that their approach is developed for static environments with predefined dimensions and scales. For instance, the size and position of the panel, the valve, and other objects in the

room are manually engineered into the system. More recent approaches generally use sensor-based movement methods which implies that the robot trajectories have not been programmed off-line. In (Anisi et al, 2011), the robot is equipped with a torque sensor and the valve which is equipped with a proximity sensor is detected using a vision sensor. The authors focus on a model-based approach to avoid over-tightening/loosening of the valve. The other phases of the valve manipulation process are accomplished using classical methods. In another publication (Anisi et al, 2012) the authors develop a valve manipulation system for an outdoor environment. The vision sensor is replaced with a thermal camera, and the (round) valve is replaced with a T-bar valve, which is easier for the robot to manipulate. The main focus of (Anisi et al, 2012) is detecting the valve and avoiding the over-tightening/loosening of the valve in an early stage using a model-based technique.

Other groups have also investigated valve turning. In (Orsag et al, 2014) a framework for valve turning is proposed using a dual-arm arial manipulator system. The framework is built based on teleoperation and employs motion detection, voice control and joystick inputs. A user-guided manipulation framework is proposed in (Alunni et al, 2013). Although the planning algorithm generates the robot motions autonomously, the search process and the object detection phase are accomplished by a human operator and the result is passed to the robot. A dual-arm impedance hierarchical controller is devised in (Ajoudani et al, 2014) that employs the upper body kinematics and dynamics of a humanoid robot for reaching and turning a valve.

3.3 Hierarchical Learning Architecture

Here we describe a hierarchical learning architecture with three different layers which are illustrated at a high level in Fig. 3. Each layer realizes specific subtasks to improve the persistent autonomy of the system. The lowest layer is responsible for evaluating demonstrations and generating smooth trajectories using learning methods. In this layer an integrated approach is used which allows the robot-arm to obtain new motor skills by kinesthetic teaching. Imitation learning (Kormushev et al, 2011) is used for training the manipulator to learn a positional profile. An early implementation of this approach for valve turning can be found in (Carrera et al, 2012).

The middle layer is responsible for evaluating relative movements and supervising the subordinate layer. Observing the feedback from the Optitrack sensor, this upper layer provides prior decisions depending on the relative behavior of the valve which affects the dynamics of the system. A reactive fuzzy system, called RFDM (Reactive Fuzzy Decision Maker), is established for producing proper decisions based on linguistic rules. The RFDM reacts to the relative movement between the AUV and the valve dynamically and alters the generated trajectory in the lower layer accordingly. The highest layer, is responsible for tuning the parameters of the RFDM system using the expert knowledge. Four various local and global optimization algorithms are implemented to find the best optimum solution.

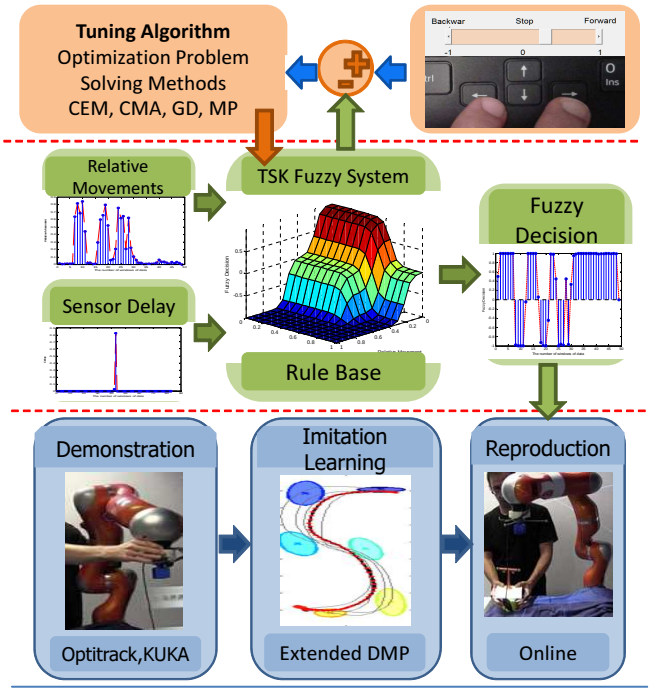


Fig. 3: A high-level diagram illustrating the three layers of the proposed hierarchical learning approach.

3.4 Learning Methodology

The valve turning task comprises two main phases: reaching and turning. First, the robot has to learn how to reach the valve. Imitation learning approach which is designed specially to learn trajectory-based tasks, is a promising choice to learn the reaching skill (Kormushev et al, 2011; Schaal et al, 2003). In order to reproduce the reaching skill towards the target, the robot utilizes feedback from the RGB-D sensor to determine the position and orientation of the valve.

When the robot is able to reproduce the reaching skill a hybrid force/motion control strategy handles the turning phase. Hybrid force/motion control is a well-established method (Raibert and Craig, 1981; Khatib, 1987; Yoshikawa and Zheng, 1993). Using such hybrid strategy, the force controller can maintain the contact between the valve and the gripper while the motion controller turns the valve. The hybrid force/motion controller utilizes feedback from a Force/Torque (F/T) sensor mounted between the end-effector and the gripper. Subsequent to the turning phase, the robot employs the reaching skill in reverse to retract from the valve.

In order to develop an autonomous system, the robot needs to deal with uncertainties. To emulate the uncertainties in our experiments, we manually apply disturbances to the system. The disturbances during the execution of the task are monitored and handled by a Reactive Fuzzy Decision Maker (RFDM). Although such reactive system can be implemented using a thresholding method, the fuzzy system is chosen. The reason is that the fuzzy system provides a continuous decision surface and it infers from a set of human-defined linguistic rules. The RFDM module, monitors the position of the gripper and the valve together with the magnitude of the forces and torques applied to the end-effector from the valve. Using this information, RFDM generates decisions that regulate the movements of the robot during the process. For example, RFDM halts the process when the magnitude of the force increases due to an undesired movement. In addition, RFDM also controls the rate of the motion. For instance, when there is no external disturbance, the robot can reach the valve faster.

As depicted in Fig. 2 the experimental setup for all the conducted experiments consists of a 7-DoF KUKA-LWR manipulator mounted on a movable (wheeled) table, a (T-bar shaped) mock-up valve mounted on the wall in the robot's workspace, a gripper designed for grasping and turning the valve, an ATI Mini45 Force/Torque (F/T) sensor which is sandwiched between the gripper and the robots end-effector, and an ASUS Xtion RGB-D sensor for detecting and localizing the valve.

Fig. 4 illustrates a flow diagram of the proposed approach. The RGB-D sensor detects the pose of the valve which is used by the reaching module and RFDM. The F/T sensor monitors the force/torque applied to the gripper, which is used by the turning module and RFDM. Observing the inputs provided by the sensors, RFDM generates proper decisions in order to modulate the behavior of the robot during the process. The RFDM system is tuned by collecting data from a human expert using optimization techniques.

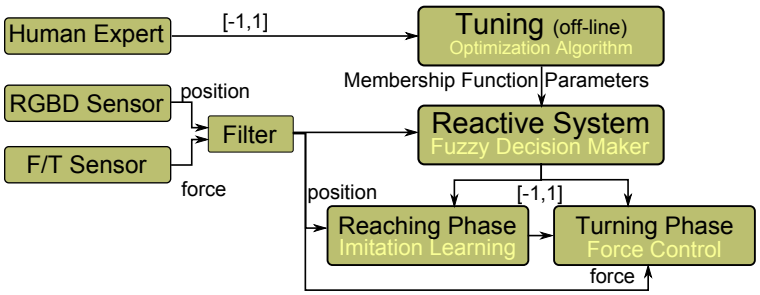


Fig. 4: A high-level flow diagram illustrating the different components of the proposed approach.

3.5 Imitation Learning

Imitation learning enables manipulators to learn and reproduce trajectory-based skills from a set of demonstrations (Schaal et al, 2003). The demonstrations are provided either by teleoperation or through kinesthetic teaching. One of the most widely-used representations for trajectory-based skills is Dynamical Movement Primitives (DMP) (Ijspeert et al, 2013). DMP allows to learn a compact representation of the reaching skill using the recorded demonstrations. In this section, we use the extended DMP approach proposed in (Kormushev et al, 2011) which also encapsulates variation and correlation information of the demonstrated skill as a mixture of dynamical systems. In order to reach a target, in this approach a set of virtual attractors is utilized. The influence of these attractors is smoothly switched along the movement on a time basis. A proportional-derivative controller is used to move the end-effector towards the target. In contrast to the original DMP, a full stiffness matrix associated with each primitives is considered. This allows to capture the variability and correlation information along the movement. The set of attractors is learned through weighted least-square regression, by using the residual errors as covariance information to estimate stiffness gain matrices.

During the demonstration phase, multiple desired trajectories are demonstrated by a human operator through kinesthetic teaching. Each demonstration $m \in \{1,\dots,M\}$ consists of a set of T_m positions x, velocities $\dot{x}$, and accelerations $\ddot{x}$, of the end-effector in Cartesian space where $x \in \mathbb{R}^3$. A dataset is formed by concatenating the $P = \sum_{m=1}^{M} T_m$ data points. A desired acceleration is computed based on a mixture of L proportional-derivative systems as follows:

$$\hat{\ddot{x}} = \sum_{i=1}^{L} h_i(t)[K_i^P(\mu_i^x - x) - k^v\dot{x}], \tag{1}$$

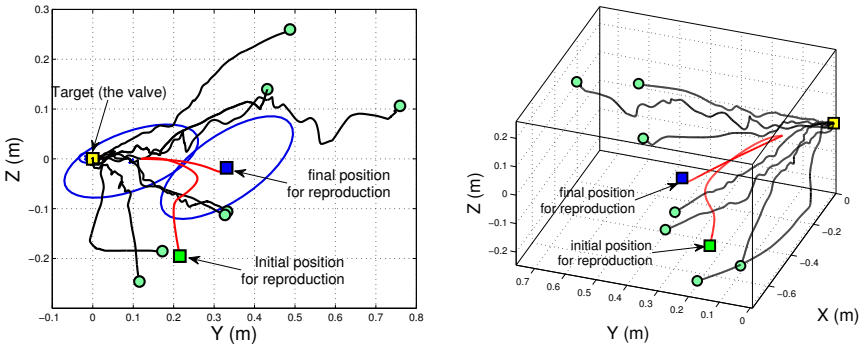
where $\hat{\ddot{x}}$ is the desired acceleration, K_i^P are the stiffness matrices, μ_i^x are the centers of the attractors in Cartesian space, $h_i(t)$ are the weighting functions, and k^v is the derivative gain.

By following the weighting functions $h_i(t)$, the system converges sequentially over time to the ordered sequence of attractors. Stiffness matrices K_i^P and the centers μ_i^x are learned from the observed data using weighted least-square regression. In the reproduction phase the system uses the learned weights and set of attractors to reproduce a trajectory to reach the target.

The recorded set of demonstrations is depicted as black curves in Fig. 5. Following the described approach, the system learns a set of attractors which can be seen in the 2D plots in Figures 5a and 5b as blue ellipsoids. Using the learned set of attractors the robot is able to reproduce a new trajectory from an arbitrary initial position towards the target. The red trajectory in Fig. 5 illustrates a reproduction. The goal, i.e. the valve in this experiment, is shown in yellow. A snapshot of the reaching skill reproduced by the robot is shown in Fig. 6.

In this section, we introduce a new capability based on the implicit timing: a reversible behavior of the system. This new capability enables the robot to perform the following: (i) reactive behavior by switching the direction of the movement towards the target or away from it; (ii) after the task is finished the robot uses this capability to retract the arm. The advantage of the presented capability is that by learning just the reaching skill the robot is capable of reproducing multiple behaviors including reaching and retracting, and switching between them. This can be achieved by changing the timing equation from $t = -ln(s)/\alpha$ to $t = t_{final} + ln(s)/\alpha$.

Fig. 5 illustrates a reproduction from an arbitrary initial position towards the target. It can be seen that, in the middle of the movement the robot reverses the motion and moves backwards. It has to be commented that, by executing the reverse motion, the robot goes back to the center of the first attractor.



(a) Trajectories and learned attractors in 2D plane.

(b) Trajectories in 3D space.

Fig. 5: The recorded trajectories that form the set of demonstrations (black), and the reproduced trajectory from an arbitrary initial position (red) are illustrated. The robot retracts from the middle of the path by receiving a command from RFDM.

3.6 Force/Motion Control Strategy

Once the robot learns the reaching skill, the turning phase begins. In this phase, the goal of the robot is to turn the valve (by 180° from its initial configuration) while maintaining the position of the gripper. To control the forces and torques applied to the end-effector, a hybrid force/motion control approach is used (Raibert and Craig, 1981; Khatib, 1987; Yoshikawa and Zheng, 1993). Hybrid force/motion controller is preferred to be used in this application because during the turning phase a zero force controller can reduce the undesired forces and torques.

The proposed hybrid strategy is designed for 6-axes full space control. Forces and torques are controlled in the 5-axes while motion is controlled around the z-axis in order to turn the valve. The assigned coordinate system is depicted in Fig. 2 which is set with respect to the initial pose of the gripper. The z-axis (surge and roll) is normal to the end-effector's palm. The y-axis (sway and pitch) is perpendicular to the z-axis and pointing sideways. And the x-axis (heave and yaw) is perpendicular to the $z-y$ plane (Das and Das, 2004). A desired normal force is set along the z-axis in order to maintain the gripper in contact with the valve. Zero forces and torques are specified along the x- and y-axes. The zero desired values of the forces and torques are designed to lessen the reactionary forces and torques along (around) the axes during the valve turning process.

The hybrid force/motion controller is suitable for autonomous mobile manipulation (Jamisola et al, 2005). In underwater environment the valve turning task is more difficult due to the highly unstructured and uncertain environment. Also, the valve can be rusty and sensitive to high forces/torques. The forces and torques are specified as follows:

$$\begin{aligned} \mathbf{F}_{con} &= \mathbf{F}_{des} + \mathbf{k}_p(\mathbf{F}_{des} - \mathbf{F}_{act}) \\ \mathbf{T}_{con} &= \mathbf{T}_{des} + \mathbf{k}_p(\mathbf{T}_{des} - \mathbf{T}_{act}) \end{aligned} \tag{2}$$

where $\mathbf{F}$ and $\mathbf{T}$ denote forces and torques respectively, and subscripts *des*, *act*, and *con* denote the desired, actual, and control parameters respectively. In the following sections, some experimental results are explained.

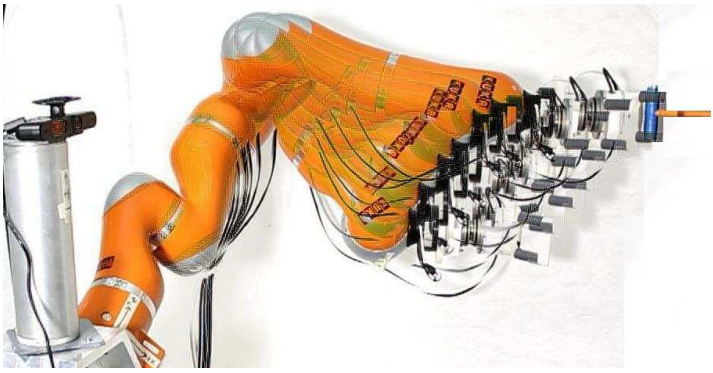


Fig. 6: The robot reaching the valve during the reproduction phase.

3.7 Learning of Reactive Behavior using RFDM

In robotic valve turning in the real world, a sudden movement of the arm can endanger both the valve and the manipulator. Also, if the robot exerts huge and un-

controlled amount of force/torque during the turning phase, it may break the valve off. In order to prevent such behaviors and developing a more autonomous and reliable system, a reactive decision maker system is designed. This system, which is a Reactive Fuzzy Decision Maker (RFDM), evaluates the dynamic behavior of the system and regulates the robot's movements reactively. We chose fuzzy systems because they are based on linguistic rules and the parameters that specify the membership functions have clear physical meanings. Also, there are methods to choose good initial values for the parameters of a fuzzy system (Wang, 1999). The RFDM system monitors the relative movement between the valve and the end-effector and generates decisions according to the defined linguistic rules. More details about the design of this RFDM system can be found in (Ahmadzadeh et al, 2013a).

The RFDM described here comprises two additional inputs. One is the distance between the gripper and the valve. This extra information gives the RFDM the capability to behave more adaptively. For instance, when the gripper is about to grasp the valve, the new RFDM generates more informed decisions and increases the sensitivity of the robot's movements with respect to the disturbances. The other input is the force/torque values applied to the gripper and reacts to the uncertainties. For instance, RFDM retracts the arm when it observes a sudden increase in force/torque during the turning phase. The inputs for RFDM is provided by RGB-D sensor that works at 30 fps and F/T sensor with 1 ms sampling-time.

Design of the Fuzzy System

The proposed fuzzy system comprises three inputs: *a*) the distance between the gripper and the valve (the norm of the distance vector); and *b*) the relative movement between the valve and the gripper (in $x-y$ plane); *c*) the forces and torques applied to the valve from the gripper.

All the inputs are first normalized in range $[0,1]$ and then are sent to the RFDM system. The third input is provided by the F/T sensor which has a sampling interval equal to 1 *ms*. The output of the sensor consists of three force and three torque elements. In this case, the torque is multiplied by a factor to be numerically comparable to the value of the force. The normalizing equation is as follows:

$$\gamma = \frac{\|\mathbf{F}\| + \beta\|\mathbf{T}\|}{F_{max}} \tag{3}$$

where $\gamma \in [0,1]$, $\beta = 10$ is a constant factor used to level-off the range of values between the forces and the torques, and $F_{max} = 30\ N$ is set as the maximum threshold. The values of these parameters are tuned using expert knowledge and taking into consideration various constraints such as the actuator saturation thresholds.

Monitoring the relative movement between the valve and the gripper, the system can detect oscillations with different amplitudes and frequencies. For instance, if the end-effector is reaching the valve, and the system senses an oscillation with say *Medium* amplitude the fuzzy system reacts to that by halting the arm. To simulate

such behavior in the experiments, the operator manually moves the table of the robot back and forth. Moreover, considering the distance between the gripper and the valve, the system can change its behavior adaptively. For example, if the gripper is *Far* from the valve, even in the presence of a disturbance, the robot still moves towards the valve. On the other hand, if the gripper is in the vicinity of the valve the robot reacts to smaller oscillations and waits or even retracts the arm. Furthermore, measuring the force/torque magnitudes applied to the gripper, generated by colliding either to the valve or other objects, the system reacts according to the defined rules.

The output of the RFDM system is the reactive decision which is a real number in range $[-1,1]$. The sign of the output specifies the direction of the movement (i.e., $+$ for going forward and $-$ for going backward). For instance, -1 means to retract with 100% speed, 0 means to stop, and 1 means to approach with 100% speed. Therefore, the RFDM system not only decides the direction of the movement, but also specifies the rate of the movement.

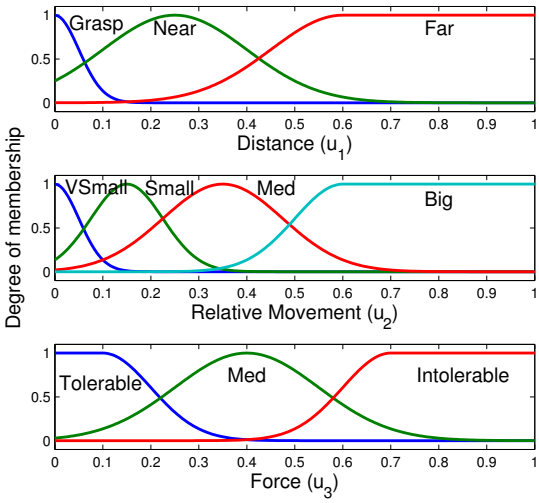


Fig. 7: Fuzzy membership functions defined for each input.

In order to design the fuzzy system, we consider the inputs to be $\mathbf{u} = [u_1, u_2, u_3]^T$ and the output as r. Firstly, $N_i (i = 1,2,3)$ fuzzy sets, $A_i^1, A_i^2, ..., A_i^{N_i}$, are defined in range $[0,1]$, which are normal, consistent, and complete with Gaussian membership functions $\mu_{A_i^1}, \mu_{A_i^2}, ..., \mu_{A_i^{N_i}}$. Then, we form $N_{rule} = N_1 \times N_2 \times N_3$ $(3 \times 4 \times 3 = 36)$ fuzzy $IF-THEN$ rules as follows:

$$IF\ u_1\ is\ A_1^{i_1}\ and\ u_2\ is\ A_2^{i_2}\ and\ u_3\ is\ A_3^{i_3}\ THEN\ y\ is\ B^{i_1 i_2 i_3} \tag{4}$$

Moreover, 7 constant membership function in range $[-1,1]$ are set for the output. The fuzzy membership functions defined for each input are shown in Fig. 7. Finally, the TSK fuzzy system is constructed using product inference engine, singleton fuzzifier, and center average defuzzifier (Wang, 1999):

$$r = \frac{\sum_{i_1=1}^{N_1}\sum_{i_2=1}^{N_2}\sum_{i_3=1}^{N_3} y^{i_1 i_2 i_3}\mu_{A_1}^{i_1}(u_1)\mu_{A_2}^{i_2}(u_2)\mu_{A_3}^{i_3}(u_3)}{\sum_{i_1=1}^{N_1}\sum_{i_2=1}^{N_2}\sum_{i_3=1}^{N_3} \mu_{A_1}^{i_1}(u_1)\mu_{A_2}^{i_2}(u_2)\mu_{A_3}^{i_3}(u_3)} \tag{5}$$

Since the fuzzy sets are complete, the fuzzy system is well-defined and its denominator is always non-zero. The designed fuzzy system cannot be illustrated in a single 3D plot because it consists of three inputs and one output. We plotted the fuzzy surface for input variables u_2 and u_3 over a single value of the variable u_1. So each surface in Fig. 8 is related to a fixed value of u_1. It can be seen from Fig. 8 that RFDM shows more sensitive and cautious behaviors as the distance to the valve decreases.

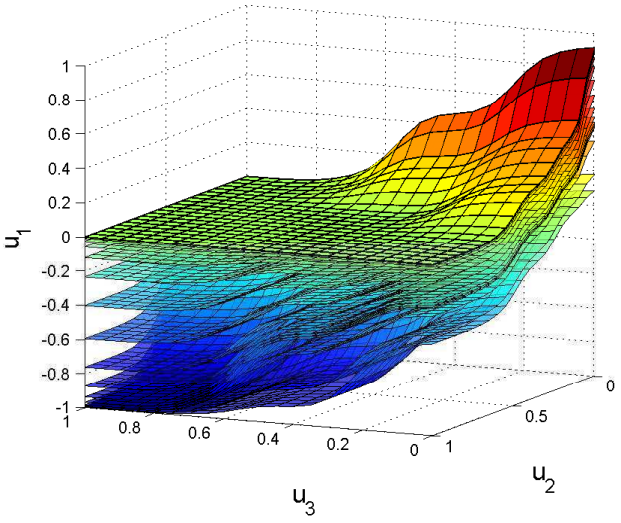


Fig. 8: Fuzzy inference system surface including three inputs (u_1, u_2, u_3). The input specifying the distance between the robot and the valve u_1 affects the sensitivity of the designed fuzzy system according to the distance from the valve. Each surface shows a fixed value of the u_1 input for the whole range of the u_2 and u_3 inputs.

Tuning the Fuzzy System

In order to tune the parameters of the devised fuzzy system, the subconscious knowledge of a human expert is derived. In this case, the human expert knows what to do but cannot express exactly in words how to do it. In order to extract the subconscious knowledge of the human expert, a tutor simulates the effect of the disturbances (e.g., underwater currents) by moving the wheeled table, while the robot tries to reach and turn the valve. Simultaneously, using a slider button, another tutor regulates the movements of the manipulator while it is following the reproduced trajectory or turning the valve. The tutor applies appropriate continuous commands in range $[-1, 1]$, to the system, where -1 means go backward along the trajectory with 100% speed and 1 means go forward along the trajectory with 100% speed. For instance, when the base of the robot is being oscillated say with a *Big* amplitude, the tutor smoothly moves the slider backwards to retract the arm and prevent it from any collision with the valve or the panel. All data, including the position of gripper and the valve, and the tutor's commands are recorded during the learning process. The recorded data is then used to tune the RFDM in off-line mode.

The error between the recorded data from the tutor, which is a fuzzy surface, and the output of the un-tuned fuzzy system which is also a fuzzy surface, is used to make an objective function. The objective function can be minimized using various optimization algorithms. More details about the implementation can be found in (Ahmadzadeh et al, 2013a).

3.8 Iterative Learning Control

Iterative Learning Control (ILC) is an alternative method for improving tracking in repetitive control scenarios (Moore, 2012; Bristow et al, 2006). ILC differs from other learning-type control strategies, such as adaptive control and neural networks. Adaptive control strategies modify the controller, which is a system, whereas ILC modifies the control input, which is a signal.

The goal of ILC is to generate a feedforward control that tracks a specific reference or rejects a repeating disturbance. ILC has several advantages over a well-designed feedback and feedforward controller. Foremost is that a feedback controller reacts to inputs and disturbances and, therefore, always has a lag in transient tracking. Feedforward control can eliminate this lag, but only for known or measurable signals, such as the reference. Therefore, ILC cannot directly be applied for disturbance rejection because disturbances are typically not repetitive.

4 Learning to Recover from Failures

Fault tolerance is the capability of a robot to complete a mission despite the failure of one or more subsystems. It is also referred to as fault control, fault accommodation or control reconfiguration. The capability to recover from failures is extremely important for autonomous robots that operate in harsh environments or difficult-to-reach places for humans, such as underwater or in outer space. In this section we describe learning methods for improving the fault-tolerance of autonomous robots in order to increase their reliability and persistent autonomy. To be more specific, we will focus on a particular application for Autonomous Underwater Vehicles (AUVs). We describe a learning-based approach that is able to discover new control policies to overcome thruster failures of AUVs as they happen.

Persistent Autonomy or operating over long missions without any human assistance, is one of the most challenging goals for underwater robotics. AUVs are supposed to deal with extreme uncertainties in unstructured environments, where a failure can endanger both the vehicle and the mission. A fault-tolerant strategy enables the system to continue its intended operation, possibly at a reduced level, rather than failing completely.

Usually, a fault-tolerance strategy consists of three steps: fault detection, fault isolation, and fault tolerance. Fault detection is the process of monitoring a system to recognize the presence of a failure. Fault isolation or diagnosis is the capability to determine which specific subsystem is subject to failure. Both topics have been extensively investigated in the literature and have several effective solutions (Caccia et al, 2001; Alessandri et al, 1998; Hamilton et al, 2001; Antonelli, 2003).

Although the failure can happen in any subsystem of an AUV, here we focus on the case of a thruster failure. Thruster blocking, rotor failure, and flooded thrusters are some of the factors that can lead to a thruster failure in real missions (Caccia et al, 2001). After the failure is detected and isolated a fault-tolerant strategy must be considered to rescue the vehicle safely.

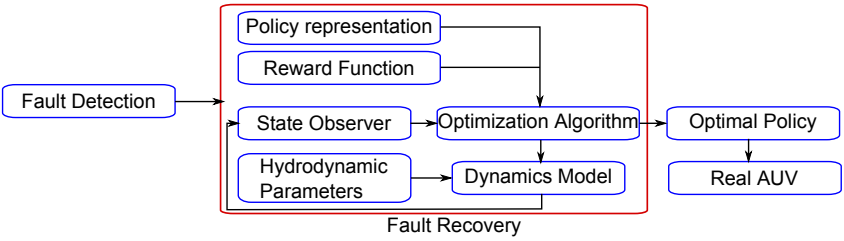


Fig. 9: The diagram shows the fault detection and fault recovery modules. The fault recovery module includes a number of elements such as policy representation, reward function and dynamics model of the system.

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 (Alessandri et al, 1998; Caccia et al, 2001; Podder et al, 2000; Podder and Sarkar, 2001). 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 (Antonelli, 2006). A few works are targeted at AUV controlled with surfaces (Perrault and Nahon, 1998; Cheng and Leonard, 1999; Seto, 2011). Those methods are specific to the kinematics and dynamics models of the considered AUV.

The methodology described here, on the other hand, makes use of the AUV model for simulation, but not in the derivation of the controller, which is of a predefined 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.

4.1 Methodology

As can be seen in Fig. 9 when a thruster is deemed faulty, the fault detection module sends a signal to the fault recovery module. This module's task is to discover a fault-tolerance control policy using the remaining assets of the system. The discovered control policy have to be able to safely bring the AUV to a station where it can be rescued.

The proposed fault recovery module is framed in the context of model-based direct policy search for reinforcement learning. This framework comprises a dynamic model of the vehicle, a parameterized representation for the control policy, a reward function, and an optimization algorithm. The dynamics model of the system is reconfigured according to the current situation of the system. In the employed model-based policy search approach the trials are performed on the on-board dynamic model and not directly by the vehicle. For AUVs this is not a practical limitation, as their dynamics have been modeled accurately.

The direct policy search utilizes a function approximation technique and an optimization heuristic to learn an optimal policy that can reach the goal specified by the reward function. The optimization heuristic can be treated as a black-box method because in policy search over a finite horizon, the particular path followed by the agent in the state-space can be ignored. In this section, all the components of the fault recovery module depicted in Fig. 9 are explained. Further details about the implementation and real-world experiments with this method can be found in (Ahmadzadeh et al, 2014a).

4.2 Fault Detection Module

The process of monitoring a system in order to recognize the presence of a failure is called fault detection. We only consider the case of thruster failure which can take place due to thruster blocking, rotor failure, flooded thrusters, etc. In a real underwater vehicle sometimes the thruster may still work but not as a fully functional module. For instance, some sea plants may twist around the propeller of the thruster and reduce its efficiency by a percentage. In this section, we consider a generic case in which a thruster can be fully functional, partially broken or totally nonfunctional.

Failure detection in AUVs and ROVs has been extensively studied before (Caccia et al, 2001; Hamilton et al, 2001; Antonelli, 2003). Therefore, we assume that a fault detection module is available and placed in a higher layer of the system architecture (see Fig. 9). This module continuously monitors all the thrusters and sends information about their coefficient of functionality (healthiness) to update the other modules. The output of this module is a vector of functionality coefficients in range $[0, 1]$, where 0 indicates a totally nonfunctional thruster, 1 represents a fully functional thruster, and for instance, 0.7 indicates a thruster with 70% efficiency.

4.3 Problem Formulation

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. The AUV we use for our experiments is Girona500 (Ribas et al, 2012) which is used in the PANDORA project (Lane et al, 2012). 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.

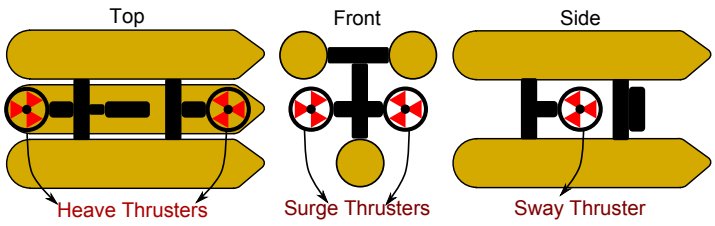


Fig. 10: A model of the Girona500 AUV equipped with 5 thrusters arranged in a particular layout as shown. In the conducted failure recovery experiments, one of the surge thrusters is broken.

4.4 Learning Methodology

We frame our approach as model-based direct policy search reinforcement learning for discovering fault-tolerant control policies to overcome thruster failures in AUVs. The described approach learns on an on-board simulated model of the AUV.

In previous research (Ahmadzadeh et al, 2014b, 2013b; Leonetti et al, 2013) fault-tolerant control policies have been discovered considering the assumption that the failure makes the thruster totally broken, meaning that a faulty thruster is equivalent to a thruster which is turned off. One of the pros of this approach is taking advantage of the remaining functionality of a partially broken thruster. Therefore, this method can deal with partially broken thrusters and use them to reach the desired goal.

The framework comprises a dynamic model of the vehicle (6), a parameterized representation for the control policy, a cost function, and an optimization algorithm, as described in the following sections.

AUV Model

A dynamic model of an AUV is formed using a set of equations and a set of parameters. The obtained model is then used to find the optimal solutions that are executed on the robot later. The dynamics equations of a 6-DoF rigid body subject to external forces and torques while moving in a fluid environment can be generally formulated as follows:

$$\begin{aligned} \dot{\eta} &= \mathbf{J}(\eta)\,\nu \\ (\mathbf{M}_{RB}+\mathbf{M}_A)\,\dot{\nu} + (\mathbf{C}_{RB}(\nu)+\mathbf{C}_A(\nu)+\mathbf{D}(\nu))\,\nu + \mathbf{g}(\eta) &= \mathbf{B}\tau, \end{aligned} \tag{6}$$

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, $\mathbf{M}_A$ is the hydrodynamic added mass matrix, $\mathbf{C}_{RB}(\nu)$ is the rigid body Coriolis and centripetal matrix, $\mathbf{C}_A(\nu)$ is the added mass Coriolis and centripetal matrix, $\mathbf{D}(\nu)$ is the hydrodynamic damping matrix, $\mathbf{g}(\eta)$ is the hydrostatic restoring force vector, $\mathbf{B}$ is the actuator configuration matrix, and the vector τ is the control input vector or command vector.

In our experiments we use Girona500 (Ribas et al, 2012) which is a reconfigurable AUV equipped with typical navigation sensors (e.g. Doppler Velocity Log Sensor), survey equipments (e.g. stereo camera) and various thruster layouts. As depicted in Fig. 10, the selected thruster layout in this work consists of five thrusters: 2 in heave direction, 2 in surge direction, and 1 in sway direction. In order to build a model of the system for simulating the behaviors of the AUV, the hydrodynamic parameters of Girona500, are substitute in the dynamics equations of the AUV (6).

The hydrodynamic parameters are extracted using an online identification method and are reported in (Karras et al, 2013).

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 (Konidaris et al, 2011). 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. More details about the function approximation using Fourier basis can be found in (Konidaris et al, 2011).

Cost Function

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

$$\mathbb{J}(\theta) = \sum_{t=0}^{T} c_t(\eta_t)\Big|_{\Pi(\chi|\theta)} \tag{7}$$

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.

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 called Modified Price's (MP) al-

gorithm (Leonetti et al, 2012), the well-known Simulated Annealing (Kirkpatrick et al, 1983), and the powerful stochastic evolutionary algorithm, Differential Evolution (Storn and Price, 1997). The first algorithm was used for online identification of Girona500 as well (Karras et al, 2013). 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.

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.

4.5 Experiments

We performed our experiments on the dynamic model of Girona500 presented in (6), whose parameters have been identified in (Karras et al, 2013). 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 underactuated and any attempt to change the allocation matrix $\mathbf{B}$ would be ineffective. We use the following definition of the immediate cost:

$$c_t(\langle p_t, v_t \rangle) = \begin{cases} \| p_t - p_d \| & \text{if } t < T \\ w \| v_t - v_d \| & \text{if } t = T \end{cases} \tag{8}$$

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

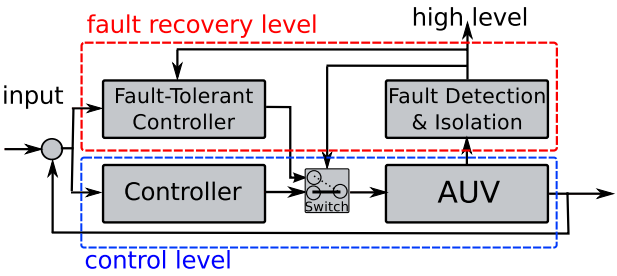


Fig. 11: Control architecture of the AUV including the controller level and the fault recovery level. The green line shows the state feedback used in the state-dependent policy representation.

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.

Experimental Result

A classical control architecture of an AUV includes a position/velocity controller that utilizes the desired inputs and the sensory feedback of the vehicle to control the position or velocity of the system. This architecture is illustrated in Fig. 11 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 recorded data is depicted in Fig. 12. It can be seen that in the second test (with broken right surge thruster) the original controller still tries to use the same thruster configuration as in the normal situation. Consequently, the controller fails to use the sway thruster as a means of recovery from the surge thruster failure. 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. 11) needs to be added to the control level architecture of the AUV (the dashed red box in Fig. 11). To this end, 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.

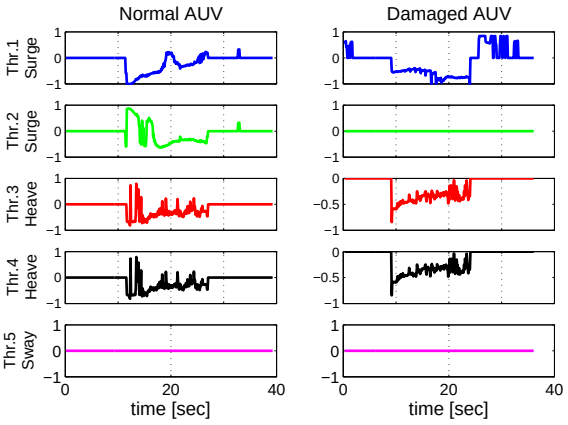


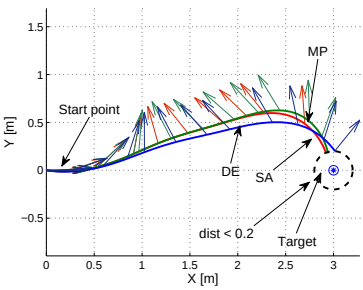
Fig. 12 The recorded thruster commands for a normal AUV (left column) and a damaged AUV (right column). In the damaged AUV case the right surge thruster is broken. The right column illustrates what happens when using the original controller scheme without any failure recovery functionality. In particular, the controller fails to use the sway thruster in order to circumvent the failed surge thruster.

Time-dependent Policy

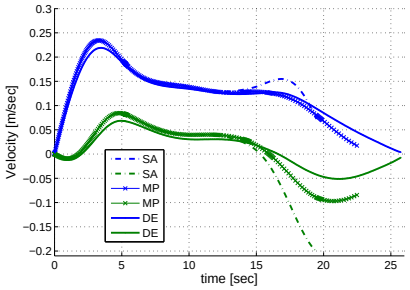
In the first experiment, the policy is represented by a linear function approximator that depends only on time t, $\Pi(t|\theta) = \theta^T \Phi(t)$. In this representation θ is the parameter vector and to represent $\Phi(t)$ we employ a 3^{rd} order Fourier basis (Konidaris et al, 2011). In this case 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. 13a, 13b, 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.

State-dependent Policy

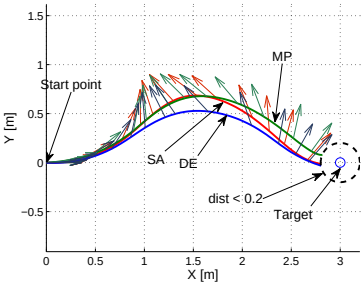
In this policy representation experiment, we close the loop by including feedback from the state variables (i.e. position, orientation, together with linear and angular velocities). In this case, the policy depends on the state variables χ, $\pi(\chi|\theta) = \theta^T \Phi(\chi)$, where θ is the parameter 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. 13c, 13d 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.



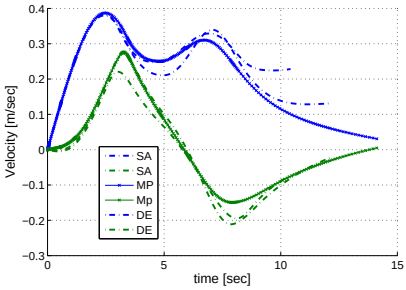
(a) Trajectories in 2D plane.



(b) Velocity profiles along X and Y axes.



(c) Trajectories in 2D plane.



(d) Velocity profiles along X and Y axes.

Fig. 13: Acquired results for the first experiment with time-dependent policy representation (a)-(b), and the experiment with state-dependent policy representation (c)-(d).

Real-world Experiment

In this real-world experiment, we test our approach on Girona500. As it is depicted in Fig. 14, 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. 14 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. 14. 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. 14. Finally, the same optimal solution is applied to the real robot and the recorded trajectory is plotted as the black trajectory in Fig. 14. 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).

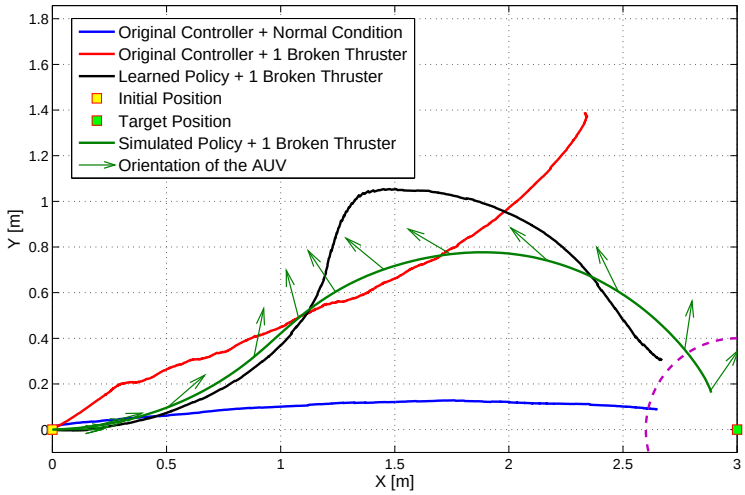


Fig. 14: The trajectories recorded in different scenarios during the real-world experiments.

5 Conclusion

The presented experiments with robot manipulators and autonomous underwater vehicles confirm the usefulness of robot learning approaches in real world scenarios. These learning methods address many of the facets of persistent autonomy, such as: coping with uncertainty, reacting to changing conditions, and fault tolerance. There are, of course, many other important issues of persistent autonomy that also need to be tackled. For example, a very important one is energy efficiency. There is promising research showing that learning methods can also be applied successfully for improving the energy efficiency of autonomous robots (Kormushev and Caldwell, 2013a,b). Another important issue is the ability to re-plan online and dynamically the mission plan according to the changes in the environment. Planning and learning are two areas of robotics research that can be mutually beneficial towards achieving the higher goal of persistent autonomy.

Acknowledgements We would like to thank Professor David Lane from the Ocean Systems Laboratory, Heriot-Watt University, UK, for introducing us to the topic of persistent autonomy.

We are grateful to Arnau Carrera, Narcís Palomeras, and Marc Carreras from the Computer Vision and Robotics Group (VICOROB), University of Girona, Spain, for making it possible to conduct real-world experiments with the Girona 500 AUV.

This work is supported by the European project PANDORA: Persistent Autonomy through learNing, aDaptation, Observation and ReplAnning, contract FP7-ICT-288273. (PANDORA, 2012)

References

Abidi MA, Eason RO, Gonzalez RC (1991) Autonomous robotic inspection and manipulation using multisensor feedback. Computer 24(4):17–31

Ahmadzadeh SR, Kormushev P, Caldwell DG (2013a) Autonomous robotic valve turning: A hierarchical learning approach. In: Robotics and Automation (ICRA), 2013 IEEE International Conference on, IEEE, pp 4614–4619

Ahmadzadeh SR, Leonetti M, Kormushev P (2013b) Online direct policy search for thruster failure recovery in autonomous underwater vehicles. In: 6th International workshop on Evolutionary and Reinforcement Learning for Autonomous Robot System (ERLARS 2013), Taormina, Italy

Ahmadzadeh SR, Jamisola RS, Kormushev P, Caldwell DG (2014a) Learning reactive robot behavior for autonomous valve turning. In: Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids 2014), Madrid, Spain

Ahmadzadeh SR, Leonetti M, Carrera A, Carreras M, Kormushev P, Caldwell DG (2014b) Online discovery of AUV control policies to overcome thruster failure. In: Robotics and Automation (ICRA), 2014 IEEE International Conference on, IEEE, pp 6522–6528

Ajoudani A, Lee J, Rocchi A, Ferrati M, Mingo E, Settimi A, Caldwell DG, Bicchi A, Tsagarakis N (2014) A manipulation framework for compliant humanoid COMAN: Application to a valve turning task. In: 2014 IEEE-RAS International Conference on Humanoid Robots (Humanoids 2014), IEEE, pp 664–670

Alessandri A, Caccia M, Veruggio G (1998) A model-based approach to fault diagnosis in unmanned underwater vehicles. In: OCEANS'98 Conference Proceedings, IEEE, vol 2, pp 825–829

Alunni N, Phillips-Grafftin C, Suay HB, Lofaro D, Berenson D, Chernova S, Lindeman RW, Oh P (2013) Toward a user-guided manipulation framework for high-dof robots with limited communication. In: Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on, IEEE, pp 1–6

Anisi DA, Persson E, Heyer C (2011) Real-world demonstration of sensor-based robotic automation in oil & gas facilities. In: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, IEEE, pp 235–240

Anisi DA, Skourup C, Petrochemicals A (2012) A step-wise approach to oil and gas robotics. In: IFAC Workshop on Automatic Control in Offshore Oil and Gas Production, Trondheim, Norway, vol 31

Antonelli G (2003) A survey of fault detection/tolerance strategies for AUVs and ROVs. In: Fault diagnosis and fault tolerance for mechatronic systems: Recent advances, Springer, pp 109–127

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

Bristow D, Tharayil M, Alleyne AG, et al (2006) A survey of iterative learning control. Control Systems, IEEE 26(3):96–114

Caccia M, Bono R, Bruzzone G, Bruzzone G, Spirandelli E, Veruggio G (2001) Experiences on actuator fault detection, diagnosis and accomodation for ROVs. International Symposiyum of Unmanned Untethered Sub-mersible Technol

Carrera A, Ahmadzadeh S, Ajoudani A, Kormushev P, Carreras M, Caldwell D (2012) Towards autonomous robotic valve turning. Cybernetics and Information Technologies 12(3)

Cheng ASf, Leonard NE (1999) Fin failure compensation for an unmanned underwater vehicle. In: Proceedings of the 11th International Symposium on Unmanned Untethered Submersible Technology, Citeseer

Das SN, Das SK (2004) Determination of coupled sway, roll, and yaw motions of a floating body in regular waves. International Journal of Mathematics and Mathematical Sciences 2004(41):2181–2197

Hamilton K, Lane D, Taylor N, Brown K (2001) 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, IEEE, vol 4, pp 3232–3237

Ijspeert AJ, Nakanishi J, Hoffmann H, Pastor P, Schaal S (2013) Dynamical movement primitives: learning attractor models for motor behaviors. Neural computation 25(2):328–373

Jamisola RS, Oetomo DN, Ang MH, Khatib O, Lim TM, Lim SY (2005) Compliant motion using a mobile manipulator: an operational space formulation approach to aircraft canopy polishing. Advanced Robotics 19(5):613–634

Karras GC, Bechlioulis CP, Leonetti M, Palomeras N, Kormushev P, Kyriakopoulos KJ, Caldwell DG (2013) On-line identification of autonomous underwater vehicles through global derivative-free optimization. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)

Khatib O (1987) A unified approach for motion and force control of robot manipulators: The operational space formulation. Robotics and Automation, IEEE Journal of 3(1):43–53

Kirkpatrick S, Jr DG, Vecchi MP (1983) Optimization by simulated annealing. science 220(4598):671–680

Konidaris G, Osentoski S, Thomas PS (2011) Value function approximation in reinforcement learning using the fourier basis. In: AAAI

Kormushev P, Caldwell DG (2013a) Improving the energy efficiency of autonomous underwater vehicles by learning to model disturbances. In: Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS), Tokyo, Japan

Kormushev P, Caldwell DG (2013b) Towards improved AUV control through learning of periodic signals. In: Proc. MTS/IEEE Intl Conf. OCEANS 2013, San Diego, USA

Kormushev P, Calinon S, Caldwell DG (2011) Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. Advanced Robotics 25(5):581–603

Lane DM, Maurelli F, Kormushev P, Carreras M, Fox M, Kyriakopoulos K (2012) Persistent autonomy: the challenges of the PANDORA project. Proceedings of IFAC MCMC

Leonetti M, Kormushev P, Sagratella S (2012) Combining local and global direct derivative-free optimization for reinforcement learning. Cybernetics and Information Technologies 12(3):53–65

Leonetti M, Ahmadzadeh SR, Kormushev P (2013) On-line learning to recover from thruster failures on autonomous underwater vehicles. In: OCEANS 2013, IEEE

Moore KL (2012) Iterative learning control for deterministic systems. Springer Science & Business Media

Orsag M, Korpela C, Bogdan S, Oh P (2014) Valve turning using a dual-arm aerial manipulator. In: Unmanned Aircraft Systems (ICUAS), 2014 International Conference on, IEEE, pp 836–841

PANDORA (2012) Persistent autonomy through learning, adaptation, observation and re-planning. http://persistentautonomy.com/, PANDORA European Project

Perrault D, Nahon M (1998) Fault-tolerant control of an autonomous underwater vehicle. In: OCEANS'98 Conference Proceedings, IEEE, vol 2, pp 820–824

Podder T, Antonelli G, Sarkar N (2000) 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, IEEE, vol 2, pp 1251–1256

Podder TK, Sarkar N (2001) Fault-tolerant control of an autonomous underwater vehicle under thruster redundancy. Robotics and Autonomous Systems 34(1):39–52

Raibert MH, Craig JJ (1981) Hybrid position/force control of manipulators. Journal of Dynamic Systems, Measurement, and Control 103(2):126–133

Ribas D, Palomeras N, Ridao P, Carreras M, Mallios A (2012) Girona 500 AUV: From survey to intervention. Mechatronics, IEEE/ASME Transactions on 17(1):46–53

Schaal S, Ijspeert A, Billard A (2003) Computational approaches to motor learning by imitation. Philosophical Transactions of the Royal Society of London Series B: Biological Sciences 358(1431):537–547

Seto ML (2011) An agent to optimally re-distribute control in an underactuated AUV. International Journal of Intelligent Defence Support Systems 4(1):3–19

Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization 11(4):341–359

Wang L (1999) A Course on Fuzzy Systems. Prentice-Hall press, USA

Yoshikawa T, Zheng XZ (1993) Coordinated dynamic hybrid position/force control for multiple robot manipulators handling one constrained object. The International Journal of Robotics Research 12(3):219–230