DOCTORAL SCHOOL
ON LIFE AND HUMANOID TECHNOLOGIES

Robot Learning of Reactive and Visuospatial Skills

Seyed Reza Ahmadzadeh

Istituto Italiano di Technologia (IIT), Italy
Università degli Studi di Genova, Italy

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy (Ph.D.)
in Robotics, Cognition and Interaction Technologies

February 2015

Robot Learning of Reactive and Visuospatial Skills

by

Seyed Reza Ahmadzadeh

Submitted to the Department of Advanced Robotics, IIT
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Ph.D.)
in Robotics, Cognition and Interaction Technologies

at the

ITALIAN INSTITUTE OF TECHNOLOGY

February 2015

Author .....
Department of Advanced Robotics, IIT
February 20, 2015

Certified by .....
Dr. Petar Kormushev
Team Leader
Thesis Supervisor

Certified by .....
Prof. Darwin G. Caldwell
Director
Thesis Supervisor

Supervisors:

Dr. Petar Kormushev

Team Leader
Learning and Interaction Group
Department of Advanced Robotics (ADVR),
Istituto Italiano di Tecnologia (IIT)

Prof. Darwin G. Caldwell

Director
Department of Advanced Robotics (ADVR),
Istituto Italiano di Tecnologia (IIT)

Committee Members:

Prof. Andrea Caiti

ISME - Interuniversity Research Center on Integrated Systems for the Marine Environment
DII, Department of Information Engineering and Centro "E. Piaggio", Università di Pisa

Prof. Elena De Momi

Department of Electronics, Information and Bioengineering,
Politecnico di Milano

Prof. Bram Vanderborght

Department of Mechanical Engineering, Faculty of Applied Sciences,
Vrije Universiteit Brussel

Dr. Leonardo De Mattos

Department of Advanced Robotics (ADVR),
Istituto Italiano di Tecnologia (IIT)

Abstract

CURRENT autonomous robots perform adequately in structured environments such as research laboratories. Facing the uncertain and dynamically changing conditions of the real-world, however, they quickly fail and need human assistance. Nowadays, the challenges of unstructured environments receive more attention from robotic researchers who aim to enhance the level of autonomy in robots. The ultimate goal is to develop autonomous systems capable of coping with uncertainty, reacting to changing conditions, fault tolerance, learning new skills, object manipulation, and so on. The present thesis pursues this goal by proposing novel robot learning approaches that elevate the robot autonomy. The proposed approaches address several important capabilities of autonomous systems by placing the focus on learning reactive and visuospatial skills. These skills are extremely vital for many aspects of autonomous systems.

Reactive behavior is one of the most crucial behaviors that an autonomous system should possess. Unlike biological systems, robots appear quite incapable of dealing with unpredictable and changing environments due to their lack of reactivity. To address this problem, a novel hierarchical learning approach is proposed that allows a robot to react to changing conditions while operating in an unstructured environment. The proposed approach is tested on the challenging task of autonomous valve turning. The hierarchical approach consists of three main layers. Each layer realizes specific subtasks to improve the autonomy of the system. In the first layer, the robot acquires novel motor skills for approaching and grasping the valve through kinesthetic teaching. A hybrid force/motion control strategy is utilized in this layer to minimize the undesired forces and torques during the turning phase. A Reactive Fuzzy Decision Maker (RFDM) is devised in the second layer which reacts to relative movements, sensor delays and applied forces/torques according to the distance between the valve and the robot. The reactive layer modulates the robot's movement accordingly by changing the direction and rate of the movement. In the third layer, a supervised learning method is utilized to tune the behavior of the system based on expert knowledge. Conscious knowledge of a human expert is utilized to generate the fuzzy rule set. Then, the subconscious knowledge of the human expert is used to tune the generated rules. The proposed hierarchical learning approach is validated through real-world experiments both in laboratory and underwater environments. The results show the feasibility and efficiency of the proposed approach.

Autonomous robots should react appropriately not only to external disturbances but also to failures and uncertainties which occur internally in the system itself. To achieve this, a learning framework is proposed that enables robots to learn a reactive behavior for dealing with internal failures. The proposed framework places the focus on improving the fault-tolerance and reliability of Autonomous Underwater Vehicles (AUVs). One of the most critical failures which can endanger both the underwater robot and the mission is thruster failure. Employing a model-based direct policy search, the proposed learning framework discovers new control policies to overcome thruster failures as they happen. The policies are learned on a simulated model of the AUV on-board. When a fault is detected and isolated, the model is reconfigured according to the new condition. Since the learning framework generates an optimal trajectory, the learned fault-tolerant policy is able to navigate the AUV towards a specified target with minimum cost. The learned policy

is then executed on the real robot in a closed-loop using the state feedback of the AUV. One of the advantages of the proposed framework is that it is applicable in both cases when the AUV becomes under-actuated in the presence of a fault or remains over-actuated. For dealing with multiple conflicting objectives, the proposed learning framework is extended by employing multi-objective reinforcement learning. The extended framework discovers a set of optimal solutions, each of which can be used to generate a trajectory that is able to navigate the AUV towards a specified target while satisfying multiple objectives. Unlike most existing methods which disregard the faulty thruster, the proposed framework can also deal with partially broken thrusters. The feasibility and efficiency of the proposed learning framework are validated through simulated and real-world experiments.

Autonomous robots should not only passively perceive the world but also actively change it. Therefore it is important to improve the robot autonomy in manipulation tasks. A novel skill learning approach is proposed that allows a robot to acquire human-like visuospatial skills for object manipulation tasks. Visuospatial skills are attained by observing the spatial relationship among objects through demonstrations. Visuospatial Skill Learning (VSL) utilizes visual perception as the main source of information. As opposed to conventional trajectory-based learning approaches, VSL is a goal-based robot learning approach that focuses on achieving a desired goal configuration of objects relative to one another while maintaining the sequence of operations. VSL is capable of learning and generalizing multi-operation skills from a single demonstration, while requiring minimum *a priori* knowledge about the objects and the environment. In contrast to many existing approaches, VSL leverages simplicity, efficiency and user-friendly human-robot interaction. Several real-world experiments are conducted to show the validity of the proposed approach. In VSL, the primitive actions have to be programmed manually into the system. To handle this drawback, VSL is further integrated with a conventional trajectory-based approach, imitation learning. The sensorimotor skills are learned through imitation learning. The sequence of performed actions is learned through demonstrations using VSL. The obtained integrated approach easily extends and adopts robot's capabilities to novel situations, even by users without programming ability. Furthermore, VSL utilizes its own simple internal planner in the reproduction phase. To utilize the advantages of standard symbolic planners, the original planner is replaced with an existing action-level symbolic planner. In order to ground the actions, the symbolic actions are defined in the planner and VSL maps identified preconditions and effects in a formalism suitable to be used at the symbolic level. The experimental results show the improvement in generalization to find a solution that can be acquired from both multiple and single demonstrations.

The proposed approaches in this thesis are validated through simulations and real-world experiments. Four robotic platforms are used: a Barret WAM manipulator, a KUKA-DLR manipulator, an iCub humanoid robot, and a Girona500 AUV. The results demonstrate the feasibility and efficiency of the proposed approaches to increase the robot autonomy in real-world scenarios.

to my beloved Fouzieh

and to my ever-supportive parents

Acknowledgments

Firstly, I would like to thank my supervisor, Dr. Petar Kormushev, who has served as a role model throughout my graduate career. I attribute my development as a researcher to his advice, patience, suggestions, and encouragement. This thesis would not be possible without his persistent support. I am also deeply thankful to Petar for his remarkable attention and continuous feedback on my ideas, with his insights and inspiration. I have learned much from him as a friend and a great example of professional researcher.

I wish to thank Prof. Darwin G. Caldwell, as the director of the Advanced Robotics Department, IIT, for his advice and support. Several robotic platforms and equipments has been made available by him for researchers to validate their ideas through real-world experiments. I am thankful to Prof. Darwin Caldwell for backing me up to attend international conferences which helped me to collect valuable experiences and shape me as a researcher.

I thank the secretaries and administrative assistants of our department, Floriana Sardi, Silvia Ivaldi, Valentina Rosso, Simona Ventriglia, Monica Vasco and Elisa Repetti for organizing several activities. I would like to especially thank Valentina Rosso for helping me out during the first months with resolving many official and personal paperwork. I am grateful to Silvia Ivaldi, who patiently helped by organizing missions and many other activities including my participation to amazing international conferences, summer schools and workshops.

In the context of my research as a member of the European project PANDORA, I would like to say a special thank you to Arnau Carrera, Marc Carreras and Narcís Palomeras and their colleagues in Girona for their support and willingness to help whenever it was necessary. In the PANDORA project, I had the pleasure of working with great people, Prof. David Lane, Francesco Maurelli, George Karras, and Bechlioulis Charalampos.

Thanks go to Dr. Lorenzo Natale, for allowing me to use the iCub humanoid robot for conducting real-world experiments during the development of a part of this thesis. I am thankful to Lorenzo for putting his trust on me and giving me the opportunity to work with his team in the iCub facility department, IIT, as a Fellow and finally a Post-Doc.

Special thanks to Dr. Fulvio Masterogiovanni, University of Genoa, for his innovative and valuable ideas that shaped a pavement to the future. Heartfelt thanks to Matteo Leonetti, with whom I had the opportunity to share and discuss interesting ideas about Reinforcement Learning. I wish to thank my friend and colleague Ali Paikan who helped me technically and spiritually and warmed me up whenever it was getting cold. I would like to thank my friends and colleagues at IIT, Davide De Tommaso, Arash Ajoudani, Rodrigo Jamisola, Matteo Leonetti, Milad Malekzadeh, Przemyslaw Kryczka, and Houman Dallali with whom I shared amazing scientific and non-scientific moments.

I would never reached the place where I am standing now without the endless support, love, and encouragement of my parents. I am thankful for their confidence in me. Many thanks to my brothers, Ebad and Azim for all their back up and nice moments we spent together.

The final word of appreciation goes most especially to my wonderful and compassionate wife, Fouzieh. I will be forever grateful for her patience, understanding, and love.

Contents

1 Introduction **29**

1.1 Motivations 29

1.2 Contributions 35

1.3 Thesis Outline 39

2 Learning Reactive Behavior: A Hierarchical Approach **43**

2.1 Motivation 43

2.2 Related Work 46

2.3 Methodology 49

2.4 Imitation Learning 56

2.5 Reactive Fuzzy Decision Maker 1 62

2.5.1 Design of the Fuzzy System (RFDM/1) 63

2.5.2 Tuning the Fuzzy System (RFDM/1) 65

2.6 Experiments with RFDM/1 77

2.6.1 Experimental Setup 77

2.6.2 Experimental Results 78

2.7 Force/Motion Control Strategy 79

2.7.1 Valve Turning with Stationary Base 82

2.7.2 Valve Turning with Moving Base 85

2.8 Reactive Fuzzy Decision Maker 2 86

2.8.1 Design of the Fuzzy System (RFDM/2) 86

2.8.2 Tuning the Fuzzy System (RFDM/2) 90
2.9 Experiments with RFDM/2 92
2.9.1 Experimental Setup 92
2.9.2 Experimental Results 92
2.10 Reactive Fuzzy Decision Maker 3 93
2.10.1 Design of the Fuzzy System (RFDM/3) 94
2.10.2 Tuning the Fuzzy System (RFDM/3) 95
2.11 Underwater Experiments with RFDM/3 95
2.11.1 Experimental Setup 96
2.11.2 Experiment with Lateral Oscillations of the AUV 97
2.11.3 Experiment with Vertical Oscillations of the Panel 100
2.11.4 Experiment with Occluding the Valve 103
2.12 Chapter Summary 106

3 Learning Reactive Behavior: A Fault-Tolerant Approach 108
3.1 Motivation 108
3.2 Related Work 111
3.3 Methodology 115
3.3.1 AUV Model 116
3.3.2 Fault Detection Module 118
3.3.3 Policy Representation 119
3.3.4 Reward Function 121
3.3.5 Optimization Algorithms 122
3.3.6 Online Procedure 124
3.4 Experiments 125
3.4.1 Setup 125
3.4.2 Controller Test 126
3.4.3 Time-dependent Policy 127
3.4.4 State-dependent Policy 129
3.4.5 Point-to-Point Navigation 131

3.4.6 Navigation through Waypoints . . . 132
3.4.7 Covariance Analysis . . . 133
3.4.8 Learning Distributions . . . 137
3.4.9 Computational Cost . . . 138
3.4.10 Real-world Experiment . . . 139
3.5 Failure Recovery using Multi-Objective Reinforcement Learning . . . 142
3.5.1 Related Work on Multi-Objective Reinforcement Learning . . . 143
3.5.2 A More Generic Fault Detection Module . . . 144
3.5.3 Vector Reward Function . . . 145
3.5.4 Multi-Objective Optimization Problem . . . 146
3.5.5 Multi-Objective Optimization Algorithms . . . 147
3.6 Experiments with Multi-Objective Reinforcement Learning . . . 150
3.6.1 Setup . . . 151
3.6.2 A Fully Broken Thruster . . . 152
3.6.3 A Partially Broken Thruster . . . 154
3.6.4 Improving the Performance . . . 157
3.7 Chapter Summary . . . 159

4 Visuospatial Skill Learning 162
4.1 Motivation . . . 162
4.2 Related Work . . . 165
4.3 Introduction to Visuospatial Skill Learning . . . 172
4.3.1 Terminology . . . 174
4.3.2 Problem Formulation . . . 175
4.3.3 Methodology . . . 175
4.4 Implementation of VSL . . . 178
4.4.1 Coordinate Transformation . . . 178
4.4.2 Image Processing . . . 179
4.4.3 Segmentation . . . 181
4.4.4 Trajectory Generation . . . 184

4.4.5 Grasp Synthesis . . . 187
4.5 Experimental Results of VSL . . . 188
4.5.1 Simulated Experiments . . . 188
4.5.2 Real-world Experiments . . . 193
4.6 3D Visuospatial Skill Learning . . . 202
4.6.1 Point Cloud Processing . . . 202
4.6.2 2D vs. 3D VSL . . . 205
4.6.3 Experimental Results of 3D VSL . . . 206
4.7 Learning Symbolic Representations of Actions using VSL . . . 211
4.7.1 Overview of Symbolic VSL . . . 213
4.7.2 Methodology . . . 215
4.7.3 Learning Sensorimotor Skills by Imitation Learning . . . 215
4.7.4 Learning Action Sequences by VSL . . . 218
4.7.5 Generalization of Learned Actions as Symbolic Action Models . . . 219
4.7.6 Implementation for Learning Symbolic Representation . . . 220
4.7.7 Experimental Results of Symbolic VSL . . . 221
4.8 Chapter Summary . . . 227

5 Conclusions **230**
5.1 Thesis Summary . . . 230
5.2 Contributions . . . 233
5.3 Future Work . . . 238

Appendices **239**

Appendix A Differential Evolution Algorithm **240**

Appendix B CMA-ES Algorithm **244**

Appendix C Cross Entropy Method **249**

Appendix D Simulated Annealing Algorithm **251**

Appendix E Modified Price Algorithm **254**

List of Publications **259**

List of Figures

2-1 Autonomous Robotic Valve Turning. 45
2-2 A high-level diagram illustrating the three layers of the proposed hierarchical learning approach. For more information, see Section 2.3. 50
2-3 A high-level flow diagram illustrating the different components of the proposed hierarchical system including RFDM/1. For more information, see Section 2.3. 52
2-4 A high-level flow diagram illustrating the different components of the proposed hierarchical system including RFDM/2. For more information, see Section 2.3. 54
2-5 A high-level flow diagram illustrating the different components of the proposed hierarchical system including RFDM/3. For more information, see Section 2.3. 55
2-6 A high-level diagram illustrating the imitation learning layer in the proposed hierarchical approach. For more information, see Section 2.4. 57
2-7 The recorded trajectories that form the set of demonstrations (black), and the reproduced trajectory from an arbitrary initial position (red) towards the target are illustrated. The blue ellipses show the attractors and the yellow square shows the target (the valve). For more information, see Section 2.4. 60

2-8 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. For more information, see Section 2.4. 61
2-9 The robot reaching the valve during the reproduction phase. For more information, see Section 2.4. 62
2-10 A high-level diagram illustrating RFDM/1. For more information, see Section 2.5. 63
2-11 Fuzzy inference system surface for RFDM/1 including two inputs (u_1 and u_2). For more information, see Section 2.5.1. 65
2-12 A high-level diagram illustrating the tuning layer in the proposed hierarchical approach. For more information, see Section 2.5.2. 66
2-13 Tuning the RFDM/1 system using Batch Gradient Descent. For more information, see Section 2.5.2. 68
2-14 Tuning the RFDM/1 system using Cross Entropy Method. For more information, see Section 2.5.2. 70
2-15 Tuning the RFDM/1 system using CMA-ES Method. For more information, see Section 2.5.2. 72
2-16 Tuning the RFDM/1 system using Modified Price's Method. For more information, see Section 2.5.2. 74
2-17 Final fuzzy surfaces are depicted in 2-17a. The tuned membership functions are illustrated in 2-17b. For more information, see Section 2.5.2. 76
2-18 A set of snapshots in 2-18a shows the execution of the task. The reproduction of the task is illustrated by the trajectory depicted in 2-18b. The effect of the reactive system can be seen in the middle of the trajectory while the valve is oscillating. For more information, see Section 2.6. 79

2-19 The experimental setup for the valve turning task in the second set of experiments. 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 tutor can create a random disturbance by perturbing the base of the robot. For more information, see Section 2.7. 81
2-20 Joint angles during the valve turning task with stationary base. For more information, see Section 2.7.1. 83
2-21 End-effector forces during the valve turning task. For more information, see Section 2.7.1. 84
2-22 End-effector torques during the valve turning task. For more information, see Section 2.7.2. 85
2-23 A high-level diagram illustrating RFDM/2. For more information, see Section 2.8. 86
2-24 Fuzzy membership functions defined for each input in RFDM/2. For more information, see Section 2.8.1. 88
2-25 Fuzzy inference system surface for RFDM/2 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. (For more information, see Section 2.8.1). 89
2-26 The results obtained from applying the CMA-ES algorithm with five restarts. The plot illustrates the time evolution of the distribution mean (default) or the recent best solution vector. For more information, see Section 2.8.2. 91
2-27 The recorded set of inputs and the generated decision commands by the RFDM/2 system during a real-world valve turning experiment. For more information, see Section 2.9. 93
2-28 A high-level diagram illustrating RFDM/3. For more information, see Section 2.10. 94

2-29 A set of snapshots showing the the Girona500 AUV, the manipulator arm and the gripper, the vision sensor and the panel in the underwater experiments. For more information, see Section 2.11.1 96
2-30 A schematic of the Girona500 AUV showing the thruster layout used in the underwater experiments. The sway thruster is used to create lateral disturbances. For more information, see Section 2.11.1 97
2-31 A schematic of the Girona500 AUV from top-view illustrates the panel, valve, arm, and AUV. Lateral disturbances applied to AUV using the sway thruster which generate relative movement between the gripper and the valve. For more information, see Section 2.11.2. 98
2-32 Lateral disturbances applied to AUV using the sway thruster. For more information, see Section 2.11.2. 98
2-33 Movements of the AUV in 3 axes recorded during the execution of the valve turning task in the first underwater experiment. For more information, see Section 2.11.2 99
2-34 Oscillating the AUV laterally through teleoperation provides the opportunity to investigate the behavior of the RFDM/3 system under uncertainties caused by increasing the relative movement. For more information, see Section 2.11.2 100
2-35 A schematic of the Girona500 AUV from back-view illustrates the panel, valve, AR marker, and AUV. Vertical disturbances applied to the panel by a human expert which generates relative movement between the gripper and the valve. For more information, see Section 2.11.3. 101
2-36 Disturbances applied to the panel by a human expert to simulate the water currents and disturbances applied to AUV. Snapshots in 2-36b and 2-36b are taken from different experiments. The dashed line in Figure 2-36a is a baseline that indicates the movement of the panel. For more information, see Section 2.11.3. 102

2-37 Movements of the AUV in 3 axes recorded during the execution of the valve turning task in the second underwater experiment. For more information, see Section 2.11.3 102
2-38 Oscillating the panel vertically provides the opportunity to investigate the behavior of the RFDM/3 system under uncertainties caused by increasing the relative movement. For more information, see Section 2.11.3 103
2-39 A schematic of the Girona500 AUV from back-view illustrates the main panel, the alternative panel used for occluding the valve, and AUV. For more information, see Section 2.11.4. 104
2-40 A set of snapshots representing the experiment with covering the valve using a panel to investigate the behavior of the RFDM/3 system under uncertainties caused by sensor delay. For more information, see Section 2.11.4 105
2-41 Movements of the AUV in 3 axes recorded during the execution of the valve turning task. For more information, see Section 2.11.4 105
2-42 Covering the valve using another panel provides the opportunity to investigate the behavior of the RFDM/3 system under uncertainties caused by sensor delay. For more information, see Section 2.11.4 106

3-1 A diagram illustrating 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. For more information, see Section 3.3. 116
3-2 The Girona500 AUV equipped with 5 thrusters (3 are visible in the photo). In the conducted experiment, one of the surge thrusters is broken. 117
3-3 The selected thruster layout for Girona500 including five thrusters. 118
3-4 Control architecture of the AUV including the controller level and the fault recovery level. Once a failure is detected and isolated, the system employs the proposed fault-tolerant framework. For more information, see Section 3.4.2 and 3.4.4. 126

3-5 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. For more information, see Section 3.4.2. 127
3-6 Acquired results for the experiments with time-dependent policy using Simulated Annealing, Modified Price and Differential Evolution algorithms. The discovered solutions by the employed algorithms are similar. For more information, see Section 3.4.3. 128
3-7 Acquired statistical results for the experiments with time-dependent policy representation over 50 runs using Simulated Annealing, Modified Price, and Differential Evolution algorithms. For more information, see Section 3.4.3. . . . 129
3-8 Acquired results for the experiments with state-dependent policy using Simulated Annealing, Modified Price and Differential Evolution algorithms. The discovered solutions by the employed algorithms are similar. For more information, see Section 3.4.4. 130
3-9 Acquired statistical results for the experiments with state-dependent policy representation over 50 runs using Simulated Annealing, Modified Price, and Differential Evolution algorithms. For more information, see Section 3.4.4. . . . 131
3-10 The result trajectories for the point-to-point navigation experiment using time-dependent policy representation. The yellow square is the initial position of the AUV, and the green squares show the desired final positions. For more information, see Section 3.4.5. 132
3-11 The desired path and acquired trajectories in the case that the AUV navigates through waypoints. The dark blue and the light blue profiles show the desired straight and curved paths respectively. The red and the light green profiles illustrate the acquired corresponding trajectories. The arrows show the orientation of the AUV at each point along the path. For more information, see Section 3.4.6. 133

3-12 The positions reached by AUV during a noisy simulation. Policy 1 had the highest score, while Policy 5 was the fifth ranking policy. The colors correlate with policies in Figure 3-13. For more information, see Section 3.4.7. 135
3-13 Simulation results showing the effect of noise on selected five policies. The policies are ranked according their reward. The figures illustrate that policies with higher ranking are not necessarily robust to noise in the environment. For more information, see Section 3.4.7. 136
3-14 The thruster commands. The blue curve is the command for the surge thruster and the red curve is the command for the sway thruster. The figure shows that in this experiment, the sway thruster reaches saturation. For more information, see Section 3.4.7. 137
3-15 The comparison before and after learning the parameter distributions. The algorithm needs less number of function evaluations to reach the same results after learning the parameter distributions. Blue lines and red lines show the results before and after learning the distributions respectively. For more information, see Section 3.4.8. 138
3-16 Computational cost of the learning method to discover a fault-tolerant policy. For more information, see Section 3.4.9. 139
3-17 Two sets of snapshots illustrating the behavior of the AUV in normal condition and in the presence of thruster failure in one of the surge thrusters. The target is set 3.0 m in front of the AUV in surge direction. For more information, see Section 3.4.10. 140
3-18 The trajectories recorded in different scenarios during the real-world experiments. For more information, see Section 3.4.10. 141
3-19 Snapshots illustrating the movement of the AUV while executing the discovered fault-tolerant policy. For more information, see Section 3.4.10. 142

3-20 Three optimal trajectories and corresponding velocity profiles acquired by employing the discovered *Pareto* optimal solutions for a fully broken thruster. The arrows show the orientation of the AUV. For each trajectory two linear velocity profiles exist (u in x direction and v in y direction). For more information, see Section 3.6.2. 153
3-21 Acquired results for the 2nd multi-objective experiment with various coefficient of functionality for the right surge thruster which is a dimensionless quantity between $[0,1]$. In all sub-figures the x and y axes show the surge and sway directions respectively in the horizontal plane of AUV and the dimensions are in meters. For more information, see Section 3.6.3. 155
3-22 Acquired results for the 2nd multi-objective experiment with $\alpha_2 = 0.4$. In sub-figures 3-22a, 3-22b, 3-22c the x and y axes show the surge and sway respectively and the dimensions are in meters. For more information, see Section 3-22. 156
3-23 The reward-space *Pareto* Front is plotted over two objectives, in which the dominant solutions are marked as black. The movement of populations towards minimum rewards is shown with a color gradient. The axis are dimensionless. For more information, see Section 3-22. 157
3-24 The discovered policy parameters from the 2nd experiment in Section 3.6.3. Each row is related to a thruster. In each row 16 parameters of the policy representation, ω, are depicted. Each parameter is re-scaled to range $[-1,1]$ and is plotted versus α. Both axes illustrate dimensionless quantities. For more information, see Section 3.6.4. 158
3-25 The box-plots show the effect of employing previously experienced knowledge in the performance of the approach. The first experiment starts from random values, whereas the second experiment uses the nearest neighbor solution. The left plot shows the number of function evaluations and the right plot depicts the time needed to reach the goal. Both experiments were averaged over 20 runs. For more information, see Section 3.6.4. 159

4-1 A high-level flow diagram illustrating the demonstration and reproduction phases in the VSL approach. For more information, see Section 4.3. 173
4-2 A schematic illustrating the virtual experimental setup for an object manipulation task. For more information, see Section 4.3. 173
4-3 Coordinate transformation from the image plane to the workspace plane (**T** represents the homography matrix). For more information, see Section 4.4.1. . . 178
4-4 Image processing techniques utilized in VSL. The result of applying the background subtraction and thresholding for a place action (right). Match finding result between the 4^{th} observation and the *world* in the 4^{th} operation during reproduction of the Domino task using SIFT (left). For more information, see Section 4.4.2. 181
4-5 Segmentation process applied to a set of captured images from the scene during the demonstration. The meaningful changes in the scene are detected by calculating the absolute difference in each step. The snapshots including a meaningful change are highlighted with a red rectangle frame. The rest of the snapshots are discarded. The bottom row shows the absolute difference between two consecutive snapshots. For more information, see Section 4.4.3. 182
4-6 Segmentation process applied to a set of captured images from the scene during the demonstration. The meaningful changes in the scene are detected by calculating the histogram counts in each step. The snapshots including a meaningful change are highlighted with a red rectangle frame. The bottom row shows the histogram in each step. For more information, see Section 4.4.3. 183
4-7 Segmentation process applied to a set of captured images from the scene during the demonstration. The meaningful changes in the scene are detected by calculating the 2D correlation coefficient in each step. The snapshots including a meaningful change are highlighted with a red rectangle frame. The rest of the snapshots are discarded. For more information, see Section 4.4.3. 184

4-8 From left to right, the re-scaled absolute difference, the re-scaled histogram difference, and the re-scaled 2D correlation coefficient for the set of images shown in Figure 4-7. The red bars highlights the snapshots containing the meaningful operations. For more information, see Section 4.4.3. 184
4-9 The generated trajectory including position and orientation profiles for a spatial pick-and-place. For more information, see Section 4.4.4. 186
4-10 The Barrett Hand grasping an object on the test-stand. For more information, see Section 4.4.5. 187
4-11 Grasping an object during a real-world experiment. For more information, see Section 4.4.5. 188
4-12 House scene simulated experiment to illustrate the relative and absolute positioning capabilities of VSL. For more information, see Section 4.5.1. 189
4-13 Roof placement simulated experiment to illustrate VSL's capability of disambiguation of multiple alternative matches. For more information, see Section 4.5.1. 191
4-14 Tower of Hanoi simulated experiment to illustrate the capabilities of VSL to perform interactive reproduction, disambiguation, and absolute positioning. For more information, see Section 4.5.1. 192
4-15 The experimental setup for a Visuospatial Skill Learning (VSL) task. For more information, see Section 4.5.2. 193
4-16 Alphabet ordering. The initial configurations of the objects in 4-16a and 4-16b are different. The red arrows show the pick-and-place operations. For more information, see Section 4.5.2. 195
4-17 Animal puzzle. The initial and the final configurations of the objects in the *world* are different in (a) and (b). The red arrows show the pick-and-place operations. For more information, see Section 4.5.2. 196
4-18 Tower of Hanoi. The sequence of operations in the reproduction phase by the robot. For more information, see Section 4.5.2. 197

4-19 The sequence of operations in the demonstration phase. Each column represents one pick-and-place operation. In each operation, the tutor picks one object and classifies it either as an 'animal' or a 'machine'. The selected object in each operation is shown in the middle row. For more information, see Section 4.5.2. 198
4-20 Two sets of match finding results in the reproduction phase are shown. For more information, see Section 4.5.2. 199
4-21 Two operations during the reproduction phase are shown. The red crosses on the objects and on the bins, show the detected positions for pick and place actions respectively. For more information, see Section 4.5.2. 199
4-22 The sequence of reproduction performed by the robot and the tutor are shown for the turn-taking task of domino. For more information, see Section 4.5.2. 200
4-23 Result of a subtraction process between a pre-pick (top-left) and a pre-place (bottom-left) point clouds. The original clouds are voxelized at a resolution of 2 mm and the distance threshold for the KNN-search is set to 1 cm. For more information, see Section 4.6.1. 203
4-24 4-24a illustrates the captured RGB-D cloud of a scene. Estimated normals in 3 axes are depicted in 4-24b,4-24c and 4-24d. For more information, see Section 4.6.1. 204
4-25 4-25a illustrates the captured RGB-D cloud of a scene. The result of the match finding process using different methods are shown in 4-25b,4-25c and 4-25d. For more information, see Section 4.6.1. 205
4-26 The experimental setup utilized for the conducted experiments in Section 4.6.3 using 3D VSL. For more information, see Section 4.6.3. 206
4-27 The set of objects used in the table cleaning task. For more information, see Section 4.6.3. 208
4-28 The sequence of operations reproduced by the robot for the table cleaning task. For more information, see Section 4.6.3. 208

4-29 Demonstration of the piling task. The initial and final configurations of the objects in the world are shown in top (RGB-D). The bottom row, from left to right, shows the sequence of operations by the tutor. For more information, see Section 4.6.3. 209
4-30 Reproduction of the piling task. The initial and final configurations of the objects in the world are shown in top (RGB-D). The bottom row, from left to right, shows the sequence of operations by the robot. For more information, see Section 4.6.3. 210
4-31 The iCub robot interacting with objects. 213
4-32 A flow diagram illustrating the proposed integrated approach comprising three main layers. The robot learns to reproduce primitive actions through imitation learning. It also learns the sequence of actions and identifies the constraints of the task using VSL. The symbolic planner is employed to solve new symbolic tasks and execute the plans. For more information, see Section 4.7.1. 214
4-33 The tutor is teaching the primitive actions including pull and push to the robot using kinesthetic teaching. For more information, see Section 4.7.3. 216
4-34 The recorded set of demonstrations for two sub-actions are shown in black. The reproduced trajectories from an arbitrary initial position towards the target (the green square) are shown in red. For more information, see Section 4.7.3. 218
4-35 By extracting the preconditions and effects of the primitive actions while executing them, the robot generalizes the learned sensorimotor skills as symbolic actions. For more information, see Section 4.7.4. 220
4-36 (a) and (b) are rectified pre-action and post-action observations; (c) is a rectified pre-action observation for the next operation; (d) is obtained by background subtraction between (a) and (b); (e) is obtained by background subtraction between (b) and (c); (f) is the result of detecting the border line for spatial reasoning. For more information, see Section 4.7.7. 223

4-37 The set of demonstrations by the tutor for a simple VSL task. The bottom row shows the workspace from the robot's point of view. For more information, see Section 4.7.7. 224
4-38 Starting from a new configuration, the robot reproduces the learned task using VSL. (a) and (c) show initial and final configurations. The bottom row shows the workspace from the robot's point of view. For more information, see Section 4.7.7. 224
4-39 Three different sets of demonstrations devised by the tutor for implying the rule of the game in the second experiment. For more information, see Section 4.7.7. 226
4-40 In the third experiment, using SP, the robot needs six sets of demonstrations for each of which the initial (a) and final (b) configurations of the objects are shown. While using VSL the robot requires a single demonstration for which the sequence of operations are shown (a-d). For more information, see Section 4.7.7. 227

List of Tables

2.1 Variants of the proposed reactive fuzzy system together with their inputs are listed. 55
2.2 The designing, tuning, and experimental results sections related to each variant of the RFDM system are reported. 56
2.3 Fuzzy Rule Base 64
2.4 RMSE Table 75

3.1 Policy selection as the value of β is changed. 135

4.1 Capabilities of VSL illustrated in each real-world experiment. 200
4.2 Execution time for different steps of VSL 201
4.3 The main differences between the 2D and 3D VSL approach. 207
4.4 The preconditions and effects extracted by VSL and written to the planner's domain file. For more information, see Section 4.7.4. 220

Acronyms

This section lists the main acronyms that are used throughout the thesis.

AUV	Autonomous Underwater Vehicle
CEM	Cross Entropy Method
CMA-ES	Covariance Matrix Adaptation - Evolution Strategy
DE	Differential Evolution
DMP	Dynamic Movement Primitives
DoF	Degrees of Freedom
IL	Imitation Learning
MODE	Multi-Objective Differential Evolution
MORL	Multi-Objective Reinforcement Learning
MP	Modified Price
RFDM	Reactive Fuzzy Decision Maker
RL	Reinforcement Learning
ROV	Remotely Operated Vehicle
SA	Simulated Annealing
SORL	Single-Objective Reinforcement Learning
VSL	Visuospatial Skill Learning

Mathematical Notation

The purpose of this section is to introduce the notation used in the subsequent parts of this thesis. The set of natural numbers is denoted by $\mathbb{N}$, while $\mathbb{R}$ denotes the set of real numbers. Vectors are denoted by lowercase bold Roman letters such as $\mathbf{x}$ and $\mathbf{y}$. All vectors are assumed to be column vectors. A superscript T denotes the transpose of a matrix or a vector, so that $\mathbf{x}^{\mathrm{T}}$ is a row vector. Scalar values and elements of a vector are denoted by normal italic letters such as x. Also, x_j represents the j^{th} element of the vector $\mathbf{x}$. A matrix is denoted by uppercase bold Roman letters $\mathbf{X}$ and x_{ij} denotes the element in the i^{th} row and j^{th} column.

"If I have seen further than others, it is by standing upon the shoulders of giants."

—Isaac Newton

1 Introduction

1.1 Motivations

NOWADAYS, autonomous robots are able to operate adequately in structured environments under well-controlled conditions. However, they quickly fail and need human assistance while facing uncertain and dynamically changing conditions in the real-world. Currently, the challenge of unstructured environments is receiving more attention from robotic researchers who aim to enhance the level of autonomy in robots. Autonomy refers to the ability to perform deliberate tasks based on sensing without human intervention. An autonomous robot should be capable of operating independently in unstructured environments without any form of external control for extended periods of time. Autonomy level for robots can be categorized into three subclasses comprising non-autonomous, semi-autonomous, and autonomous [1]. Non-autonomous robots are

entirely controlled by human operators e.g. through tele-operation. In such systems, the operator controls each movement manually. Semi-autonomous robots possess limited cognitive capabilities and to some extent are dependent on other agents. In this case, a human operator specifies high-level decisions and the robot generates low-level movements in order to achieve the goal of the desired task. Most of the existing robots can be categorized in this subclass. A mobile robot that follows a path devised by a human expert, while at the same time can avoid obstacles independently, is an instance of semi-autonomous robots. Finally, autonomous robots are systems with full cognitive capacities that can solve complex tasks independently. These systems are able to create, plan and execute a complete task without human intervention. They must be able not only to learn to deal with unexpected environmental conditions, but also to recognize when to update their learned models and how to learn in current situation.

The challenge of autonomy has been pursued in different robotic areas for instance for land robots [2], space robots [2, 3], underwater robots [4], and humanoids [2]. Recent DARPA Robotics Challenge[1] (DRC), which brought together the world's best robotics teams, aims at developing robotic systems capable of completing a series of challenging tasks related to disaster response in a human-engineered environment. The primary technical goal of the DRC is to improve the semi-autonomy level in robotic systems. During the DRC trials in 2013, the participating teams presented some of the most advanced robotic research to accomplish the desired tasks, and also the DRC finals will take place in June, 2015. Although the participating robots are equipped with innovative architectures, state-of-the-art algorithms and cutting-edge technology, the results implies that existing robotic platforms are not good at being autonomous. In other words, even though some robotic systems were able to successfully accomplish a number of the desired tasks, a closer inspection reveals that these systems are equipped with several task-specific behaviors designed particularly for the test environment and they still require human-assistance occasionally. Since the existing robots does not show outstanding

[1]DRC is a competition funded by the US Defence Advanced Research Project Agency, held from 2012 to 2015. It aims to develop semi-autonomous ground robots capable of performing complex tasks in human-engineered environments. The DRC trials was held in 2013 and the DRF finals will take place in June, 2015. For more information, see http://www.theroboticschallenge.org/ .

autonomy capabilities, easily fail and require assistance while operating in real-world environments, developing autonomous class systems is considered as the ultimate goal of robotic research. The present thesis pursues this goal by proposing novel robot learning approaches that elevate the robot autonomy. The proposed approaches address several important capabilities of autonomous systems by placing the focus on learning reactive and visuospatial skills. These skills are extremely vital for many aspects of autonomous systems. In addition, unlike most other capabilities of autonomous systems which are comprehensively studied, these skills are still under investigation and deserve further attention.

One of the most crucial capabilities that an autonomous system should possess is reactive behavior. By definition, reactive means tending to be responsive or to react to a stimulus. In physiology, a stimulus is a detectable change in internal or external environment that causes a response in a body part or organism. Sense organs are sensitive to stimuli and are responsible for detecting the changes. The corresponding response can be either a reaction or a reflex. The main question then becomes: What is the difference between reactive and reflexive behaviors.

In nature, a reflex is a spontaneous and unconscious response to a stimulus. For the movements that demand a very quick response, the signal is passed directly from a sensory neuron to a motor neuron. For instance, humans pull their hand away from a flame involuntarily. Reflexes are faster than reactions and are used to protect the system. A reaction, on the other hand, is conscious decision to a stimulus according to a plan. The reactive systems deliberate before acting. In this case, the signal is passed from a sensory neuron to the brain to be processed and the decision goes to a motor neuron for action.

The main class of primitive stimuli in animals includes finding food, moving away from predators, and finding mate. It can be seen that responding to such stimuli results in the animal's survival. Unlike biological systems, robots appear quite incapable of dealing with unpredictable and changing environments due to their lack of reactivity. Accordingly, an autonomous robot should be able to react appropriately to changing conditions in the environment. As previously stated, stimuli can also occur in the internal states of the system. Internal uncertainties such as failures and noise in the subsystems of the robot

can endanger both the robot and the mission. Therefore, an autonomous system should be able to learn a reactive behavior for dealing with such uncertainties.

> The first objective of this thesis is to enable robots to cope with uncertain and dynamically changing conditions by learning reactive behaviors.

To attain this objective, the present thesis proposes novel approaches for learning reactive behaviors that enable robots to deal with both internal and external stimuli. A novel hierarchical learning approach is proposed in Chapter 2 that allows a robot to react to external changing conditions while operating in an unstructured environment. The proposed approach which is tested on the challenging task of autonomous valve turning, consists of three main layers. Each layer realizes specific subtasks to improve the autonomy of the system. In the first layer, the robot acquires novel motor skills for approaching and grasping the valve through kinesthetic teaching. A hybrid force/motion control strategy is utilized in this layer to minimize the undesired forces and torques during the turning phase. A Reactive Fuzzy Decision Maker (RFDM) is devised in the second layer which reacts to relative movements, sensor delays and applied forces/torques according to the distance between the valve and the robot. The reactive layer modulates the robot's movement accordingly by changing the direction and rate of the movement. In the third layer, a supervised learning method is utilized to tune the behavior of the system based on expert knowledge. Conscious knowledge of a human expert is utilized to generate the fuzzy rule set. Then, the subconscious knowledge of the human expert is used to tune the generated rules. The proposed hierarchical learning approach is validated through real-world experiments both in laboratory and underwater environments. The results show the feasibility and efficiency of the proposed approach.

Autonomous robots should also react appropriately to failures and uncertainties which occur internally in the system itself. To achieve this, a learning framework is proposed in Chapter 3 that enables robots to learn a reactive behavior for dealing with internal failures. The proposed framework places the focus on improving the fault-tolerance and reliability of Autonomous Underwater Vehicles (AUVs). One of the most critical failures which can endanger both the underwater robot and the mission is thruster failure. Employing a

model-based direct policy search, the proposed learning framework discovers new control policies to overcome thruster failures as they happen. The policies are learned on a simulated model of the AUV on-board. When a fault is detected and isolated, the model is reconfigured according to the new condition. Since the learning framework generates an optimal trajectory, the learned fault-tolerant policy is able to navigate the AUV towards a specified target with minimum cost. The learned policy is then executed on the real robot in a closed-loop using the state feedback of the AUV. One of the advantages of the proposed framework is that it is applicable in both cases when the AUV becomes under-actuated in the presence of a fault or remains over-actuated. For dealing with multiple conflicting objectives, the proposed learning framework is extended by employing multi-objective reinforcement learning. The extended framework discovers a set of optimal solutions, each of which can be used to generate a trajectory that is able to navigate the AUV towards a specified target while satisfying multiple objectives. Unlike most existing methods which disregard the faulty thruster, the proposed framework can also deal with partially broken thrusters. The feasibility and efficiency of the proposed learning framework are validated through simulated and real-world experiments.

Autonomous robots should not only passively perceive the world but also actively change it. Therefore it is important to improve the robot autonomy in skill learning especially for manipulation tasks. Efficient skill learning capability helps the system to adapt to new conditions by learning a new task easily and effortlessly. In early skill learning approaches, a human expert models the task as accurately as possible and eliminates all uncertainties of the environment. Such process is equal to recording and replaying a trajectory in a pre-structured environment with precisely placed objects. In other words, those approaches are highly engineered and suitable for structured industrial and research environments. The main issue is that this process cannot handle changes. In other words, the whole process should be repeated or revised when a small change in the environment happens. To address this issue, several skill learning approaches based on human demonstrations have been proposed during the past decade [5]. These approaches that enable a robot to learn a skill to achieve the desired goal of the task, can be categorized in two main groups: trajectory-based and goal-based. In trajectory-based approaches, the

main focus is placed on learning and reproducing a desired trajectory, whereas goal-based approaches focus on learning the goal of the task. Each group has its own advantages and disadvantages. For instance trajectory-based approaches are suitable for learning primitive motor skills such as pick and place actions. Instead, in some tasks, they increase the complexity of the learning process unnecessarily. The goal-based techniques, on the other hand, are suitable for learning the main goal of the task regardless of the demonstrated trajectories. Most of goal-based approaches utilize visual perception as the main source of information. These approaches are called visual skill learning or learning by watching [6–8]. Over the past decade, with the advent of widely accessible and low-price powerful vision sensors, the interest around innovative visual learning approaches has considerably arisen. However, many existing visual learning approaches inherently comprise many steps that make the process complicated. Another drawback is that they require a significant amount of *a priori* knowledge to be manually engineered into the system. The level of autonomy in robots can be augmented by addressing such drawbacks.

The second objective of this thesis is to enable robots to efficiently learn new manipulation skills through visual perception.

To achieve this goal, a novel skill learning approach is proposed in Chapter 4 that allows a robot to acquire human-like visuospatial skills for object manipulation tasks. Visuospatial skills are attained by observing the spatial relationship among objects through demonstrations. Visuospatial Skill Learning (VSL) utilizes visual perception as the main source of information. As opposed to conventional trajectory-based learning approaches, VSL is a goal-based robot learning approach that focuses on achieving a desired goal configuration of objects relative to one another while maintaining the sequence of operations. VSL is capable of learning and generalizing multi-operation skills from a single demonstration, while requiring minimum *a priori* knowledge about the objects and the environment. In contrast to many existing approaches, VSL leverages simplicity, efficiency and user-friendly human-robot interaction. Several real-world experiments are conducted to show the validity of the proposed approach. In VSL, the primitive actions have to be programmed manually into the system. To handle

this drawback, VSL is further integrated with a conventional trajectory-based approach, imitation learning. The sensorimotor skills are learned through imitation learning. The sequence of performed actions is learned through demonstrations using VSL. The obtained integrated approach easily extends and adopts robot's capabilities to novel situations, even by users without programming ability. Furthermore, VSL utilizes its own simple internal planner in the reproduction phase. To utilize the advantages of standard symbolic planners, the original planner is replaced with an existing action-level symbolic planner. In order to ground the actions, the symbolic actions are defined in the planner and VSL maps identified preconditions and effects in a formalism suitable to be used at the symbolic level. The experimental results show the improvement in generalization to find a solution that can be acquired from both multiple and single demonstrations.

After reviewing the research motivations and describing the objectives, the main goal of this thesis is stated here as:

> Increasing the robot autonomy in real-world environments by developing innovative robot learning approaches for acquiring reactive behaviors and visuospatial skills.

1.2 Contributions

All the approaches presented in this thesis are validated through various simulated and real-world experiments. For the real-world experiments, a number of robotic platforms have been employed including: a 7-DoF Kuka robotic manipulator, a 7-DoF Barrett WAM equipped with a Barrett Hand, a Girona500 AUV equipped with a 4-DoF robotic arm, and an iCub humanoid robot. During the development of the objectives and the goal of this thesis, several research contributions are achieved that are listed below.

Contributions of Chapter 2

- A hierarchical learning approach is proposed that enables a robot to acquire a reactive behavior for coping with uncertainties. The approach is devised for

performing the challenging task of autonomous robotic valve turning. The proposed hierarchical architecture consists of three main layers each of which realizes specific subtasks to improve the autonomy of the system.

- The robot acquires novel motor skills for approaching and grasping the valve through kinesthetic teaching based on imitation learning. In addition, a hybrid force/motion control strategy is utilized to dissipate undesired forces and torques during the turning phase. Integration of the imitation learning technique with a hybrid control strategy is one of the contributions of this thesis.

- A reactive fuzzy decision maker system is devised that modulates the movements of the robot during the execution of the task by observing the uncertainties and disturbances. The reactive system affects the behavior of the robot by modulating the rate and the direction of the movement. The proposed reactive system enables the robot to deal with uncertainties in the environment by learning from human knowledge. Conscious knowledge of a human expert is utilized to devise the fuzzy rule set. Whereas, the subconscious knowledge of the human expert is used to tune the rules.

- Autonomous valve turning is a challenging task that many robotic groups are trying to accomplish. One of the contributions of this thesis is achieving autonomous valve turning in underwater environment by employing the proposed hierarchical learning approach. The reactive behavior of the robot during the execution of the task evidently plays a crucial role that results in elevating the autonomy of the system.

Contributions of Chapter 3

- To address the problem of dealing with changes in the internal states of the system, a novel learning framework is proposed that enables a robot to learn a reactive behavior. The proposed framework focuses on elevating fault-tolerance and reliability in AUVs. Using a model-based direct policy search, new control policies

are discovered to overcome thruster failures as they happen. The policies are learned on a simulated model of the AUV on-board. The model is adapted according to the new condition when a fault is detected and isolated. The learned policy is then executed on the real robot in a closed-loop using the state feedback of the AUV. One of the advantages of this approach over existing approaches is that it is applicable in both cases which the robot becomes under-actuated or remains over-actuated in the presence of the failure. Most existing approaches have been developed for over-actuated systems and there are few cases dealing with under-actuated scenarios which are not applicable when the system remains over-actuated. In addition, the proposed 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. Unlike most existing methods which disregard the faulty thruster, the proposed framework can also deal with partially broken thrusters to increase the autonomy of the AUV.

- In underwater domain, there are few cases involving online learning of a behavior. Demonstrating the feasibility of the proposed learning framework in real-world experiments therefore, is an important contribution. In other words, using a state-dependent policy and applying a closed-loop fault-tolerant control with state feedbacks, an optimal policy can be computed on-board and executed online. So, contrary to many existing methods, the proposed framework generates an optimal policy that can be directly utilized by the AUV in real-world scenarios. This capability also reduces the involvement of human supervisor in the level of trajectory generation, and consequently improves the autonomy of the system.

- To deal with multiple objectives which can be complementary, conflicting, or independent, two variants of the proposed frameworks are developed. The first variant utilizes linear scalarization technique which is suitable for independent and complementary objectives. The second variant, on the other hand, utilizes a multi-objective algorithm that can deal with multiple conflicting objectives by discovering multiple optimal solutions.

- The proposed learning framework can be generalized and implemented in various underwater vehicles and it is not developed exclusively for the Girona500 AUV. The reason is that the theoretical aspect of the approach 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.

Contributions of Chapter 4

- One of the contributions of this thesis is the Visuospatial Skill Learning approach which allows a robot to acquire new visuospatial skills for object manipulation by observing a demonstration. VSL utilizes visual perception as the main source of information. As opposed to conventional trajectory-based learning approaches, VSL is a goal-based robot learning approach that focuses on achieving a desired goal configuration of objects relative to one another while maintaining the sequence of operations. VSL is capable of learning and generalizing multi-operation skills from a single demonstration, while requiring minimum *a priori* knowledge about the objects and the environment. In contrast to many existing approaches, VSL leverages simplicity, efficiency and user-friendly human-robot interaction.

- In VSL, the primitive actions have to be programmed manually into the system. To overcome this drawback, VSL is integrated with a conventional trajectory-based approach, imitation learning. Thus, the robot firstly learns the required primitive actions through kinesthetic teaching. And then it learns the sequence of actions to reach the desired goal of the task through VSL. The obtained approach easily extends and adopts robot's capabilities to novel situations, even by users without programming ability.

- VSL has its own simple internal planner in the reproduction phase. To utilize the advantages of standard symbolic planners, the original planner is replaced with an existing action-level symbolic planner. In order to ground the actions, the symbolic actions are defined in the planner and VSL maps identified preconditions and effects in a formalism suitable to be used at the symbolic level. The integration of VSL and

SP is a substantial contribution of this thesis. The coupling between VSL and SP is natural and intuitive and brings significant advantages over using VSL separately. Learning preconditions and effects of the actions in VSL approach and making symbolic plans based on the learned knowledge deals with the long-standing important problem of Artificial Intelligence, namely Symbol Grounding. The feasibility and capability of the proposed framework are experimentally validated. In the experiments, it is shown that the symbols and rules are grounded through learning in the problem domain.

- By integrating Imitation Learning (IL), Visuospatial Skill Learning (VSL), and conventional planning methods a novel learning framework is developed. In this framework, the sensorimotor skills are learned through IL. The sequence of performed actions is learned through demonstrations using VSL. A standard action-level planner is used to represent a symbolic description of the skill, which allows the system to represent the skill in a discrete, symbolic form. VSL identifies the underlying constraints of the task and extracts symbolic predicates, thereby updating the planner representation while the skills are being learned. Therefore the planner maintains a generalized representation of each skill as a reusable action, which can be planned and performed independently during the learning phase. The feasibility and efficiency of the integrated framework are validated through real-world experiments

1.3 Thesis Outline

The contents of this thesis can be classified in two main parts. The first part introduces solutions for dealing with internal and external stimuli that can endanger both the robot and the mission. This part proposes novel approaches for learning reactive behaviors that can comply with uncertainties and changing conditions in environment for different tasks. The main objective of the proposed approaches is to enhance the robot autonomy, thereby increasing the performance and the efficiency of the system while operating in an unstructured and unpredictable environment in extended periods of time.

The second part introduces a novel skill learning approach for object manipulation tasks that allows a robot to interact with its environment and change it. The robot learns human-like visuospatial skills through visual perception by observing human demonstrations. Since it learns by observing visuospatial representations, the approach is called Visuospatial Skill Learning (VSL). The main advantage of VSL is that contrasting to several existing techniques, it allows a robot to learn new visuospatial skills efficiently and effortlessly. VSL is capable of learning and generalizing different skills while requiring minimum *a priori* knowledge about the objects and the environment. A brief description of each chapter of this thesis is presented next.

Chapter 2 - Learning Reactive Behavior: A Hierarchical Learning Approach

This chapter presents a hierarchical learning approach for the challenging task of autonomous robotic valve turning emphasising the fact that the robot needs to react adaptively to the external stimuli in the environment whether it is performing in a laboratory environment or underwater. The robot learns a reactive behavior that augments the autonomy of the system by coping with uncertainties in a dynamic environment while performing the autonomous valve turning task. After a discussion of some related work, the methodology and the architecture of the proposed approach are described. Then, the kinesthetic teaching method based on imitation learning is discussed. In the next step, the hybrid force/motion control strategy which is used for learning the turning phase, is explained. In this chapter, three variants of the proposed reactive system are developed. For each variant, the designing and tuning steps are described and related experimental results conducted in laboratory and underwater environments are reported.

Chapter 3 - Learning Reactive Behavior: A Fault-Tolerant Approach

This chapter presents a reactive learning approach for dealing with internal stimuli which affect the functionality of the robot while performing a task. The chapter places the focus on developing a learning framework for dealing with thruster failures in AUVs. The reactive learning framework is able to discover fault-tolerant policies to recover from failures as

they happen. The proposed framework has been tested on an autonomous underwater robot facing a thruster failure in which the original control system cannot recover the robot. After reviewing related work, the methodology and the main components of the proposed framework based on Single-Objective Reinforcement Learning (SORL) and linear scalarization techniques are described. The conducted simulated and real-world experiments for a scalarized objective function are reported. Then, an extension of the presented framework is developed which utilizes Multi-Objective Reinforcement Learning (MORL) in order to deal with conflicting multiple objectives. The experimental results for MORL are reported.

Chapter 4 - Visuospatial Skill Learning

This chapter introduces a novel skill learning approach for learning visuospatial skills based on human demonstration. The approach which is called Visuospatial Skill Learning (VSL) allows a robot to acquire new visuospatial skills by observing a sequence of operations demonstrated by a tutor. After a discussion of some related work, the terminology and methodology of the VSL approach are explained in details. Afterwards, implementation steps for the VSL approach are described. Results for conducted simulated and real-world experiments are reported. The VSL approach is then extended to 3D and experimental results for 3D real-world experiments are reported. Afterward, VSL is used for learning symbolic representation of actions and a novel learning framework is proposed. The implementation steps for the proposed framework are discussed and experimental results using the proposed framework are reported.

Chapter 5 - Conclusions

This chapter summarizes the thesis by reviewing the contents described in the core chapters. It points out the novelties and the contributions extracted from each chapter individually. It also makes a few remarks on the possible research extensions.

Appendices

Throughout the evolution of this thesis a number of optimization algorithms have been employed. Since the algorithms are not developed by the author, they are not includes in the core chapters of the thesis. Instead a number of appendices are attached at the end of the thesis that provide a brief description and a pseudocode for each algorithm. Since, most of the material presented in the core chapters of the thesis have been published in peer-reviewed conference proceedings and scientific journals, a list of publications is provided at the end of the thesis. Furthermore, related videos of the robot experiments that are reported in the thesis are available online at my personal website[2].

[2]My personal website http://www.ahmadzadeh.info. The videos are also available at my supervisor's website at: http://www.kormushev.com.

"The only thing that makes life possible is permanent, intolerable uncertainty: not knowing what comes next."

—Ursula K. Le Guin

2

Learning Reactive Behavior: A Hierarchical Approach

2.1 Motivation

In artificial intelligence, learning reactive behavior which is also known as reactive planning denotes a category of methods for action selection by autonomous agents. In contrast to classical methods, reactive approaches can cope with highly dynamic and unpredictable environments. There are several methods for representing a reactive behavior including if-then rules, finite state machines [9], and fuzzy approaches [10].

In robotics, learning reactive behavior or reactive planning approaches have been mostly developed for obstacle avoidance and steering tasks in autonomous robot navigation [10–12]. Furthermore, there exist several methods applied to the problem

of reactive path planning in robot manipulators [13, 14]. Performing robotic tasks in uncertain environments, almost all robotic platforms necessitate reactiveness in order to be able to demonstrate autonomous capabilities. Robotic valve turning is one of the most challenging tasks specially in unstructured environments with increasing level of uncertainty [15–19]. 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. 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, thereby introducing 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.

With the advent of robust guidance systems, advanced processing capabilities and high capacity lightweight batteries, nowadays, AUV robots are becoming the platform of choice for conducting underwater missions with ever increasing duration and complexity. Implementing innovative machine learning methods enhances capabilities such as operating under extreme uncertainty, interacting with highly unexpected underwater environment autonomously, learning from sensor data continuously, and re-planning the mission optimally. The European project PANDORA[1], aims to make underwater robot persistently autonomous by developing and evaluating new computational methods [4]. Autonomous grasping and turning a valve is one of the most challenging tasks which has been defined in PANDORA. The learning approach proposed in this chapter[2] focuses on learning reactive robot behaviors to deal with the challenging task of autonomous robotic valve turning firstly in a laboratory and finally in underwater environment. A hierarchical learning approach is proposed to resolve this challenge. Kinesthetic teaching

[1] PANDORA is an EU FP7-ICT STREP research project between 5 universities across Europe in response to Challenge 2 of the Cognitive Systems and Robotics call. PANDORA stands for 'Persistent Autonomy through learNing, aDaptation, Observation and Re-plAnning'. For more information, check the website of the project: http://persistentautonomy.com/.

[2] The main results of this chapter were previously published in [20–22].

based on imitation learning is used to learn and reproduce the reaching and grasping skills. The robot detects the valve using visual perception and approaches it along the produced trajectory by the learning module. In order to increase the autonomy of the system, a reactive fuzzy decision maker system is developed. This module evaluates the dynamic behavior of the system and modulates the robot's movements accordingly. In this chapter, three variants of the reactive fuzzy decision maker are gradually devised. Each of these variants is developed with a different fuzzy structure according to various accessible source of information. The proposed reactive system is tuned using optimization algorithms by expert knowledge.

The rest of the chapter is organized as follows. Related work is reviewed in Section 2.2. The methodology and the architecture of the proposed approach are described in Section 2.3. The kinesthetic teaching method consisting of demonstration, learning and reproduction phases is discussed in Section 2.4. The designing and tuning steps of the first variant of the reactive system are explained in Section 2.5 and experimental results using the proposed hierarchical approach are reported in Section 2.6. A hybrid force/motion control strategy is used for learning the turning phase and is explained in Section 2.7. The designing and tuning steps of the second variant of the reactive system are explained in Section 2.8 and experimental results using the proposed hierarchical approach are reported in Section 2.9. The experimental results in underwater environment are reported in Section 2.11. And finally, conclusions of the research are drawn in Section 2.12.

Figure 2-1: Autonomous Robotic Valve Turning.

2.2 Related Work

Learning reactive behavior in robot manipulators has been investigated by several groups [3]. A reflexive collision avoidance technique consisting of four control layers was proposed by Wikman and Newman in which two layers are used to avoid overshooting and the other two layers deal with the collision avoidance task using online inspection of configuration space [23]. Khatib proposed an obstacle avoidance approach for mobile robots and manipulators based on the artificial potential field concept [24]. The method is also extended for dealing with moving obstacles using a time-varying potential field. The manipulator control problem is reformulated as direct control of manipulator motion in operational space rather than joint space. In order to detect moving obstacles a visual sensor is used. The proposed approach in this chapter, on the other hand, performs independently without modifying the robot's original control system.

A collision avoidance approach for kinematically redundant manipulators was proposed by Glass *et al.*, by defining a set of task-space inequality constraints and ensure that the constraints are satisfied in control-loop level [25]. Zhang and Wang proposed a similar approach using a recurrent neural network for finding an optimal solution to the kinematic control of redundant manipulators [26]. Based on physical quantities such as total energy and generalized momentum of the robot manipulator, a collision detection method was proposed by De Luca *et al.* that uses only proprioceptive robot sensors and provides also directional information for a safe robot reaction after collision [27]. In the conducted experiments, the method was applied to both rigid arms and arms with flexible joints.

Yoshida *et al.*, developed a reactive motion planning method that is capable of online replanning when the environment changes [13, 28]. To deal with incomplete perception, their method utilizes *roadmap* concept which is a graph whose nodes are collision free configurations of the robot. Graph search methods are used to find a collision free path from initial to goal configurations. In their work, valid collision-free edges in the *roadmap* are stored and updated continuously throughout the planning process. The trajectory

[3] Since the desired goal of this chapter is learning reactive behaviors for challenging task of robotic valve turning, related work on the obstacle avoidance for mobile robots is not mentioned.

smoothing is achieved by adding a short collision-free connecting edge between the current and next configuration nodes. One of the difficult situations for the proposed method occurs when obstacles move continuously. In addition, due to rapid obstacle motion, the replanning process may not finish before reaching the newly estimated point which means collision avoidance is not guaranteed. In this sense, their proposed method is more suitable for the environmental changes that are discrete and gradual with respect to the robot motion and the planning capacity.

Luo *et al.*, proposed a repulsive reaction vector generator for obstacle avoidance in manipulators that uses arm angle as a redundant parameter to generate a vector including position, orientation, and arm angle [14]. The manipulator avoids the obstacle by rotating the arm plane. In their conducted experiments, the input sensor was a Microsoft Kinect and the NTU-iCeiRa arm was employed. After calculating the reaction vector and the arm plane, a trajectory generation technique based on analytical inverse kinematics is employed.

Robotic valve manipulation contains a number of complex and challenging subtasks. Consequently, 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. Abidi *et al.,* tried to achieve inspection and manipulation capabilities in a semi-autonomous operation of a control panel in a nuclear power plant [15]. A 6 DoF industrial robot equipped with a number of sensors (e.g., vision, range, sound, proximity, force/torque, and touch) was used in the conducted experiments. 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 [16], 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 their next research [29], Anisi *et*

al., developed the valve manipulation task in 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 [29] 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. A framework for valve turning was proposed in [17] 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 was proposed in [18]. 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 was devised in [19] that employs the upper body kinematics and dynamics of a humanoid robot for reaching and turning a valve.

In underwater domain, still Remotely Operated underwater Vehicles (ROV) are being used due to the complexity and uncertainty of the environment. These ROVs are operated by one or two skilled operators, usually one keeps the robot stable while the other controls the manipulator [30].

Machine learning methods have received special attention in recent years. The key idea is that, although the above mentioned approaches cannot deal with unknown dynamic environments (e.g., underwater environment), using new learning methods the robot can acquire new skills to overcome the uncertainties presented by the real world. Some skills can be successfully transferred to the robot using imitation strategies [31, 32] the others can be learned very efficiently by the robot using reinforcement learning [33]. This realization leads further developments in autonomous AUV valve turning which is a focus of this chapter [20–22]. In order to achieve autonomous valve turning, in this chapter a hierarchical learning approach is proposed that enables a robot manipulator to reach, grasp and turn a valve while reacting to uncertainties reactively.

The robot acquires the reaching and grasping skills through kinesthetic teaching based on imitation learning [32]. A hybrid force/motion control strategy is devised to accomplish the turning phase. In order to increase the autonomy of the system a

reactive fuzzy decision maker is proposed that learns a reactive behavior using expert knowledge. The reactive layer evaluates the dynamic behavior of the system and modulates the robot's movements accordingly. The reactive system deals with uncertainty caused by disturbances, noise, and sudden movements from environment. The reactive system is tuned based on expert knowledge. The validity and performance of the proposed approach were successfully demonstrated through several real-world valve turning experiments both in a laboratory and in underwater environments. It is shown that learning such reactive behavior effectively improves the efficiency and autonomy of the system specially in underwater valve turning scenario.

2.3 Methodology

The proposed approach is organized as a hierarchical architecture with three different layers which are illustrated as a high-level outline in Figure 2-2. Each layer realizes specific subtasks to improve the 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 manipulator to obtain new motor skills through kinesthetic teaching [34]. Imitation learning [32] approach which is designed specially to learn trajectory-based tasks, is a promising choice to learn the reaching motor skill. In order to reproduce the reaching skill towards the target, the robot needs to detect the target by utilizing feedbacks from a pose estimation module.

In contrast to conventional trajectory generation algorithms and path planning techniques that require users to manually program a specific skill into the system, imitation learning enables a robot to acquire new skills by observing several demonstrations through kinesthetic teaching. For instance, the tutor can demonstrate the skill while holding and moving the arm. The observations are recorded and used to reproduce the skill by the robot. The learned skills can easily be extended and adapted to novel situations even by non-expert users. This layer is also responsible for performing the turning phase after the reaching and grasping phase is accomplished. The turning phase is accomplished firstly by pre-programming the robot and finally by implementing a hybrid force/motion con-

troller. Further details are given later in this section.

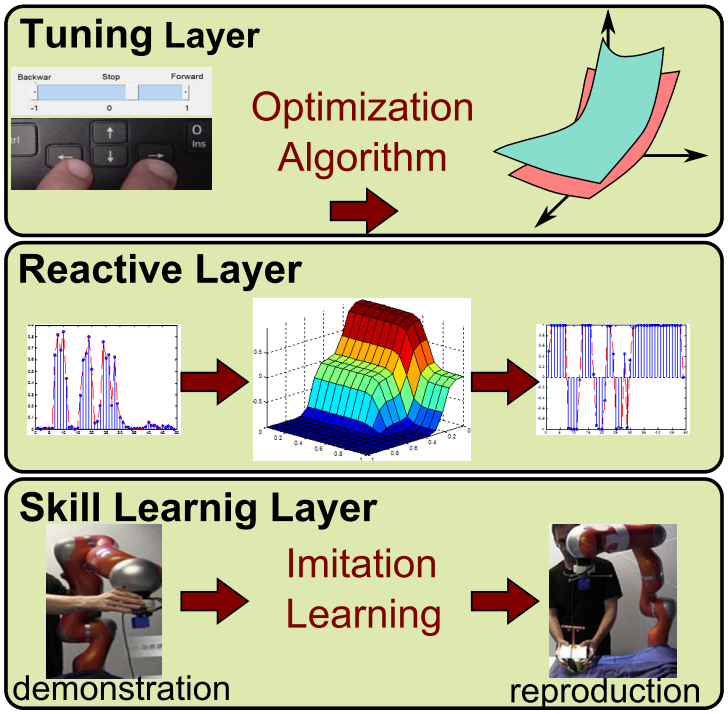


Figure 2-2: A high-level diagram illustrating the three layers of the proposed hierarchical learning approach. For more information, see Section 2.3.

In order to develop an autonomous system, the robot needs to deal with uncertainties. The disturbances during the execution of the task are monitored and handled by a Reactive Fuzzy Decision Maker (RFDM) which is placed in the middle layer of the proposed architecture. Although such reactive system can be implemented using a thresholding method (e.g. if-then rules), fuzzy systems are chosen. The reason is that fuzzy systems provide a continuous decision surface and infer from a set of human-defined linguistic rules. The qualitative nature of fuzzy logic makes it an adequate tool to address the problem of autonomous valve turning in uncertain environments where prior knowledge about the environment is incomplete, uncertain, and approximate. For instance, perceptually acquired information is usually unreliable. In addition, real-world environments typically have complex and unpredictable dynamics. Moreover, the transition in resulted behavior is smoother in fuzzy approaches because the conditions, states and actions are no more boolean or “yes/no” respectively but are approximate and smooth.

In the proposed approach, the rule-base of the reactive fuzzy system is designed using conscious knowledge of a human expert. Parameters of the system, however, are tuned using optimization algorithms based on unconscious knowledge of human expert. The reactive system modulates the behavior of the robot according to the acquired information from the environment without generating or modifying the trajectory generated by the first layer.

During the execution of the task, the RFDM module monitors the changes in the environments (i.e. uncertainties) and generates decisions that regulate the movements of the robot. 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.

The final goal of this research is to develop a reactive learning approach for dealing with the challenging task of valve turning eventually in underwater environment. Since real-world environments typically have complex and unpredictable dynamics, the appropriate sources of uncertainty should be considered while designing the system. One of the main factors that cause uncertainty during the process of valve turning is the relative movement between the robot and the valve due to existing disturbances in the environment, underwater currents, or sudden movements of the robot. Therefore, the relative movement which is extracted by detecting the position of the valve and calculating the relative relocation between the valve and the robot's gripper, is considered as one of the inputs for the reactive system.

During the execution of the task, the system should estimate and update the pose of the target continuously. In the real-world, the pose estimation process is accomplished using a vision sensor which is unreliable and prune to noise. In addition, the vision system may be occluded and thus introduce a delay in updating the pose. Thus, the sensor delay is considered as the second factor of uncertainty.

As previously mentioned, three variants of the proposed reactive system are developed in this chapter. Each variant contains different inputs and varied structure. Based on the two above-mentioned inputs, the first variant of the RFDM system, entitled 'RFDM/1', is designed in Section 2.5.1 and tuned in Section 2.5.2. Figure 2-3 illustrates different

components of the proposed hierarchical system including RFDM/1. In order to validate the proposed reactive system, the first set of experiments are conducted in the lab environment and the results are reported in Section 2.6. In this set of experiment, two decisions were made. First, the base of the robot is kept fixed while the valve can be moved manually by a tutor. By such configuration, the efficiency and validity of the imitation learning layer can be tested, because the tutor can move and replace the valve multiple times and the robot manipulator should follow the target until the reaching skill is accomplished. And also, the tutor can easily control the movements of the valve and test the reactive system under various disturbances (i.e. by controlling the amplitude of the movement while oscillating the valve). Secondly, in order to capture the real-time 3D position and orientation of the valve and the gripper an Optitrack system was utilized. The Optitrack system includes a number of motion capture cameras and a set of markers. Despite the fact that the Optitrack system cannot be employed in outdoor environment, the pose estimation process in the first set of the experiments, is very precise and allows the tutor to control the magnitude of the noise in the system. It is worth noting that during the first set of experiments, the reactive system includes the two above mentioned inputs and after the reproduction of the reaching skill is accomplished, the turning action is performed by pre-programming the robot.

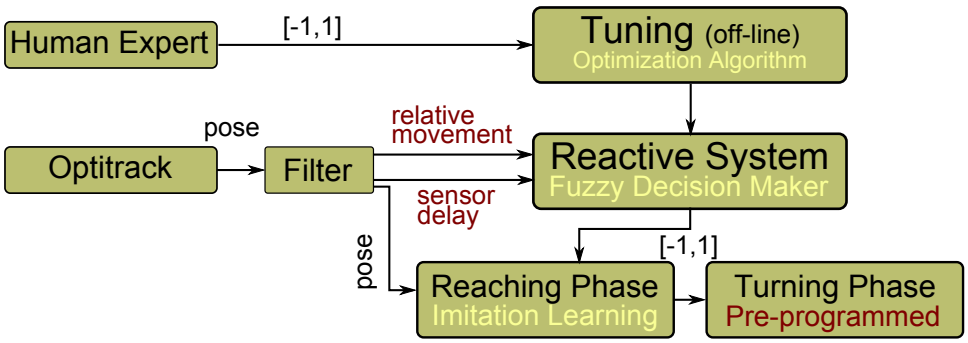


Figure 2-3: A high-level flow diagram illustrating the different components of the proposed hierarchical system including RFDM/1. For more information, see Section 2.3.

In order to increase the autonomy of the system, the turning phase should be improved. To achieve this goal, in the next step a hybrid force/motion control strategy is designed to handle the turning phase. 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. During the grasping and turning phase the robot may exert huge forces/torques on the valve which is another hazardous and highly probable situation. In such case, 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. Consequently, the second variant of the reactive system, entitled 'RFDM/2', which is designed in Section 2.8.1, additionally observes the amount of force/torque applied to the valve. Figure 2-4 illustrates different components of the proposed hierarchical system including RFDM/2. Moreover, in the second set of experiments, the valve maintained fixed while the base of the robot was placed on a movable table that allows a tutor to simulate the movements of the robot caused by underwater currents and disturbances easily by oscillating the table. The Optitrack system is replaced with a vision sensor (i.e. an RGB-D camera) similar to the one of the real AUV that detects the pose of the valve using an AR marker. During the experiments, after the valve is detected by the vision sensor, the first layer executes the reaching skill and the arm starts extracting towards the valve. The tutor can generate oscillations and disturbances by moving the base of the robot. These movements are estimated by the vision system and reported as the relative movement between the valve and the gripper to the reactive layer. To increase the reliability and autonomy of the whole system during the execution of the task, the reactive system modulates the behavior of the system accordingly

One of the shortcoming of the RFDM/1 (see Section 2.5.1) is that its behavior to the uncertainties remain unchanged regardless of the distance between the robot and the valve. For instance, consider the end-effector is far from the valve and is reaching it. In such case, if the uncertainty increases say due to growing relative movement between the gripper and the valve, the arm still can continue reaching the valve. In the vicinity of the valve, on the other hand, the system must behave very carefully and take into consideration smaller oscillations more importantly. Such behavior demands a system with more adaptive behavior with respect to the distance between the gripper and the

valve. The distance is extracted by the vision sensor. Considering the distance as another input to the reactive system, the adaptiveness, the safety and the autonomy of the system is improved. Therefore, the reactive system designed and employed in the second set of experiments, RFDM/2, consists of three inputs namely the relative movement, the force/torque magnitude, and the distance between the gripper and the valve. It is worth noting that, in order to decrease the number of variables in the system that facilitates the analysis of the system, the sensor delay has not been considered as one of the inputs in RFDM/2. However, normal delays from the vision sensor affect the process naturally. The results are reported in Section 2.9.

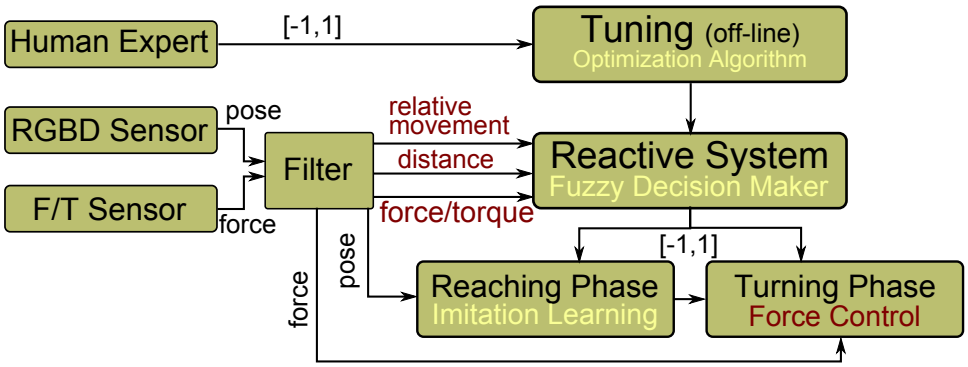


Figure 2-4: A high-level flow diagram illustrating the different components of the proposed hierarchical system including RFDM/2. For more information, see Section 2.3.

Both sets of experiments using RFDM/1 and RFDM/2 are conducted in the laboratory environment. To accomplish the autonomous robotic valve turning task in underwater environment, an AUV equipped with a robotic arm and a vision sensor is employed. The vision sensor extracts the pose of the valve which is fixed on a panel in a pool. The pose estimation process is obviously noisy due to deficient visibility in underwater environment. So sensor delay must be considered as one of the inputs to the reactive system. The relative movement and the distance between the gripper and the valve are two other inputs for the system. It is worth noting that, the robotic arm utilized in the underwater experiments lacks a F/T sensor, so the force/torque data input was excluded from the reactive system. Thus, the designed reactive system for underwater experiments, entitled 'RFDM/3', includes three inputs namely, the relative movement, the distance, and the sensor delay. Figure 2-5 illustrates different components of the proposed hierarchical

system including RFDM/2.

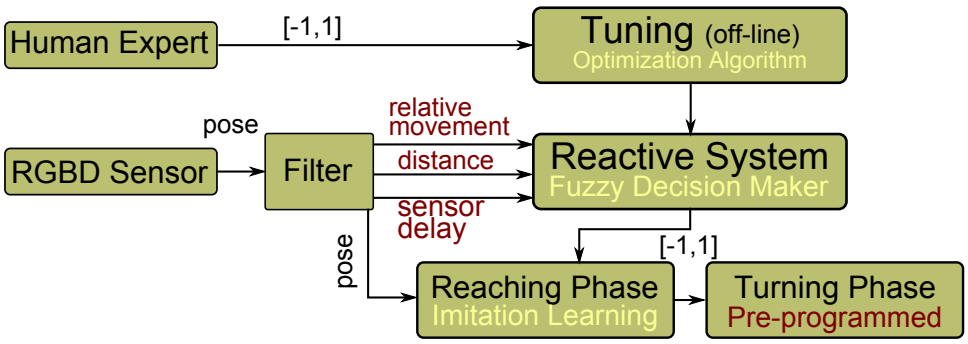


Figure 2-5: A high-level flow diagram illustrating the different components of the proposed hierarchical system including RFDM/3. For more information, see Section 2.3.

In all three variants of the RFDM system, the output is a real number in range $[-1, 1]$ that is used to modulate the skill reproduction process. For instance, -1, 0, 0.2, 1 mean go backward with 100% rate, halt, go forward with 20% rate, and go forward with 100% rate respectively. Further details about the reactive system can be found in Sections 2.5, 2.8 and 2.10. Different variants of the reactive system designed and used in this chapter together with their input variables are reported in Table 2.1. In addition, Table 2.2 reports the sections in which the variants of the RFDM system are designed, tuned and tested in real-world experiments.

Table 2.1: Variants of the proposed reactive fuzzy system together with their inputs are listed.

	Inputs			
	Relative Movement	Sensor Delay	F/T data	Distance to target
RFDM/1	✓	✓	–	–
RFDM/2	✓	–	✓	✓
RFDM/3	✓	✓	–	✓

Table 2.2: The designing, tuning, and experimental results sections related to each variant of the RFDM system are reported.

	Section		
	Design	**Tuning**	**Experiment**
RFDM/1	2.5.1	2.5.2	2.6
RFDM/2	2.8.1	2.8.2	2.9
RFDM/3	2.10.1	2.10.2	2.11

2.4 Imitation Learning

Imitation learning enables manipulators to learn and reproduce trajectory-based skills from a set of demonstrations [35]. The demonstrations are provided either by teleoperation or through kinesthetic teaching. There are several representations developed for learning and reproducing trajectory-based skills.

Gaussina Mixture Models (GMM) and Gaussian Mixture Regression (GMR) are used to encode demonstrated trajectories [36, 37] based on Gaussian Process (GP). Using the change in variability between demonstrations over time, the main features of a task can be extracted.

A learning from demonstration approach based on averaging trajectories, called LAT, has been proposed by Reiner *et al.* [38]. The demonstrated trajectories are firstly aligned over time and transformed into the target's frame of reference. The means and standard deviations for every dimensions and targets are extracted in both task-space and joint-space. During the reproduction phase, a one-dimensional normal distribution for each task-space coordinate is averaged to approximate at any time step. The final trajectory is generated in target's frame.

One of the most widely-used representations for encoding and generating trajectory-based skills is Dynamical Movement Primitives (DMP) [39]. A task-space movement is generated by integrating a set of differential equations which can be interpreted as a damped linear spring system perturbed by an external force term. Gaussian basis functions are used to approximate the reproduced trajectory and linear regression is used for finding the weights of the basis functions.

Among the mentioned representations, LAT cannot deal with the demonstrations in

which the targets are aligned parallel to one Cartesian coordinate axis, but not in the reproduction phase. This issue is similarly valid for GP. In other words, Both methods cannot extract constraints from the demonstrations with objects aligned parallel to a Cartesian coordinate axis. GP are able of handling demonstrations including few data points, whereas, LAT needs many data points to generate a smooth trajectory. Another disadvantages of both GP and LAT is that unlike DMP they cannot reproduce trajectories while the target is moving.

In the proposed architecture, DMP allows the robot to learn a compact representation of the reaching skill using the recorded demonstrations. A number of improved extensions of DMP has been proposed [34, 40]. In this section, the extended DMP approach proposed in [34] is exploited which also encapsulates variation and correlation information of the demonstrated skill as a mixture of dynamical systems.

As shown in Figure 2-6, the input to the system is the pose of the gripper of the robot and the pose of the target which is the valve. The extracted pose information is recorded as trajectories during the demonstration phase. These demonstrations are used to learn the skill.

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 primitive 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.

Figure 2-6: A high-level diagram illustrating the imitation learning layer in the proposed hierarchical approach. For more information, see Section 2.4.

During the demonstration phase, multiple desired trajectories are demonstrated by a

human tutor through kinesthetic teaching. Each demonstration $m \in \{1,\dots,M\}$ consists of a set of T_m positions $\mathbf{x}$, velocities $\dot{\mathbf{x}}$, and accelerations $\ddot{\mathbf{x}}$, of the end-effector in Cartesian space where $\mathbf{x} \in \mathbb{R}^3$. 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:

$$\hat{\ddot{\mathbf{x}}} = \sum_{i=1}^{L} h_i(t)[\mathbf{K}_i^P(\mu_i^x - \mathbf{x}) - k^v \dot{\mathbf{x}}], \tag{2.1}$$

where $\hat{\ddot{\mathbf{x}}}$ is the desired acceleration, $\mathbf{K}_i^P$ 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.

During the demonstration phase, the trajectories are recorded independent of the explicit time. Instead, in order to create an implicit time-dependency,

$$t = \frac{-ln(s)}{\alpha}, \tag{2.2}$$

a canonical system is defined as follows:

$$\dot{s} = -\alpha s \tag{2.3}$$

where s is the decay term initialized by $s = 1$ that monotonically converges to 0. Furthermore, a set of Gaussian basis functions are defined as $\mathcal{N}(\mu_i^T, \Sigma_i^T)$ in time space, where the centers μ_i^T are equally distributed in time and the variance parameters Σ_i^T are set to a constant value inversely proportional to the number of states. α is a fixed value which depends on the duration of the demonstrations.

By determining the weighting functions $h_i(t)$ through the decay term s, the system sequentially converges to the set of attractors. Stiffness matrices $\mathbf{K}_i^P$ 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.

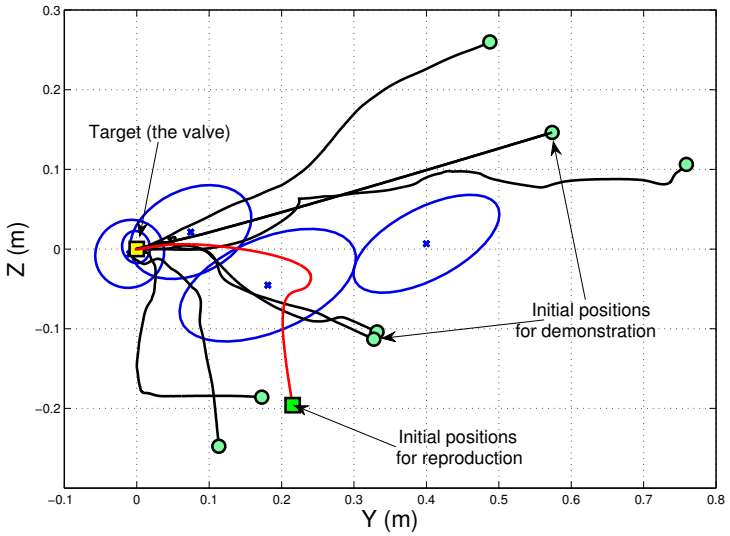
The recorded set of demonstrations is depicted as black curves in Figure 2-7. Following the described approach, the system learns a set of attractors which can be seen in the 2D

plots in Figures 2-7a and 2-8a 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. Each red trajectory in Figures 2-7 and 2-8 illustrates a reproduction. In both figures the goal, (the valve), is shown in yellow. A snapshot of the reaching skill reproduced by the robot is shown in Figure 2-9.

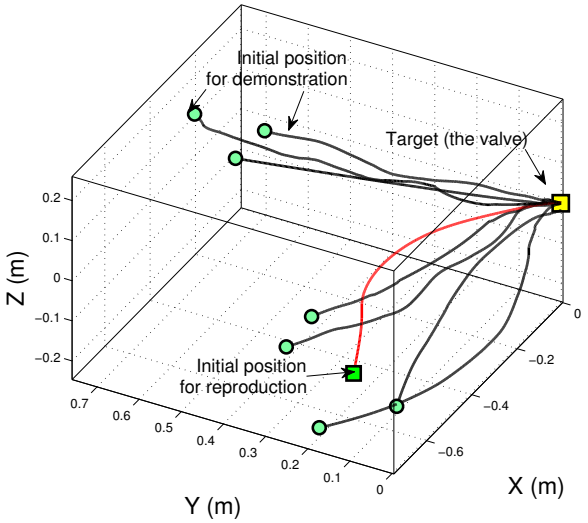
In this thesis, a new capability of the extended DMP based on the implicit timing is introduced that leads to a reversible behavior of the system. This new capability enables the robot to perform two behaviors: (i) reactive behavior by switching the direction of the movement towards the target or away 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 (2.3) to (2.4):

$$t = t_{\text{final}} + \frac{l\, n(s)}{\alpha}. \tag{2.4}$$

Figure 2-8 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 is worth noting that, by executing the reverse motion, the robot is gravitated back to the center of the first attractor.

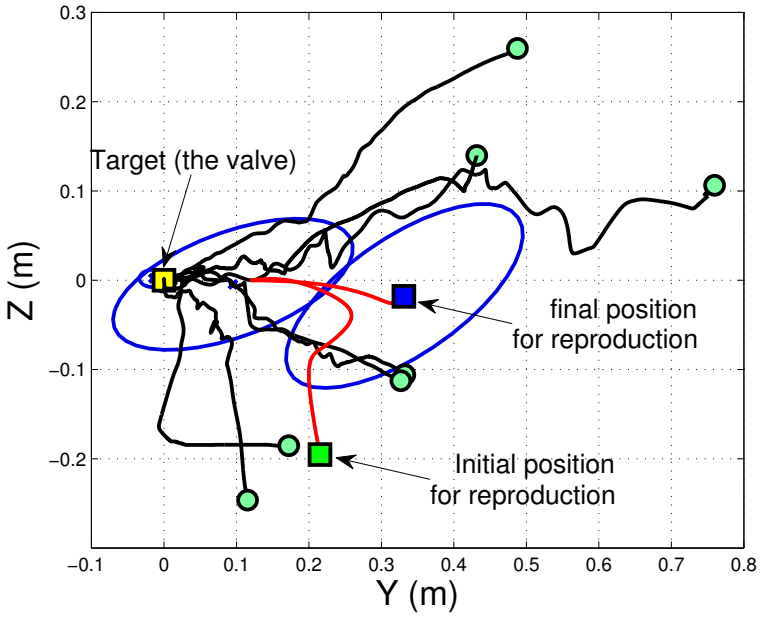


(a) Trajectories and learned attractors in 2D plane.

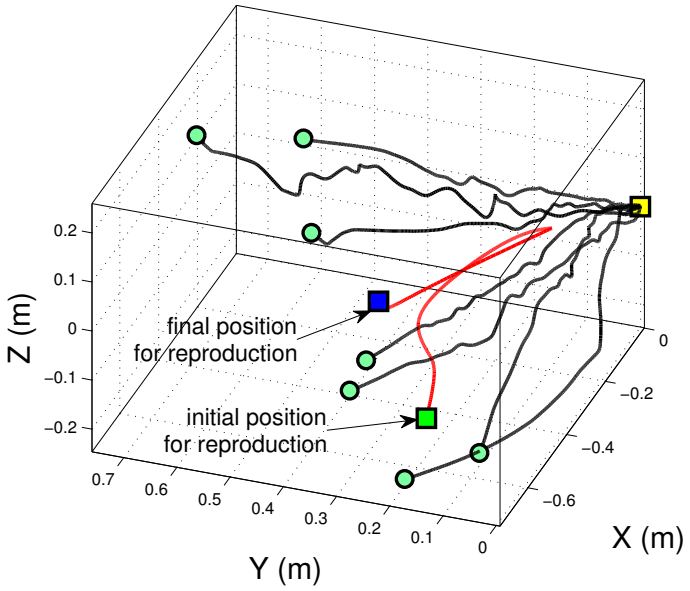


(b) Trajectories in 3D space.

Figure 2-7: The recorded trajectories that form the set of demonstrations (black), and the reproduced trajectory from an arbitrary initial position (red) towards the target are illustrated. The blue ellipses show the attractors and the yellow square shows the target (the valve). For more information, see Section 2.4.



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



(b) Trajectories in 3D space.

Figure 2-8: 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. For more information, see Section 2.4.

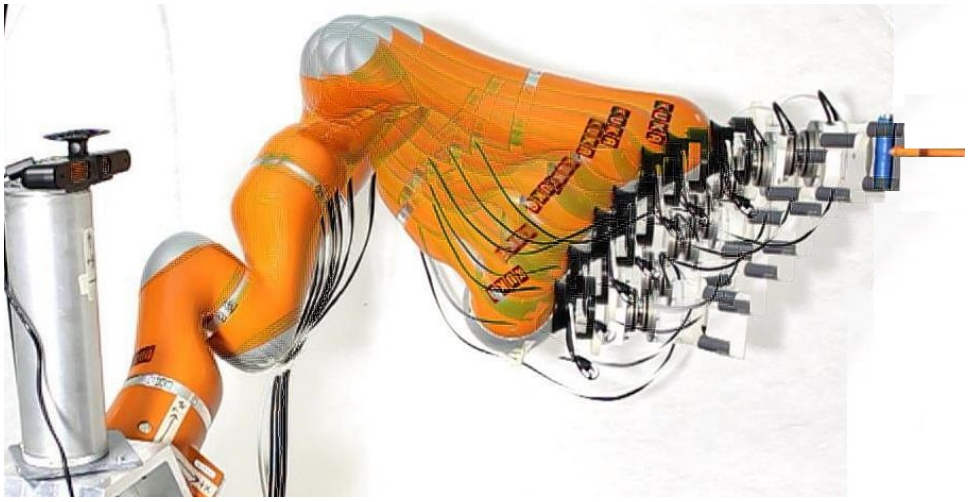


Figure 2-9: The robot reaching the valve during the reproduction phase. For more information, see Section 2.4.

The described imitation learning process is placed in the lowest layer in all of the proposed hierarchical learning architectures mentioned in the previous section.

2.5 Reactive Fuzzy Decision Maker 1

The proposed hierarchical architecture depicted in Figure 2-3 includes the imitation learning layer which was explained in the previous section. In this section, the reactive fuzzy system employed in the proposed architecture which is one of the contributions of this thesis, is described in details.

As explained in Section 2.3 and illustrated in Figure 2-3, the first variant of the Reactive Fuzzy Decision Maker, RFDM/1, is designed for experiments in the lab using a manipulator, a gripper, a mock-up valve, and a motion capture system. During the execution of the valve turning task, the RFDM module monitors the changes in the environments (i.e. uncertainties) and generates decisions that regulate the movements of the robot. 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.

Two inputs which are the main factors of generating uncertainties are considered for the fuzzy system. As shown in Figure 2-10, the first input is the relative movement between

the valve and the gripper of the robot due to disturbances and noise in the environment. The relative movement is extracted by detecting the position of the valve and calculating the relative relocation between the valve and the gripper. During the execution of the task, the system estimates the pose of the valve continuously. The pose estimation in real-world is a noise-prune and unreliable process. Also, the target may be occluded by an object. Therefore, as shown in Figure 2-10, the delay in updating the pose of the valve is considered as an input to the reactive system.

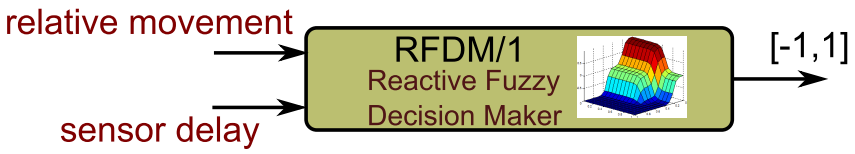


Figure 2-10: A high-level diagram illustrating RFDM/1. For more information, see Section 2.5.

The next section presents the design of the RFDM/1, and after that the tuning process of the parameters of the system are described.

2.5.1 Design of the Fuzzy System (RFDM/1)

As previously stated, the developed RFDM system takes two inputs: the estimated relative movement, and the time delay since last sensor update. The RFDM estimates the dynamical behavior of the relative movement over a frame of received data. In addition, it evaluates certainty of the current situation due to the time delay of the sensor. If the delay is *big* then the uncertainty of the current situation of the valve is *high* and vice versa. Since the duration of the movement in the reproduction phase is not fixed and the trajectory is time-independent, the movement itself changes when the RFDM reacts to the relative movement of the valve.

The task is to design a fuzzy system g that approximates a function on the compact set $[\alpha_1,\beta_1]\times[\alpha_2,\beta_2]\subset\mathbb{R}^2$ and the analytic formula of g is unknown. Suppose that for any $[u_1,u_2]\in U$, $g(u_1,u_2)$ can be obtained. So the inputs are considered as $\mathbf{u}=[u_1,u_2]^{\mathrm{T}}$. Firstly, $N_i(i=1,2)$ fuzzy sets, $A_i^1,A_i^2,...,A_i^{N_i}$, are defined in $[\alpha_i,\beta_i]=[0,1]$, which are normal, consistent, and complete with Gaussian membership functions $\mu_{A_i^1},\mu_{A_i^2},...,\mu_{A_i^{N_i}}$. Three

Gaussian fuzzy sets for each input variable are defined. Then, $N_{\text{rule}} = N_1 \times N_2$ $(3 \times 3 = 9)$ fuzzy IF$-$THEN rules are formed as follows:

$$\text{IF } u_1 \text{ is } A_1^{i_1} \text{ and } u_2 \text{ is } A_2^{i_2} \text{ THEN } y \text{ is } B^{i_1 i_2} \tag{2.5}$$

where $i_1 = 1, 2, \ldots, N_1$, $i_2 = 1, 2, \ldots, N_2$, and the center if the fuzzy set $B^{i_1 i_2}$ denoted by $\bar{y}^{i_1 i_2}$. The constructed rule base of the system is complete, continuous, and consistent, and is reported in Table 2.3. Moreover, three constant membership functions in range $[-1, 1]$ are set for the output y, where -1, 0, and 1 correspond to maximum speed retracting, waiting, and maximum speed approaching the valve, respectively. As stated previously, the advantage of using fuzzy systems is that they are based on linguistic rules and the parameters that specify membership functions have clear physical meanings and there are methods to choose good initial values for them [41].

Table 2.3: Fuzzy Rule Base

		Relative Movement		
		Small	Medium	Big
Sensor Delay	Low	Forward	Stop	Backward
	Medium	Forward	Stop	Backward
	High	Stop	Backward	Backward

The constructed TSK[4] fuzzy system [41], including product inference engine, singleton fuzzifier, and center average defuzzifier, which is used to develop the RFDM system, is formed as follows:

$$g(u_1, u_2) = \frac{\sum_{i_1=1}^{N_1} \sum_{i_2=1}^{N_2} y^{i_1 i_2} \mu_{A_1}^{i_1}(u_1) \mu_{A_2}^{i_2}(u_2)}{\sum_{i_1=1}^{N_1} \sum_{i_2=1}^{N_2} \mu_{A_1}^{i_1}(u_1) \mu_{A_2}^{i_2}(u_2)} \tag{2.6}$$

Since the fuzzy sets are complete, the fuzzy system is well-defined and its denominator is always non-zero. The fuzzy surface for input variables u_1 and u_2 is plotted in Figure 2-11.

[4]The TSK fuzzy model was proposed by Takagi, Sugeno, and Kang in an effort to develop an approach for generating fuzzy rules from a given input-output dataset [42, 43].

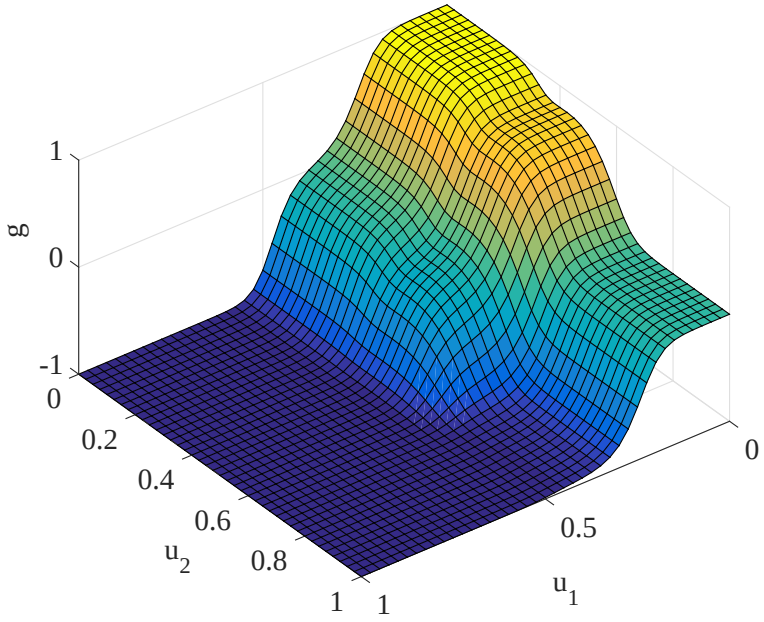


Figure 2-11: Fuzzy inference system surface for RFDM/1 including two inputs (u_1 and u_2). For more information, see Section 2.5.1.

2.5.2 Tuning the Fuzzy System (RFDM/1)

Generally, the human knowledge about a particular engineering problem can be classified in two categories: conscious knowledge and subconscious knowledge. The conscious knowledge is the knowledge that can be explicitly expressed in words or numbers. For instance, in section 2.5.1, the conscious knowledge of the expert tutor is used to design the fuzzy rule base. By the subconscious knowledge we refer to the situation where the human expert knows what to do but cannot express exactly in words how to do it. In the proposed approach, the subconscious knowledge of the expert is extracted and used for tuning the developed RFDM system, i.e. RFDM/1. This tuning process is placed in the highest layer of the proposed hierarchical structure, as depicted in Figure 2-3.

The designed RFDM/1 in Section 2.5.1, contains 6 Gaussian membership functions (A_i^1, A_i^2), 9 linguistic rules, and 3 constant outputs ($B^{i_1 i_2}$). Each Gaussian membership function includes two tunable parameters: a center (c) and a variance (σ):

$$A = \exp\left(-\left(\frac{(x-c)}{\sigma}\right)^2\right) \tag{2.7}$$

Since the linguistic rules are defined according to the desired physical behavior of the fuzzy system (conscious knowledge of the expert), the other 15 parameters including 6 centers, 6 variances, and 3 constant outputs, are considered to be tuned. After the kinesthetic teaching phase is accomplished and the first layer is capable of generating new trajectories, the tuning process can be described as following steps:

(i) a human tutor simulates the effect of the underwater currents by oscillating the valve in different Cartesian directions in specific times while the robot is following the reproduced trajectory in order to reach the valve.

(ii) Simultaneously, using a slider button, another expert tutor can supervise the manipulator by applying appropriate continuous commands to the system in range $[-1,1]$ (-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 valve is oscillating with a *big* amplitude, the tutor smoothly moves the slider backwards to retract the arm and prevent it from any collision.

(iii) All data, including the position of the end-effector and the valve, and the tutor's commands are recorded during the apprenticeship learning process.

(iv) The recorded data is used to tune the RFDM in off-line mode using optimization algorithms. The error between the recorded data from the tutor and the output of the the untuned fuzzy system is used to make the objective function.

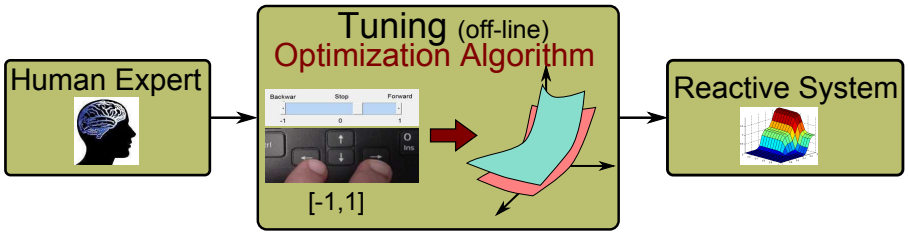


Figure 2-12: A high-level diagram illustrating the tuning layer in the proposed hierarchical approach. For more information, see Section 2.5.2.

In the rest of this section, in order to perform the tuning task, a number of local and global optimization algorithms are implemented.

Batch Gradient Descent

The Batch Gradient Descent (BGD) algorithm is a first-order local optimization algorithm. BGD is based on the observation that if a multi-variable function, $J(\theta)$, is defined and differentiable in a neighborhood of a point θ_0, then the function decreases fastest if one goes from θ_0 in the direction of the negative gradient of $J(\theta)$ at θ_0. This can be formulated as follows:

$$\theta_{n+1} = \theta_n - \alpha \nabla J(\theta_n), \tag{2.8}$$

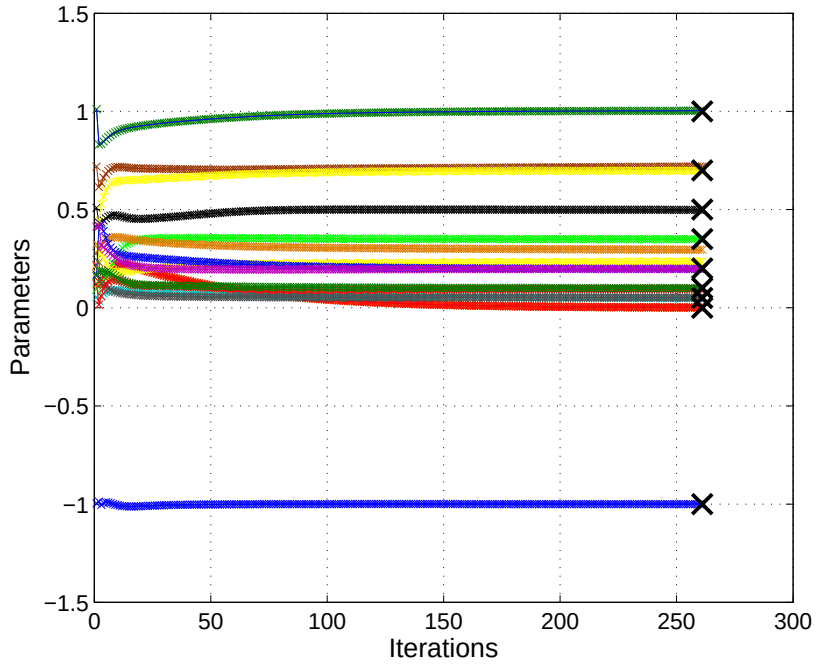
where $\alpha \in [0,1]$ is the learning rate. Consider the objective function to be:

$$J(\theta) = \frac{1}{2N} \sum_{i=1}^{N} (g_{\text{RFDM}}(\theta) - g_{\text{des}}), \tag{2.9}$$

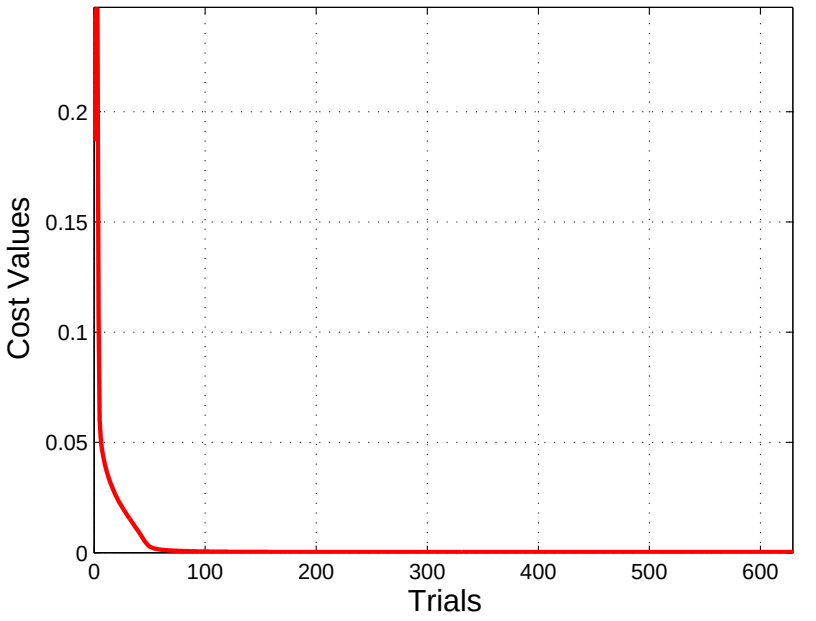
where g_{RFDM} and g_{des} are the output of the un-tuned RFDM system and the recorded subconscious knowledge of the tutor respectively. θ is the set of the parameters to be tuned including the centers, c, and the variances, σ, of the fuzzy membership functions, and the constant outputs, y. In oterh words, $\theta = \{c, \sigma, y\}$. The gradient of the defined objective function with respect to the parameters in θ is calculated in the form of following equation:

$$\frac{\partial J}{\partial \theta_i} = \frac{\partial J}{\partial g_{\text{RFDM}}} . \frac{\partial g_{\text{RFDM}}}{\partial Z_i} . \frac{\partial Z_i}{\partial \theta_i}, \quad \theta \in \{y_i, c_i, \sigma_i\} \tag{2.10}$$

where, $i = 1 \ldots 15$. Applying BGD method to the set of equations in (2.10), the algorithm minimizes the objective function. The behavior of the BGD algorithm drastically depends on the initial guess. As shown in Figure 2-13a, by choosing a proper initial guess, the algorithm converges to the optimal solution very fast (in this case 200-300 iterations). However, the algorithm may converge very slowly (10000-15000 iterations) or even diverge using other initial conditions. The cost is plotted over the trials and is illustrated in Figure. 2-13b.



(a) Convergence of the RFDM parameters.



(b) Cost function values with respect to the number of trials.

Figure 2-13: Tuning the RFDM/1 system using Batch Gradient Descent. For more information, see Section 2.5.2.

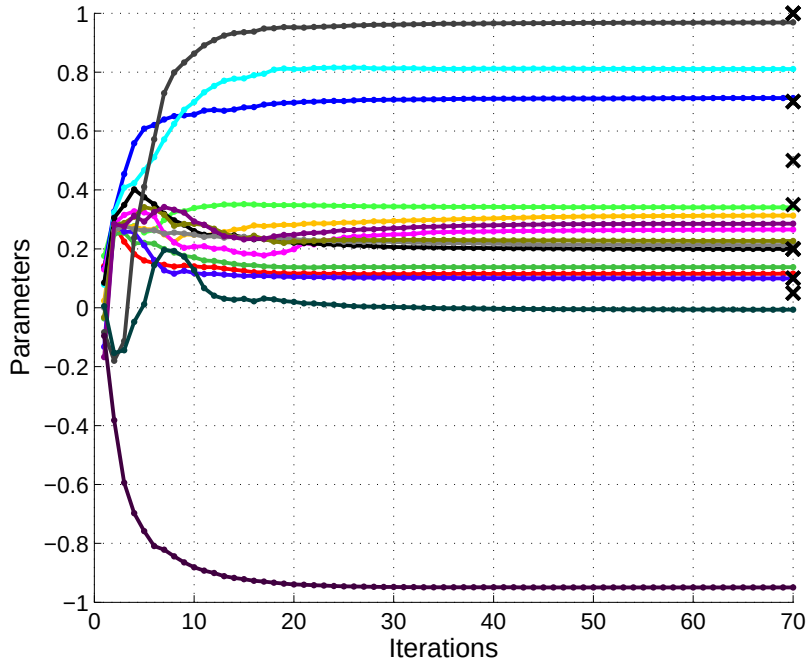
Cross Entropy Method

The Cross Entropy Method (CEM) is a generic approach to combinatorial and multi-external optimization and rare event simulation [44]. CEM involves an iterative procedure where each iteration can be broken down into two phases:

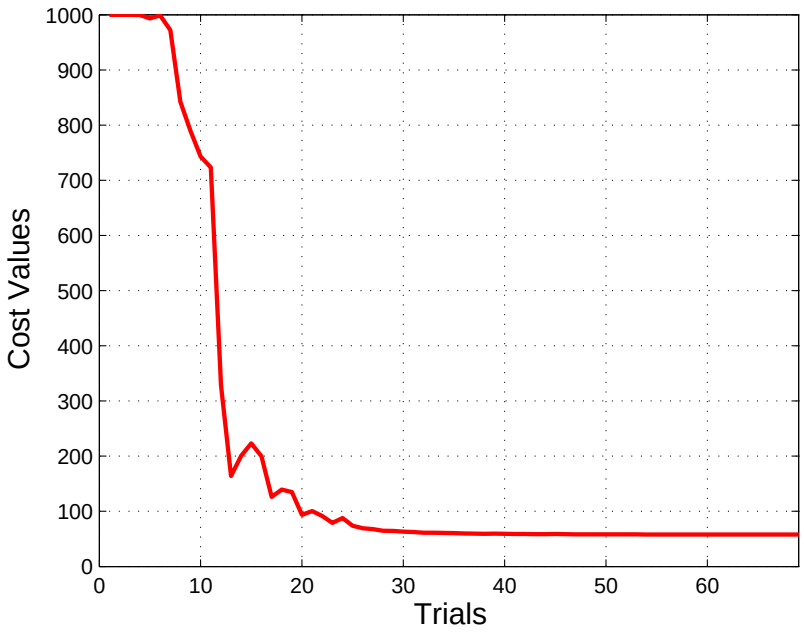
(i) Generate a random data sample according to a specified random mechanism;

(ii) Update the parameters of the random mechanism based on the data to produce a 'better' sample in the next iteration.

The significance of CEM is that it defines a precise mathematical framework for deriving fast and in some sense 'optimal' updating/learning rules, based on advanced simulation theory [45]. In this tuning process, a variation of initial sample size between 50 and 400 for the first sample generation is used, whereas the next generation samples are provided using the best 10% of the previous samples. The objective function in (2.9) is utilized. Depending on the initial mean and variance values of the samples, the algorithm converges to a local optimal solution in 50 - 500 iterations (see Figure 2-14a). The cost is plotted over the trials and is illustrated in Figure 2-14b. For more information about CEM, see Appendix C.



(a) Convergence of the RFDM parameters.

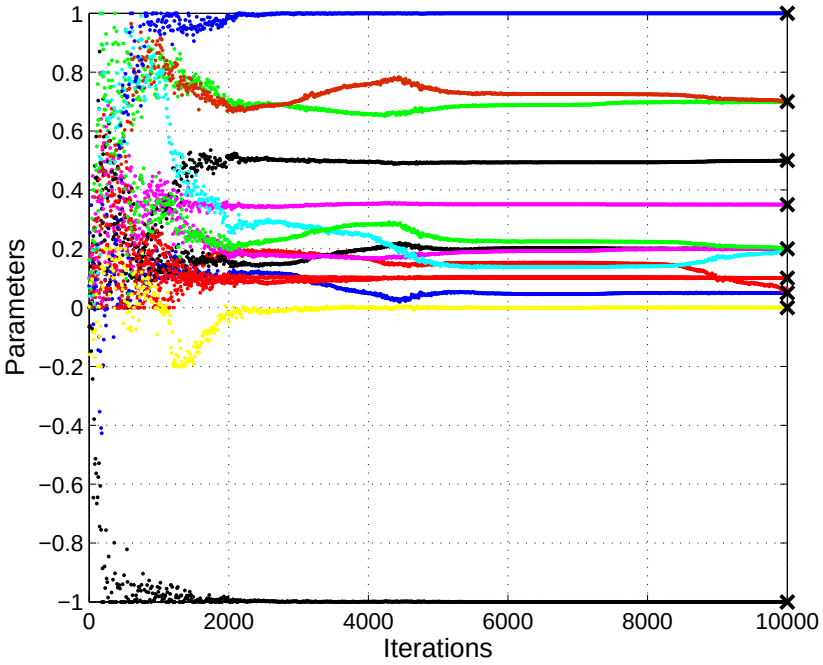


(b) Cost function values with respect to the number of trials.

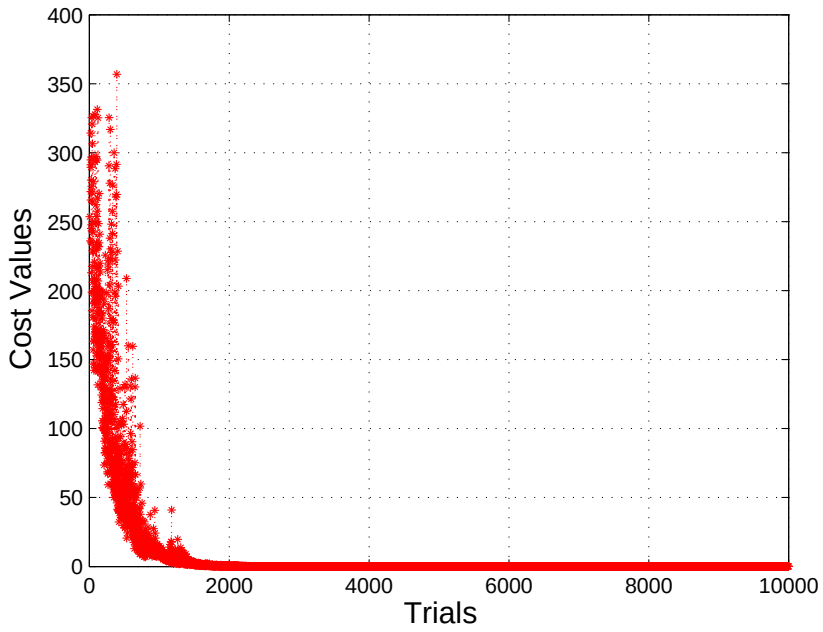
Figure 2-14: Tuning the RFDM/1 system using Cross Entropy Method. For more information, see Section 2.5.2.

Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

The CMA-ES is a stochastic, derivative-free method for real-parameter (continuous domain) optimization of non-linear, non-convex optimization problems [46]. In each generation (iteration) new individuals (candidate solutions) are generated by sampling a multi-variable normal distribution. In the next step some individuals are selected for the next generation based on their fitness or objective function value. Pairwise dependencies between the variables in this distribution are represented by a covariance matrix. The CMA is a method to update the covariance matrix of this distribution. Over the generation sequence, individuals with better fitness are generated. This optimization algorithm is particularly useful if the objective function is ill-conditioned. In contrast to most other evolutionary algorithms, the CMA-ES is quasi parameter-free. However, the number of candidate samples (population size) can be adjusted by the user in order to change the characteristic search behavior [46]. For more information about CMA-ES, see Appendix B. For the optimization task using the objective function represented as (2.9) with 15 parameters to tune, the algorithm converges around 10000 iterations (see Figure 2-15a). However, increasing the population size (e.g., by 3 times) decreases the number of iterations (to 3000-4000). Among the implemented algorithms, the CMA-ES is the fastest in terms of computation time, and requires a smaller number of initial parameter settings. The cost is plotted over the trials and is illustrated in Figure 2-15b.



(a) Convergence of the RFDM parameters.

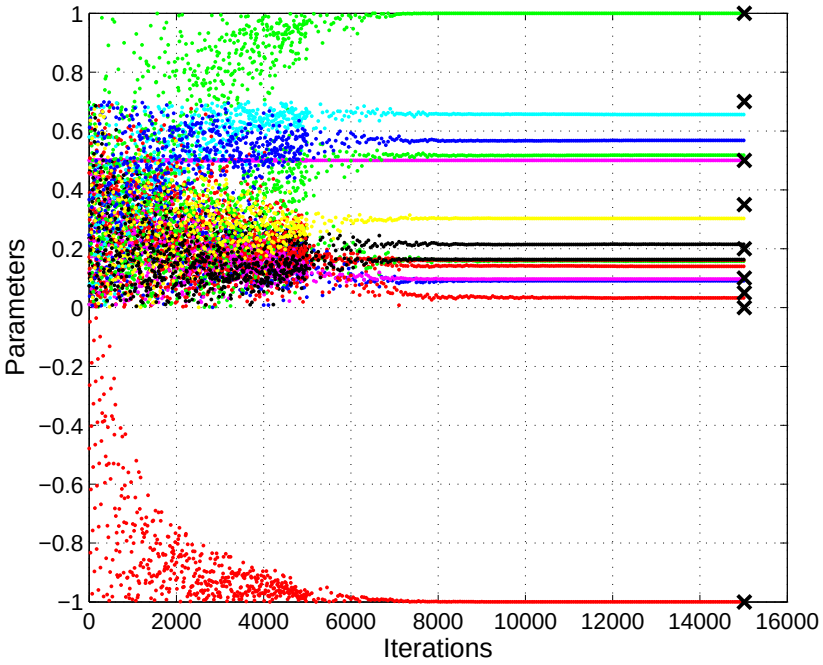


(b) Cost function values with respect to the number of trials.

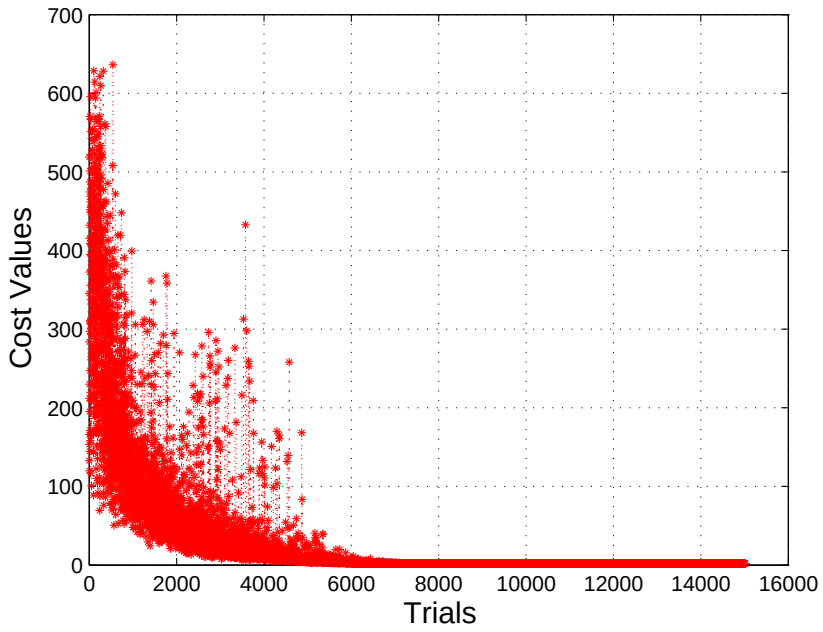
Figure 2-15: Tuning the RFDM/1 system using CMA-ES Method. For more information, see Section 2.5.2.

Modified Price's Algorithm

This algorithm is designed for particularly difficult global optimization problems in which the evaluation of the object function is very expensive, and the derivatives of the objective function are not available. To solve this problem, a modified Price's algorithm by Brachetti *et al.,* [47] is implemented. The modified Price's algorithm is a population-based algorithm with global and local search parts. The global search part consists of the weighted centroid and the weighted reflection. The Number-theoretic method is applied to generate the initial population and a simplified quadratic approximation using the three best points is adopted, instead of the quadratic model of the objective function in original Price's algorithm. There is just one initial parameter for this algorithm to be set and that is the size of population. See Appendix E for more information about the Modified Price's algorithm. As depicted in Figure 2-16a, the algorithm converges around 15000 iterations with a population size of 300. The algorithm converges very slowly even if the population size is increased to 1500. The cost is plotted over the trials and is illustrated in Figure 2-16b.



(a) Convergence of the RFDM parameters.



(b) Cost function values with respect to the number of trials.

Figure 2-16: Tuning the RFDM/1 system using Modified Price's Method. For more information, see Section 2.5.2.

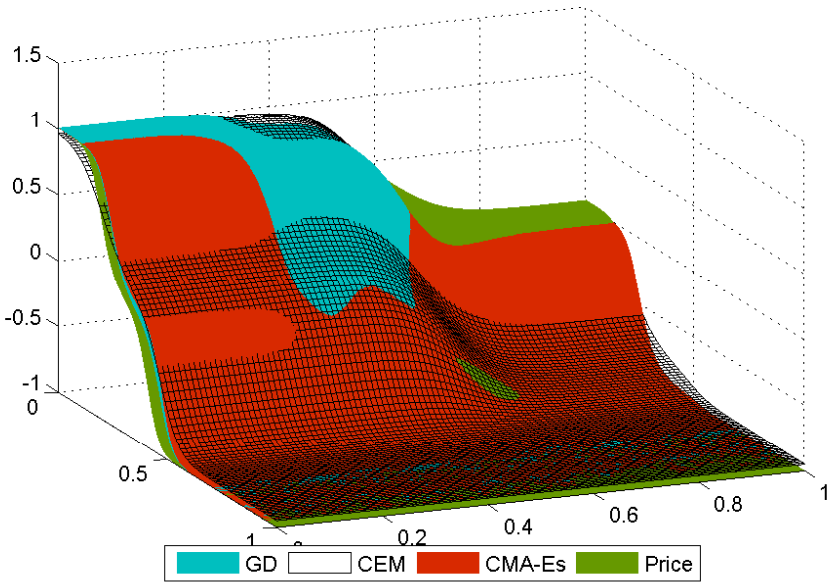
Results and Comparison

Assigning obtained values from each optimization algorithm to the designed RFDM/1 structure, 4 slightly different RFDM structures are acquired. As an illustration, the output surface of each attained system is shown in Figure 2-17a. In addition, the tuned Gaussian membership functions by each algorithm are depicted in Figure 2-17a.

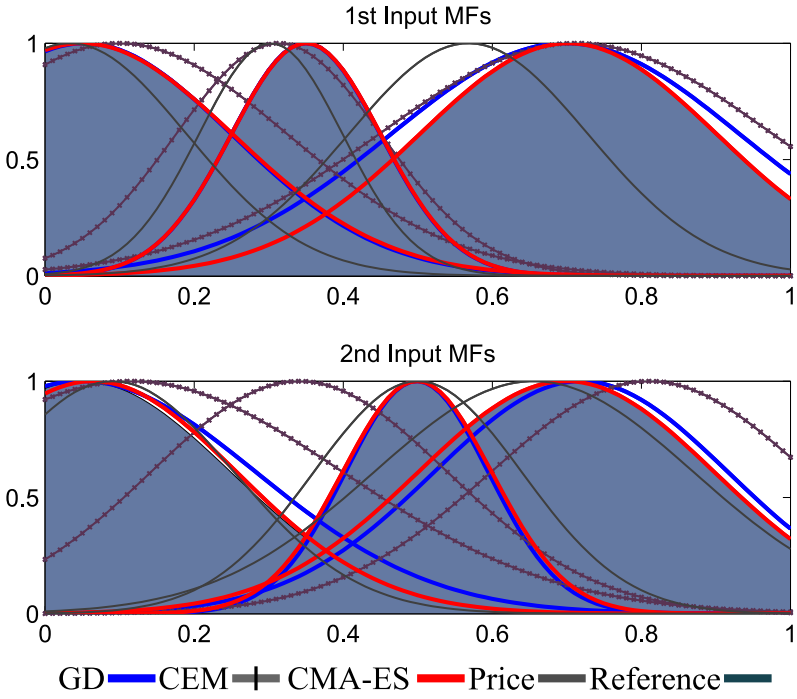
Finally, Root-Mean-Square Error (RMSE) between each two solutions is calculated and reported in Table 2.4. One can see that, the closest solutions to the reference subconscious knowledge of the tutor are obtained from the CMA-ES and the BGD methods.

Table 2.4: RMSE Table

	BGD	CEM	CMA-ES	Price
Reference	0.0263	0.1015	0.0044	0.1665
BGD	—	0.1067	0.0285	0.1558
CEM	—	—	0.0994	0.1955
CMA-ES	—	—	—	0.1699



(a) Fuzzy surfaces



(b) Fuzzy membership functions

Figure 2-17: Final fuzzy surfaces are depicted in 2-17a. The tuned membership functions are illustrated in 2-17b. For more information, see Section 2.5.2.

2.6 Experiments with RFDM/1

In this section, the proposed hierarchical learning approach including the skill learning approach presented in Section 2.4 and the reactive system designed and tuned in Section 2.5, is applied to the task of valve turning in a laboratory environment[5].

2.6.1 Experimental Setup

To accomplish the valve turning task using the proposed hierarchical learning approach, an integrated system has been built in the lab including an Optitrack system and a lightweight KUKA-DLR robotic arm. The Optitrack system captures real-time 3D position and orientation data of a rigid body using a number of motion capture cameras and a set of markers. An important feature of the Optitrack system is that, it provides precise and high frequency data, whereas the real AUV is equipped with different types of sensors, e.g., stereo camera and gyro-enhanced AHRS[6]. Such feature allows the tutor to add specific amount of noise to the system manually which results in improving the capability of analysing the system. Furthermore, in order to use kinesthetic teaching technique, the KUKA-DLR robotic arm is used under Cartesian impedance control mode [48]. The robot's position, orientation and joint or Cartesian stiffness commands are sent to KUKA controller using the DLR's Fast Research Interface libraries [49]. Although the KUKA-DLR is a different kind of manipulator than the one is attached to the underwater vehicle, the proposed trajectory generation learning method is independent of the kinematics of the manipulator.

Another assumption considered in this experiments is that, the base of the robot maintains fixed while the tutor can hold and relocate the valve in the workspace of the manipulator. This assumption helps the tutor to apply supervised amount of oscillation to the valve in order to generate relative movement between the valve and the gripper. And finally, in this set of experiments, to accomplish the turning phase a torque control strategy is pre-programmed to the robot. However, in Section 2.7, a force/motion control

[5] A video of the real-world experiments reported in this section is available online at my personal website `http://www.ahmadzadeh.info/videos`.

[6] Attitude and Heading Reference System

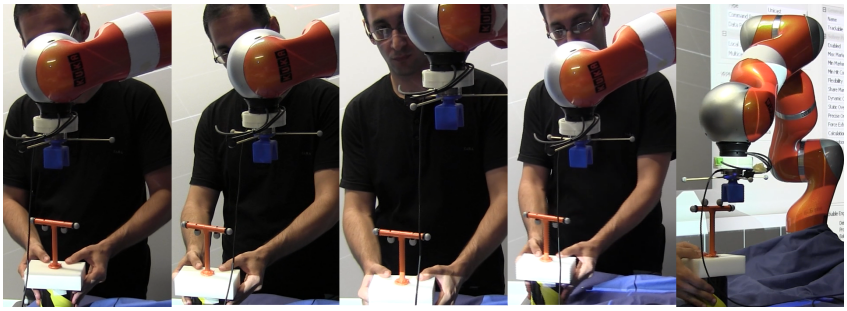
strategy is employed to accomplish the turning phase.

2.6.2 Experimental Results

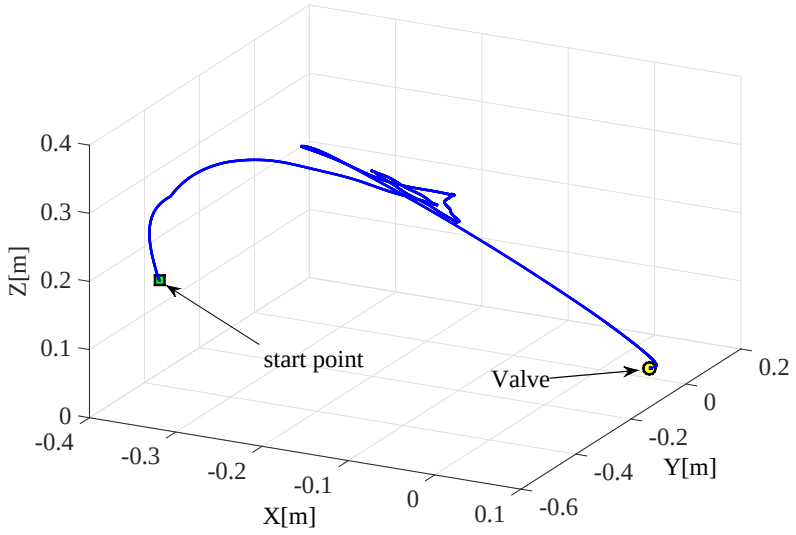
The complete autonomous robotic valve turning experiment using the proposed hierarchical learning approach is summarized as following steps:

- Recording a set of demonstrations from various arbitrary initial positions. The recorded trajectories are shown in Figure 2-7.
- Applying the Extended DMP method to learn the new motor skill. The method is explained in Section 2.4.
- Recording the expert knowledge, while robot is in the reproduction mode. The method is explained in Section 2.5.1.
- Tuning the RFDM/1 system using the recorded data from previous step. The method is described in Section 2.5.2.
- Accomplishing reproduction mode and grasping the valve using the tuned RFDM/1.
- Applying a torque control on end-effector an turn the valve (This part is pre-programmed to the robot).

The set of images in Figure 2-18a represents five steps of the experiment, in which the robot is following the reproduced trajectory to reach the target. The recorded position of the end-effector in a complete experiment is shown in Figure 2-18b.



(a) Set of images from different steps of the experiment.



(b) The recorded positions of the end-effector supervised by tuned RFDM/1

Figure 2-18: A set of snapshots in 2-18a shows the execution of the task. The reproduction of the task is illustrated by the trajectory depicted in 2-18b. The effect of the reactive system can be seen in the middle of the trajectory while the valve is oscillating. For more information, see Section 2.6.

2.7 Force/Motion Control Strategy

One of the shortcomings of the proposed system in the previous section is that the turning phase is pre-programmed into the system. During the grasping and turning phase, the robot may exert huge forces/torques on the valve which is another hazardous and highly probable situation. In such case, 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. In order to increase the autonomy of the system, the turning module should be improved. To achieve this goal, a hybrid force/motion control strategy is designed to handle the turning phase. 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. Consequently, the reactive system entitled 'RFDM/2', which is designed and used in Section 2.8, additionally observes the amount of force/torque applied to the valve.

Once the robot reaches the target, 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 [50–52]. 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 Figure 2-19 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 [53]. 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.

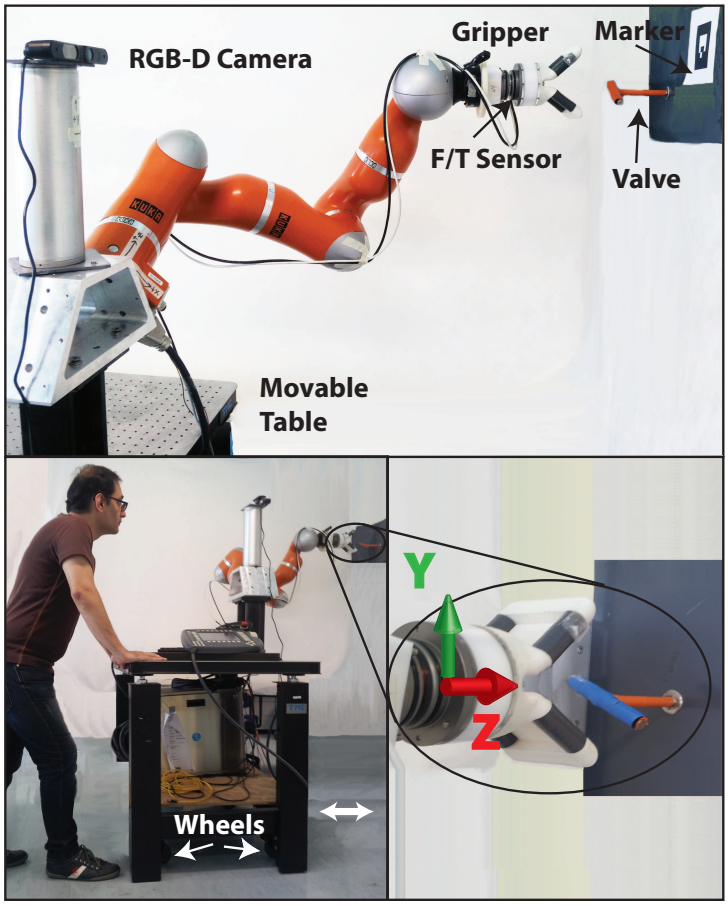


Figure 2-19: The experimental setup for the valve turning task in the second set of experiments. The valve is detected and localized using an RGB-D sensor through an AR-marker. The manipulator is equipped with a gripper and is mounted on a movable (wheeled) table. During the execution of the task, a human tutor can create a random disturbance by perturbing the base of the robot. For more information, see Section 2.7.

The hybrid force/motion controller is suitable for autonomous mobile manipulation [54]. In addition, 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. In the proposed control strategy the forces and torques are specified as follows:

$$\mathbf{f}_{\text{con}} = \mathbf{f}_{\text{des}} + \mathbf{k}_p(\mathbf{f}_{\text{des}} - \mathbf{f}_{\text{act}}) \tag{2.11}$$

$$\tau_{\text{con}} = \tau_{\text{des}} + \mathbf{k}_p(\tau_{\text{des}} - \tau_{\text{act}}), \tag{2.12}$$

where $\mathbf{f}$ and τ denote force vector and torque vector respectively, and subscripts des,

act, and con denote the desired, actual, and control parameters respectively. In the following sections, in order to evaluate the efficiency of the designed control strategy, some experimental results are explained. In Section 2.7.1, the valve turning experiment is performed with stationary base in which there is no manual disturbances. In Section 2.7.2, on the other hand, the tutor applies manual disturbances to the system in order to test the behavior of the hybrid control strategy in presence of noise.

The designed hybrid force/motion control strategy is replaced with the pre-programmed turning module in Figure 2-3 that consequently results in a new architecture as depicted in Figure 2-4. The corresponding reactive system, RFDM/2 is described in the Section 2.8.

2.7.1 Valve Turning with Stationary Base

In the first set of experiments the base of the robot remained stationary during the valve turning task. So, no external disturbances are applied to the valve or the gripper. In this case, to ensure proper contact between the valve and the gripper during the turning phase a 20 N force along the z-axis is applied, $f^z_{\text{des}} = 20$ N. The other desired forces and torques are set to zero as follows:

$$f^x_{\text{des}} = 0 \tag{2.13}$$

$$f^y_{\text{des}} = 0 \tag{2.14}$$

$$f^x_{\text{des}} = 0 \tag{2.15}$$

$$f^y_{\text{des}} = 0. \tag{2.16}$$

The desired roll angle is specified to be 180° clockwise with respect to the current orientation of the end-effector. The duration of turning is set to 10 s. The joint angle displacements through the valve turning execution are shown in Figure 2-20, where the last joint moved from −90° to 90° angle, while the rest of the joint angles move slightly in order to dissipate the forces generated at the end-effector during the execution of the task.

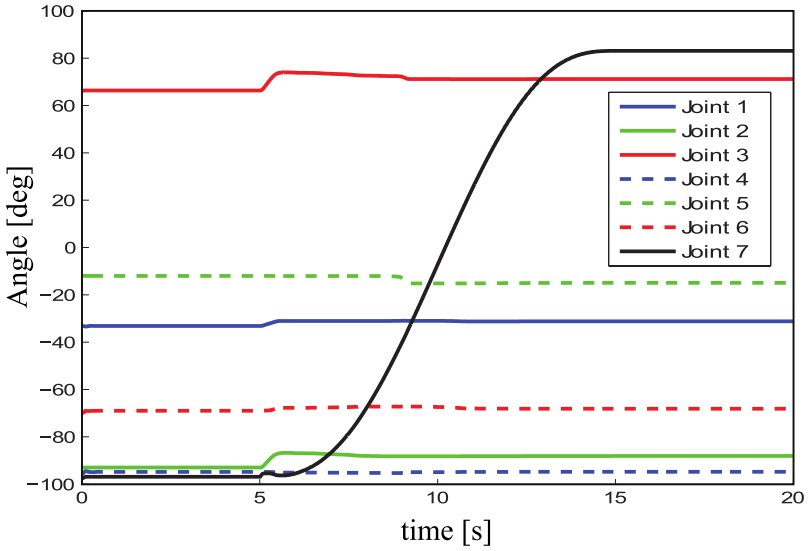


Figure 2-20: Joint angles during the valve turning task with stationary base. For more information, see Section 2.7.1.

It can be seen in the graphs that the first 5 s of the operation is used for the gripper to make contact with the valve and then dissipate the impact force. From 5 s to 15 s, the valve is turned by the motion controller and the generated forces and torques at the end-effector are recorded. Figure 2-21 (top figure) shows that the force along the z-axis is maintained at +20 N right after the impact forces have been dissipated. The controller can maintain the magnitude of the force before, during, and after the valve turning successfully.

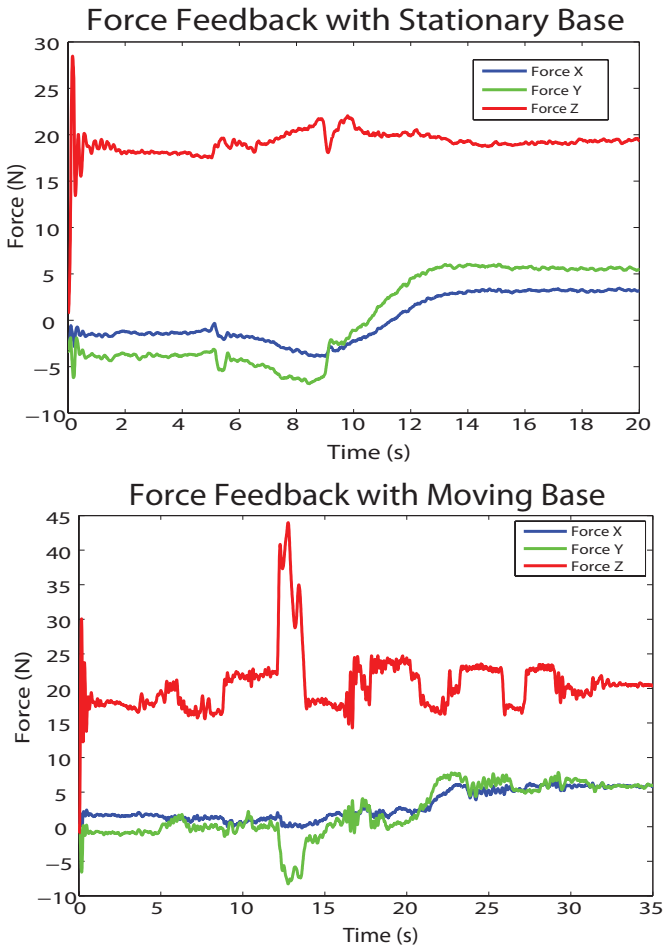


Figure 2-21: End-effector forces during the valve turning task. For more information, see Section 2.7.1.

The exertion of the necessary torque to turn the valve and the required force to maintain the contact between the valve and the gripper, generates residual forces and torques at the end-effector. In order to dissipate these reaction forces along the x- and y-axes, a set of forces in range -5 to $+5$ N are generated. During the turning phase until the 20 s of the task, the z-axis torque averaged at around $+1$ Nm. This is shown in Figure 2-22 (top figure). On the other hand, the torque around x-axis is maintained at -0.5 Nm at the start of the contact until the end of the turning phase, and then decreased to 0 Nm by the end of the task. The torque around the y-axis moved from its initial torque, (after impact that is around -0.5 Nm) to -1 Nm in an attempt to dissipate the residual torques. However, due to the physical limitation of the valve, (i.e., the gripper has a slot along the y-axis), it cannot create the necessary torque-resistance to dissipate the residual torque.

2.7.2 Valve Turning with Moving Base

In this experiments, the base of the manipulator, that is mounted on a wheeled table, is moved manually to create a disturbance during the turning task. As stated before, these oscillatory disturbances simulate the dynamics of currents in the underwater environment. The duration of the valve turning process is set to 30 s to give enough time to the operator to disturb the manipulator. The recorded force and torque data during the valve turning with perturbed base are shown in bottom sub-figures in Figures 2-21 and 2-22. The perturbation is generated from 3 s up to 33 s of the overall time.

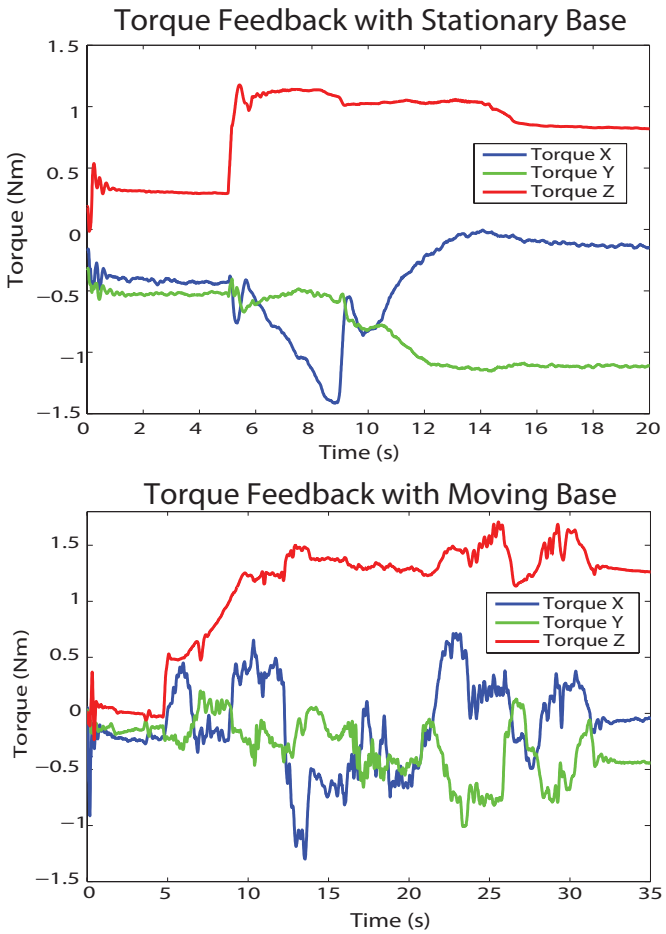


Figure 2-22: End-effector torques during the valve turning task. For more information, see Section 2.7.2.

2.8 Reactive Fuzzy Decision Maker 2

Replacing the designed hybrid force/motion control strategy with the pre-programmed turning module, the robot is able to perform the turning phase autonomously. 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 uncontrolled 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, the reactive system designed in this section, RFDM/2, utilizes the data acquired from the F/T sensor.

One of the drawbacks of the RFDM/1 system is that, it is independent of the distance between the gripper and the valve. This means that despite the distance between the gripper and the valve, the robot shows identical behaviors. In addition to the relative movement between the valve and the gripper, the proposed RFDM in this section comprises two more inputs. The first 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 watchful decisions and increases the sensitivity of the robot's movements with respect to the disturbances. The second 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. Figure 2-23 illustrates the input/output components of the RFDM/2. In the next section, the design process of the system is described. The tuning process is described in Section 2.8.2.

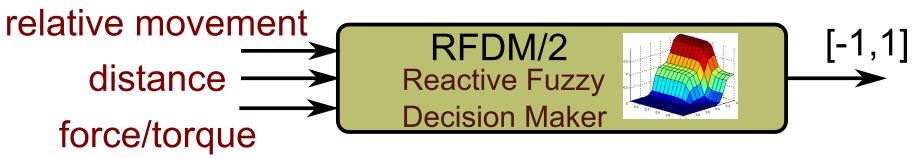


Figure 2-23: A high-level diagram illustrating RFDM/2. For more information, see Section 2.8.

2.8.1 Design of the Fuzzy System (RFDM/2)

As shown in Figure 2-23, the proposed fuzzy system comprises three inputs: (i) the distance between the gripper and the valve (the norm of the distance vector); (ii) the

relative movement between the valve and the gripper in $x-y$ plane; and (iii) the forces and torques applied to the valve from the gripper.

All the inputs are first re-scaled in range $[0,1]$ and then sent to the RFDM/2 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:

$$\gamma = \frac{\|\mathbf{f}\| + \beta \|\boldsymbol{\tau}\|}{f_{\max}} \tag{2.17}$$

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.

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 sideways. 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.

Similar to RFDM/1, the output of the RFDM/2 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.

In order to design the fuzzy system, the inputs is considered to be $\mathbf{u} = [u_1, u_2, u_3]^{\mathrm{T}}$ and the output is considered as $g(\mathbf{u})$. 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 $\mu_{A_i^1}, \mu_{A_i^2}, ..., \mu_{A_i^{N_i}}$. The defined membership functions for each input are depicted in Figure 2-24. Then, $N_{\text{rule}} = N_1 \times N_2 \times N_3$ $(3 \times 4 \times 3 = 36)$ fuzzy IF$-$THEN rules are formed as follows:

$$\text{IF } u_1 \text{ is } A_1^{i_1} \text{ and } u_2 \text{ is } A_2^{i_2} \text{ and } u_3 \text{ is } A_3^{i_3} \text{ THEN } y \text{ is } B^{i_1 i_2 i_3} \tag{2.18}$$

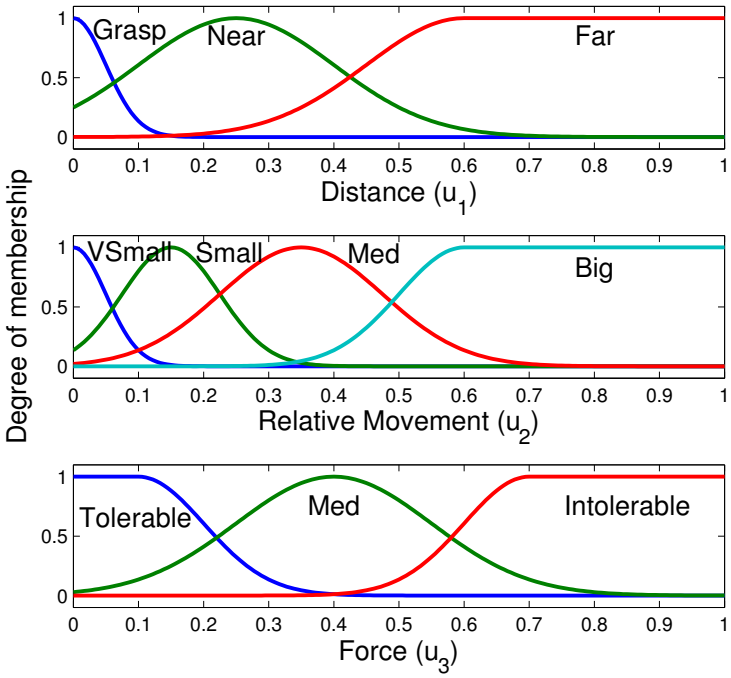
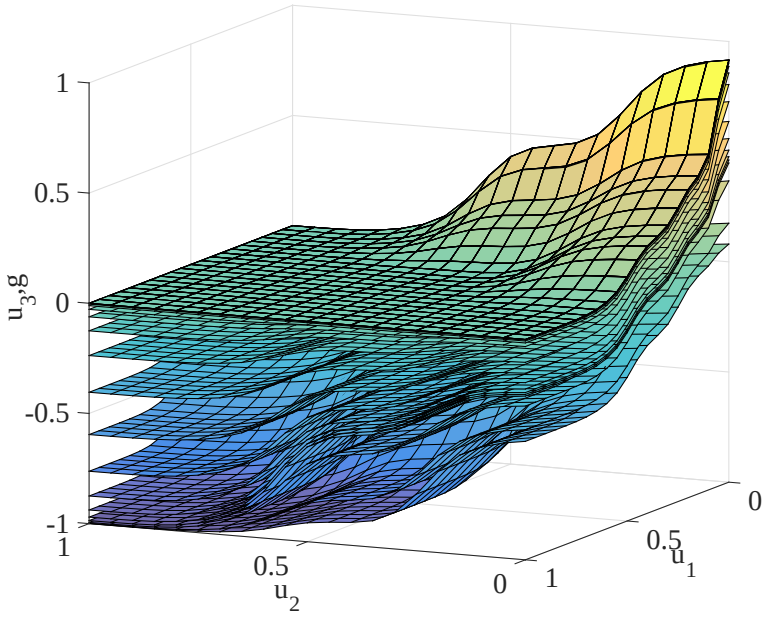


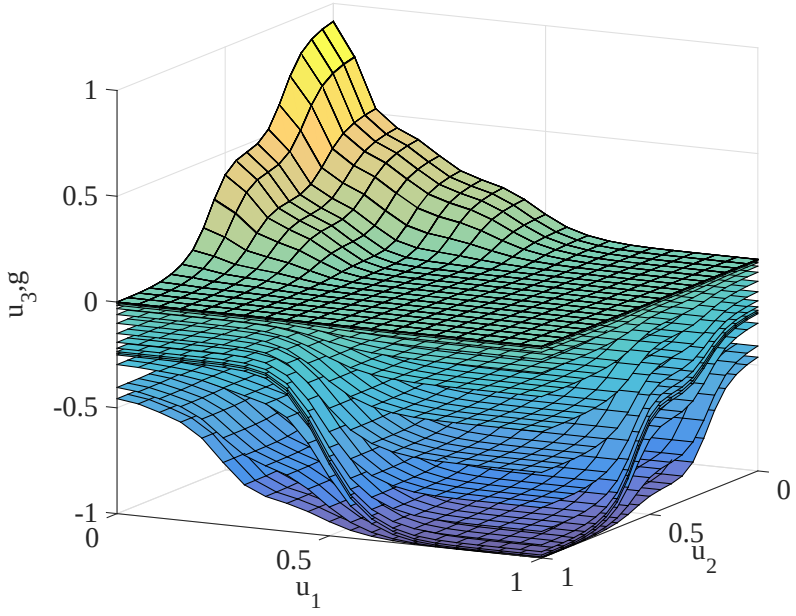
Figure 2-24: Fuzzy membership functions defined for each input in RFDM/2. For more information, see Section 2.8.1.

Moreover, 7 constant membership function in range $[-1,1]$ are set for the output. Finally, the TSK fuzzy system is constructed using product inference engine, singleton fuzzifier, and center average defuzzifier [41]:

$$g(u_1, u_2, u_3) = \frac{\sum_{i_1=1}^{N_1} \sum_{i_2=1}^{N_2} \sum_{i_3=1}^{N_3} y^{i_1 i_2 i_3} \mu_{A_1}^{i_1}(u_1) \mu_{A_2}^{i_2}(u_2) \mu_{A_3}^{i_3}(u_3)}{\sum_{i_1=1}^{N_1} \sum_{i_2=1}^{N_2} \sum_{i_3=1}^{N_3} \mu_{A_1}^{i_1}(u_1) \mu_{A_2}^{i_2}(u_2) \mu_{A_3}^{i_3}(u_3)} \tag{2.19}$$



(a)



(b)

Figure 2-25: Fuzzy inference system surface for RFDM/2 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. (For more information, see Section 2.8.1).

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. Thus, the fuzzy surface for input variables u_2 and u_3 is plotted over a single value of the variable u_1. Consequently, each surface in Figure 2-25 is related to a fixed value of u_1. It can be seen from Figure 2-25 that the behavior of the RFDM/2 system varies adaptively with respect to the distance to the target and the system shows more sensitive and cautious behaviors as the distance to the valve decreases.

2.8.2 Tuning the Fuzzy System (RFDM/2)

In order to tune the parameters of RFDM/2 using the subconscious knowledge of 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.

Similar to the performed process in Section 2.5.2, 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, are used to make an objective function. The objective function can be minimized using various optimization algorithms. In Section 2.5.2, four different optimization algorithms including batch gradient-descent, cross entropy method [44], covariance matrix adaptation-evolution strategy (CMA-ES) [46], and modified Price algorithm [47] were applied. Considering the results reported in Section 2.5.2, CMA-ES

seems to be a suitable choice for the tuning task, also CMA-ES is typically applied to search space dimensions between three and a hundred [46]. The number of optimization parameters for this problem is equal to the number of membership functions multiplied by two (one for the center, c, and the other for the standard deviation, σ), plus the number of constant outputs. In RFDM/2 the number of optimization parameters is equal to 35 (14 Gaussian Membership Functions $\times$ 2 parameters for each Gaussian + 7 constant outputs).

Furthermore, the implementation of the CMA-ES algorithm contains a restart option that allows to repeat the optimization process with different settings. So the algorithm begins with the default small population size. The restart option is used for an automated multistart where the population size is increased by a specific factor (two by default) before each restart. Figure 2-26 illustrates the results obtained from applying the CMA-ES algorithm with five restarts in order to tune the parameters of RFDM/2. For more information about the parameters of the CMA-ES algorithm, see Appendix C.

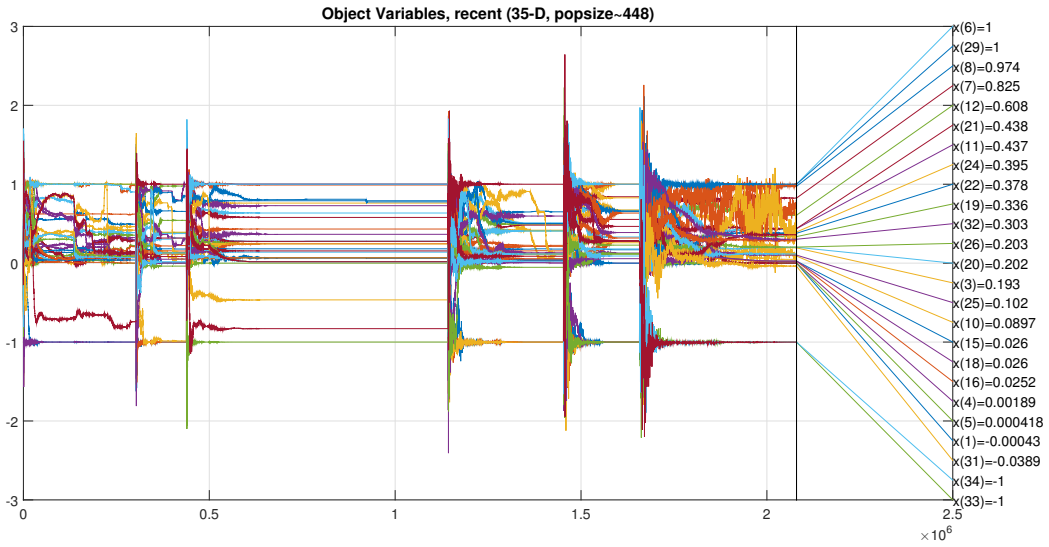


Figure 2-26: The results obtained from applying the CMA-ES algorithm with five restarts. The plot illustrates the time evolution of the distribution mean (default) or the recent best solution vector. For more information, see Section 2.8.2.

2.9 Experiments with RFDM/2

In this section, the proposed hierarchical learning approach illustrated in Figure 2-23, including the imitation learning layer, the hybrid force/torque control strategy, and the tuned RFDM/2 is applied to the task of valve turning in a laboratory environment.

2.9.1 Experimental Setup

As depicted in Figure 2-19, 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 robot's end-effector, and an ASUS Xtion Pro Live RGB-D sensor for detecting and localizing the valve.

Figure 2-4 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.

2.9.2 Experimental Results

In this section, the behavior of the RFDM/2 system is investigated during a real-world valve turning experiment. All three inputs of the reactive system are illustrated in Figure 2-27. The first input, the distance, shows that initially the gripper was located far from the valve and gradually approached towards it. The second input, the relative movement, shows that, at some point a relative movement between the gripper and the valve is occurred. The relative movement was created manually by moving the robot's base. And the third input, the force, shows small values during the process but at the end a sudden jump in the force was occurred. The jump in the force magnitude was generated manually

by pushing the manipulator towards the valve during the turning phase. The generated decision commands by the RFDM/2 system is plotted in Figure 2-27. The effect of both the manual oscillation of the base and the manual push on the gripper is observed by RFDM/2 and proper decisions are generated. During the manual oscillation of the base, the RFDM system decreases the rate of the motion towards zero. However, it does not retract the arm because at this point the gripper has not reached the valve yet. Also by sensing the force created by the sudden push, the reactive system retracts the gripper from the valve.

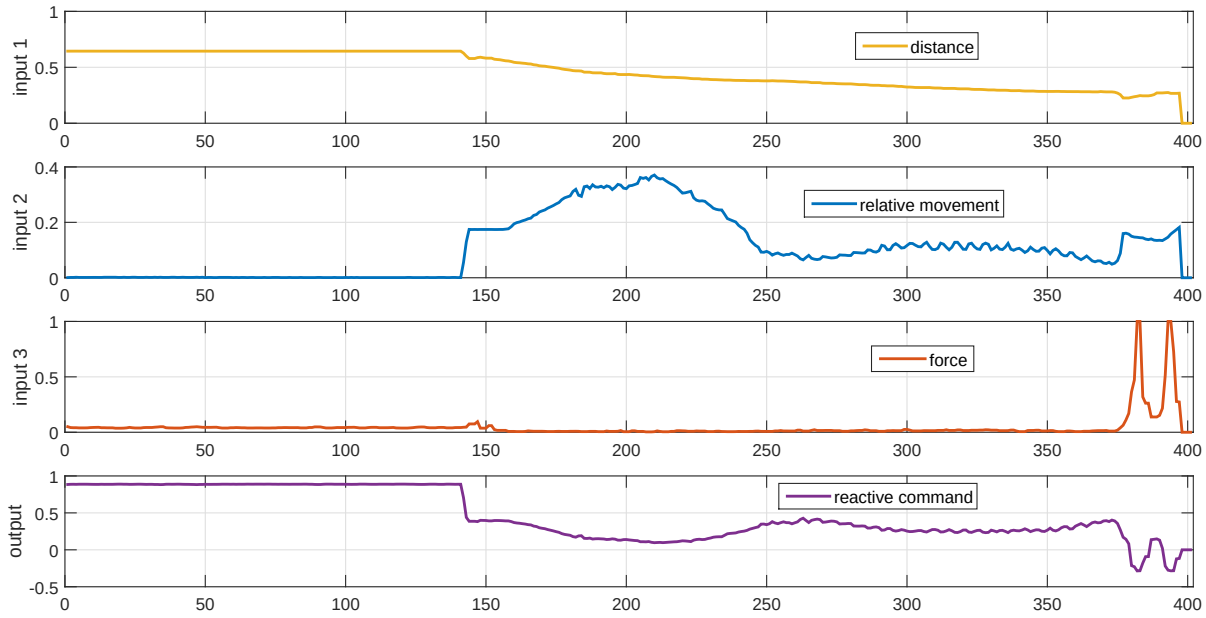


Figure 2-27: The recorded set of inputs and the generated decision commands by the RFDM/2 system during a real-world valve turning experiment. For more information, see Section 2.9.

2.10 Reactive Fuzzy Decision Maker 3

The reactive fuzzy systems designed in the previous sections are validated through several experiments in the laboratory environment. In this section, a variant of the reactive system is designed in order to be used in underwater environment. The RFDM/2 system seems a good choice for this purpose, however, in the setup for underwater experiment the robot which is equipped with a manipulator and a gripper, lacks an F/T sensor. Consequently, the force/torque input is excluded from the proposed architecture in this section. Instead,

the sensor delay is included which is a very important factor that can cause uncertainty due to occlusion, poor visibility, etc. Thus, as depicted in Figure 2-28, the proposed reactive system consists of three inputs, namely, the relative movement, the sensor delay, and the distance between the gripper and the valve.

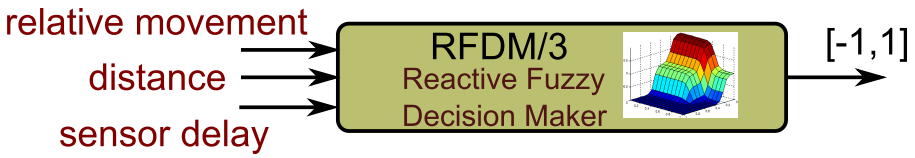


Figure 2-28: A high-level diagram illustrating RFDM/3. For more information, see Section 2.10.

2.10.1 Design of the Fuzzy System (RFDM/3)

The design process is similar to that of RFDM/2 described in Section 2.8.1. As shown in Figure 2-28, the proposed fuzzy system comprises three inputs: (i) the distance between the gripper and the valve (the norm of the distance vector); and (ii) the relative movement between the valve and the gripper (in $x-y$ plane); (iii) the delay in updating the pose from the vision sensor. All the inputs are first re-scaled in range $[0,1]$ and then sent to the RFDM/3 system.

Monitoring the relative movement between the valve and the gripper, the system can detect oscillations with different amplitudes and frequencies. To simulate such behavior in the underwater experiments, the operator applies random lateral disturbances (i.e. oscillations) to the AUV using joystick while the system is autonomously executing the valve turning process. Alternatively, the tutor can oscillate the panel including the valve during the execution of the task. Both methods are utilized in the experiments. Moreover, similar to RFDM/2, 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, by measuring the delay in updating the pose of the valve, the system reacts to the produced uncertainties accordingly.

Similar to the previous reactive systems the output of the RFDM/3 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.

The designed fuzzy system includes three inputs, $\mathbf{u} = [u_1, u_2, u_3]^{\mathrm{T}}$ and one output $g(\mathbf{u})$. $N_i (i = 1,2,3)$ fuzzy sets, $A_i^1, A_i^2, ..., A_i^{N_i}$, are defined in range $[0,1]$, with Gaussian membership functions $\mu_{A_i^1}, \mu_{A_i^2}, ..., \mu_{A_i^{N_i}}$. Then, $N_{\text{rule}} = N_1 \times N_2 \times N_3$ fuzzy IF$-$THEN rules are formed as in (2.18). Also, 7 constant membership function in range $[-1,1]$ are set for the output. Finally, the TSK fuzzy system is constructed using product inference engine, singleton fuzzifier, and center average defuzzifier as in (2.19).

2.10.2 Tuning the Fuzzy System (RFDM/3)

Similar to Section 2.8.2, the error between the recorded data obtained 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. In RFDM/3 the number of optimization parameters is equal to 35. Considering the results reported in Section 2.5.2, CMA-ES seems to be a suitable choice for the tuning task, also CMA-ES is typically applied to search space dimensions between three and a hundred [46].

2.11 Underwater Experiments with RFDM/3

In this section, the proposed hierarchical learning approach illustrated in Figure 2-5, including the imitation learning layer, the tuned RFDM/3 layer and the pre-programmed turning layer, is applied to the task of valve turning in underwater environment[7].

[7] A video of the real-world experiments reported in this section is available online at my personal website `http://www.ahmadzadeh.info/videos`.

2.11.1 Experimental Setup

The capabilities of the proposed approach is validated on Girona500 which is a compact and lightweight AUV with hovering capabilities, reconfigurable propulsion system, and mission-specific payloads [55]. Girona500 is equipped with typical navigation sensors (e.g. Doppler Velocity Log, pressure gauge) and basic survey equipment (profiler sonar, side scan sonar, video camera and sound velocity sensor). As illustrated in Figure 2-30, the thruster layout in Girona500 is equipped with 5 thrusters including two vertical thrusters to actuate the heave, one lateral thruster for the sway, and two horizontal thrusters for the surge and the yaw. In addition, the AUV is equipped with a vision sensor, a manipulator and a gripper which can be seen in Figure 2-29. The valve which is attached on a panel is analogous to the other valves used in previous experiments in the lab. Instead, in this case the panel contains more than one valve that a higher level planner can choose among them and perform a valve turning task on. The pose of the valve is detected using feature detection techniques either through AR-markers attached on the panel or through SIFT[8] features.

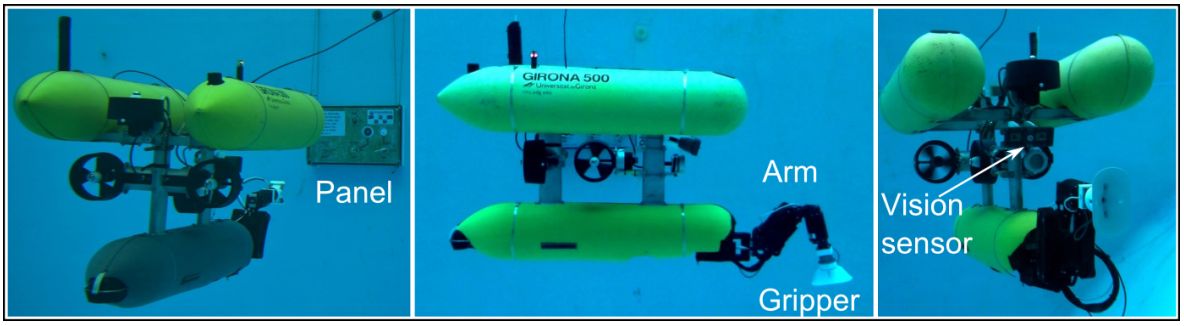


Figure 2-29: A set of snapshots showing the the Girona500 AUV, the manipulator arm and the gripper, the vision sensor and the panel in the underwater experiments. For more information, see Section 2.11.1

[8]SIFT stands for Scale-Invariant Feature Transform which is an algorithm for detecting and describing local features in images [56].

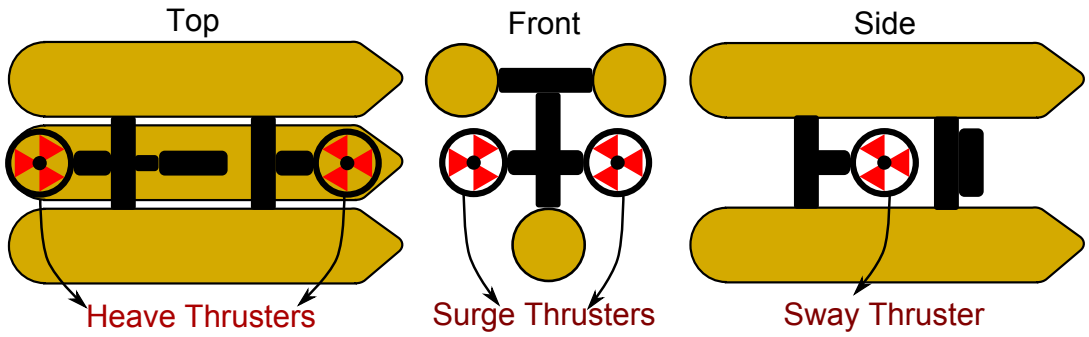


Figure 2-30: A schematic of the Girona500 AUV showing the thruster layout used in the underwater experiments. The sway thruster is used to create lateral disturbances. For more information, see Section 2.11.1

In order to learn the reaching skill, kinesthetic teaching is used for recording demonstrations through teleoperation. The learning method is applied to the set of recorded demonstrations. A high-level planner, detects the panel and navigates the AUV in front of it. From this point on, the imitation learning layer reproduces trajectories to reach the valve by moving the AUV and the arm. It means that first the AUV moves towards the panel and then at a certain distance the arm starts reaching the valve. The same process happens for the retracting behavior, first the arm retracts from the target and then the AUV moves backward. The turning phase is pre-programmed that allows the robot to turn the valve with a specific threshold in the torque magnitude.

2.11.2 Experiment with Lateral Oscillations of the AUV

In the first underwater experiment, the effect of underwater current or disturbances during the valve turning is analyzed. To generate relative movement between the valve and the gripper, the sway thruster of the AUV is used. By applying sudden lateral disturbances to the vehicle, the relative movement between the valve and the gripper increases and the RFDM system reacts to the uncertainty. A schematic illustrating the lateral oscillations is shown in Figure 2-31. It is worth noting that while the AUV is executing the autonomous valve turning, the manual movements applied by a tutor does not affect the process. By giving a higher priority to the movements generated by the human, the robot firstly executes the commands received from the human operator and immediately switches back to the autonomous task.

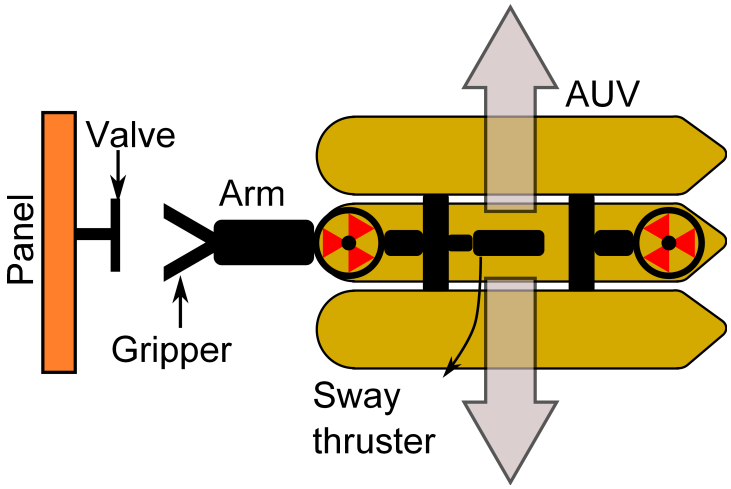


Figure 2-31: A schematic of the Girona500 AUV from top-view illustrates the panel, valve, arm, and AUV. Lateral disturbances applied to AUV using the sway thruster which generate relative movement between the gripper and the valve. For more information, see Section 2.11.2.

Figure 2-32 shows a set of snapshots during the execution of the valve turning task in which the lateral disturbances are applied to the robot in order to increase the relative movements between the gripper and the valve. The raw data representing the movements of the AUV along 3 axes are recorded during the execution of the task and are depicted in Figure 2-33.

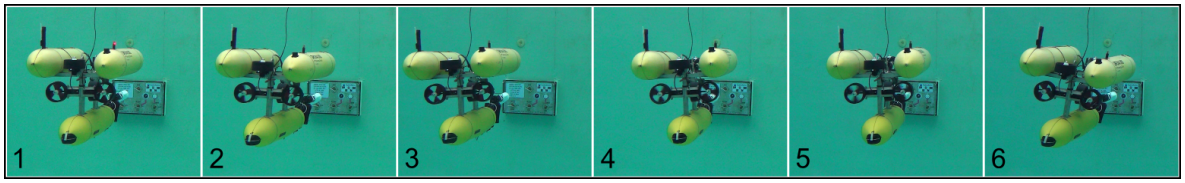


Figure 2-32: Lateral disturbances applied to AUV using the sway thruster. For more information, see Section 2.11.2.

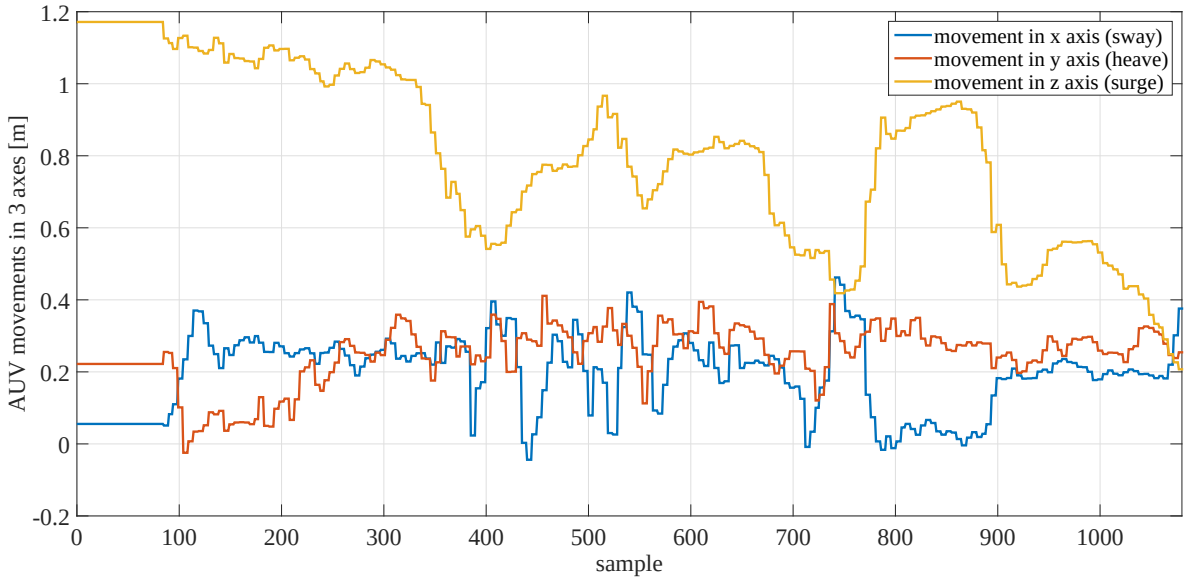


Figure 2-33: Movements of the AUV in 3 axes recorded during the execution of the valve turning task in the first underwater experiment. For more information, see Section 2.11.2

Figure 2-34 illustrates the three inputs and the output of the RFDM/3 during the execution of the task. The first input is the distance between the gripper and the valve. It can be seen that the AUV gradually moves towards the valve, but in several situations due to the increase of uncertainty the reactive system commands the AUV to go backward. The second input represents the relative movements between the gripper and the valve. The lateral oscillation applied to the robot can be seen in this subplot that is the main factor causing uncertainty in this experiment. The third input represents the sensor delay which is remained in a normal condition during the execution of the task. The output shows the reactive decisions generated by the reactive system, RFDM/3. It can be seen that the reactive decisions become negative when the oscillation of the relative movement increases.

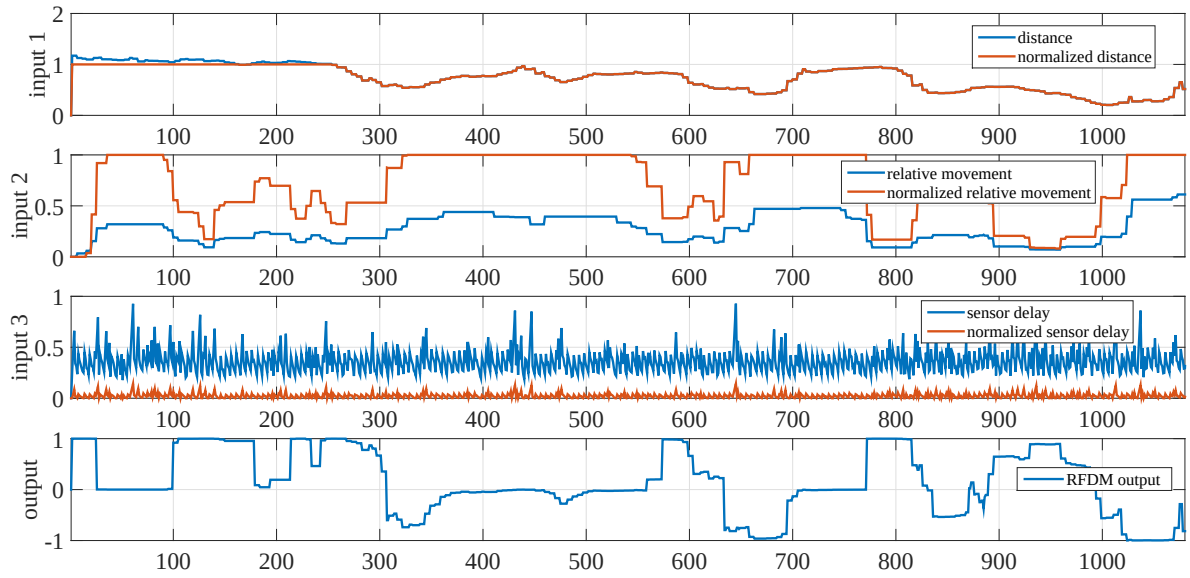


Figure 2-34: Oscillating the AUV laterally through teleoperation provides the opportunity to investigate the behavior of the RFDM/3 system under uncertainties caused by increasing the relative movement. For more information, see Section 2.11.2

2.11.3 Experiment with Vertical Oscillations of the Panel

In the next underwater experiment, again the effect of underwater currents or disturbances during the valve turning is investigated. This time the relative movement between the valve and the gripper is generated by moving the panel on which the valve is installed. A tutor moves the panel up and down while the AUV is executing the valve turning task. By applying sudden vertical oscillations to the panel, the relative movement between the valve and the gripper increases and the RFDM system reacts to the uncertainty. A schematic illustrating the vertical oscillations is shown in Figure 2-35. In this experiment, the movement was performed several times with various amplitude and in different distances between the gripper and the valve.

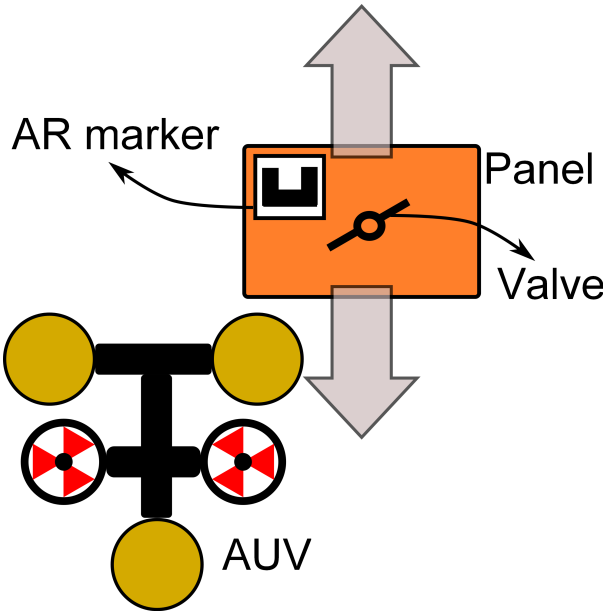
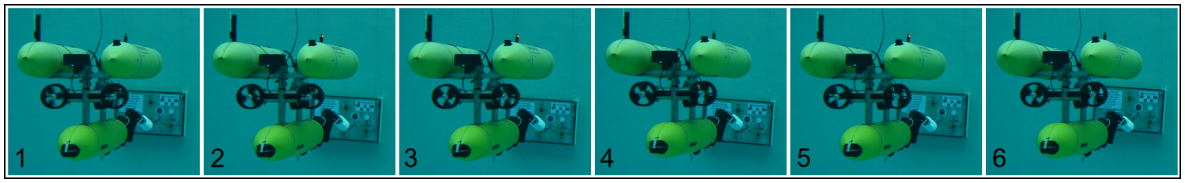
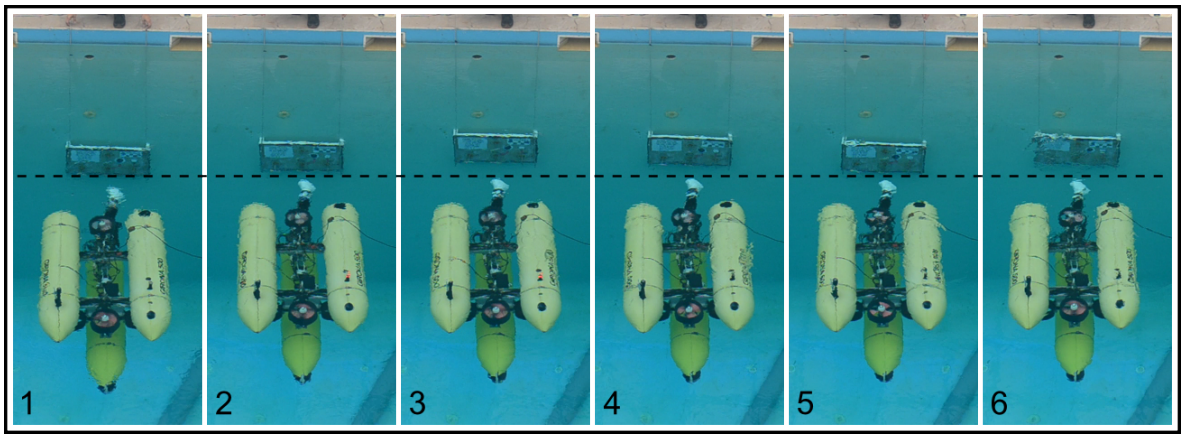


Figure 2-35: A schematic of the Girona500 AUV from back-view illustrates the panel, valve, AR marker, and AUV. Vertical disturbances applied to the panel by a human expert which generates relative movement between the gripper and the valve. For more information, see Section 2.11.3.

Figure 2-36 shows a set of snapshots during the execution of the valve turning task in which the vertical disturbances are applied to the panel in order to increase the relative movements between the gripper and the valve. The raw data representing the movements of the AUV along 3 axes are recorded during the execution of the task and are depicted in Figure 2-37.



(a) Snapshots captured from the control room.



(b) Snapshots captured from above the pool.

Figure 2-36: Disturbances applied to the panel by a human expert to simulate the water currents and disturbances applied to AUV. Snapshots in 2-36b and 2-36b are taken from different experiments. The dashed line in Figure 2-36a is a baseline that indicates the movement of the panel. For more information, see Section 2.11.3.

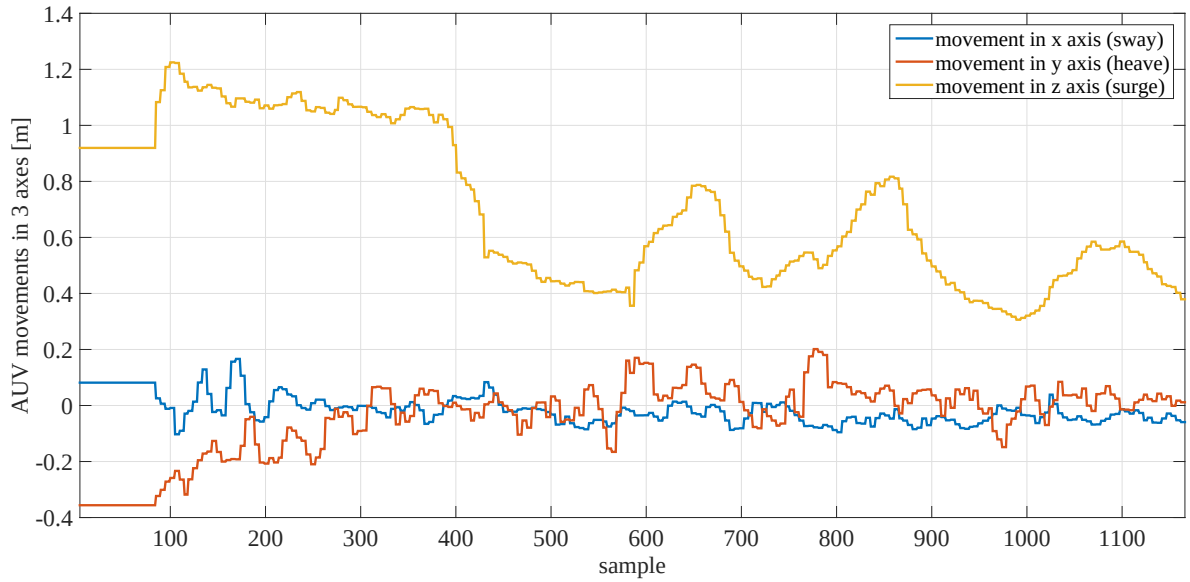


Figure 2-37: Movements of the AUV in 3 axes recorded during the execution of the valve turning task in the second underwater experiment. For more information, see Section 2.11.3

Figure 2-38 illustrates the three inputs and the output of the RFDM/3 during the execution of the task. The first input is the distance between the gripper and the valve. It can be seen that the AUV gradually moves towards the valve, but in several situations due to the increase of uncertainty the reactive system commands the AUV to go backward. The second input represents the relative movements between the gripper and the valve. In this experiment, the main factor causing uncertainty is the vertical oscillation applied to the panel which is detected by the pose estimation through the vision sensor and can be seen in this plot. The third input represents the sensor delay which is remained in a normal condition during the execution of the task. The output shows the reactive decisions generated by the reactive system, RFDM/3. It can be seen that the reactive decisions become negative when the oscillation of the relative movement increases.

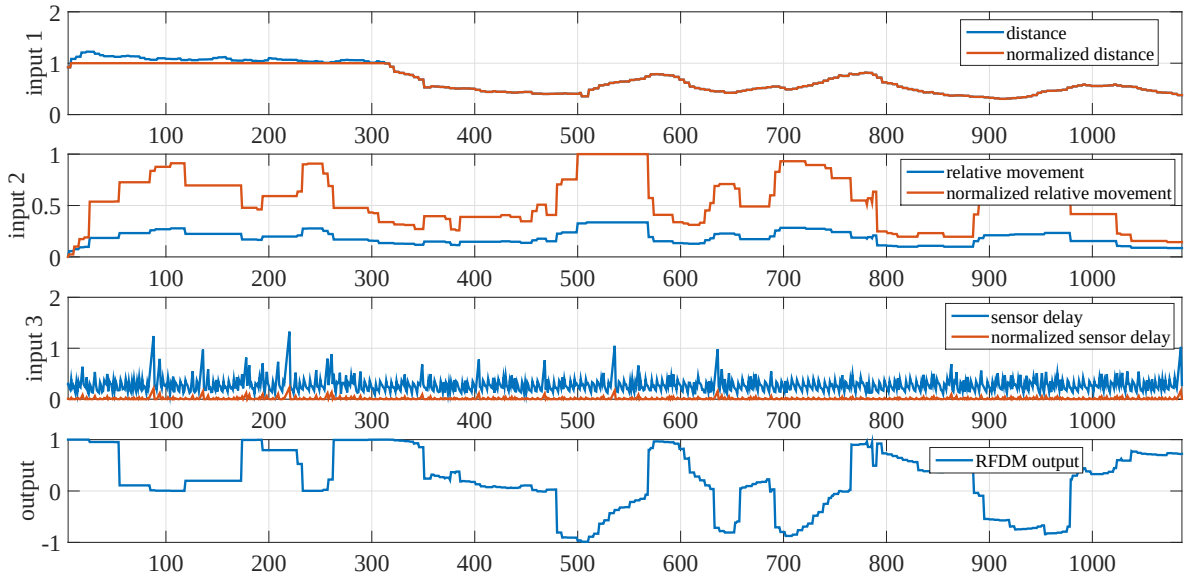


Figure 2-38: Oscillating the panel vertically provides the opportunity to investigate the behavior of the RFDM/3 system under uncertainties caused by increasing the relative movement. For more information, see Section 2.11.3

2.11.4 Experiment with Occluding the Valve

In the next underwater experiment, the effect of sensor delay during the execution of the task is investigated. The sensor delay implies the fact that the valve cannot be detected by the vision sensor due to poor visibility or occlusion. Consequently, the pose of the

valve is not updated in a executable period which implies that any further movements can endanger both the robot and the valve. Although the sensor delay could be simulated by simply disabling the marker detection module in the vision layer, to have a more realistic scenario in this experiment, the main panel is occluded using another panel by a human expert. A schematic illustrating the setup of the experiment is shown in Figure 2-39. The main panel has been occluded several times with respect to different distances between the gripper and the valve.

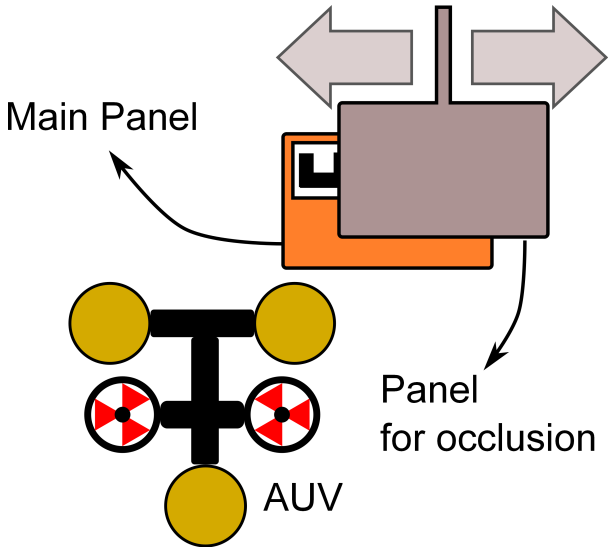


Figure 2-39: A schematic of the Girona500 AUV from back-view illustrates the main panel, the alternative panel used for occluding the valve, and AUV. For more information, see Section 2.11.4.

Figure 2-40 shows a set of snapshots during the execution of the valve turning task in which the valve is occluded by a panel in order to increase the sensor delay. The raw data representing the movements of the AUV along 3 axes are recorded during the execution of the task and are depicted in Figure 2-41.

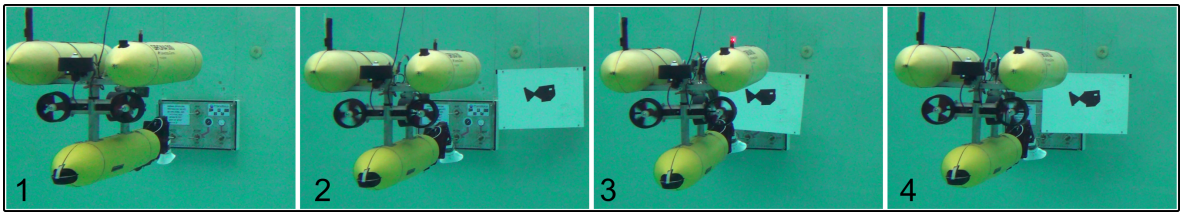


Figure 2-40: A set of snapshots representing the experiment with covering the valve using a panel to investigate the behavior of the RFDM/3 system under uncertainties caused by sensor delay. For more information, see Section 2.11.4

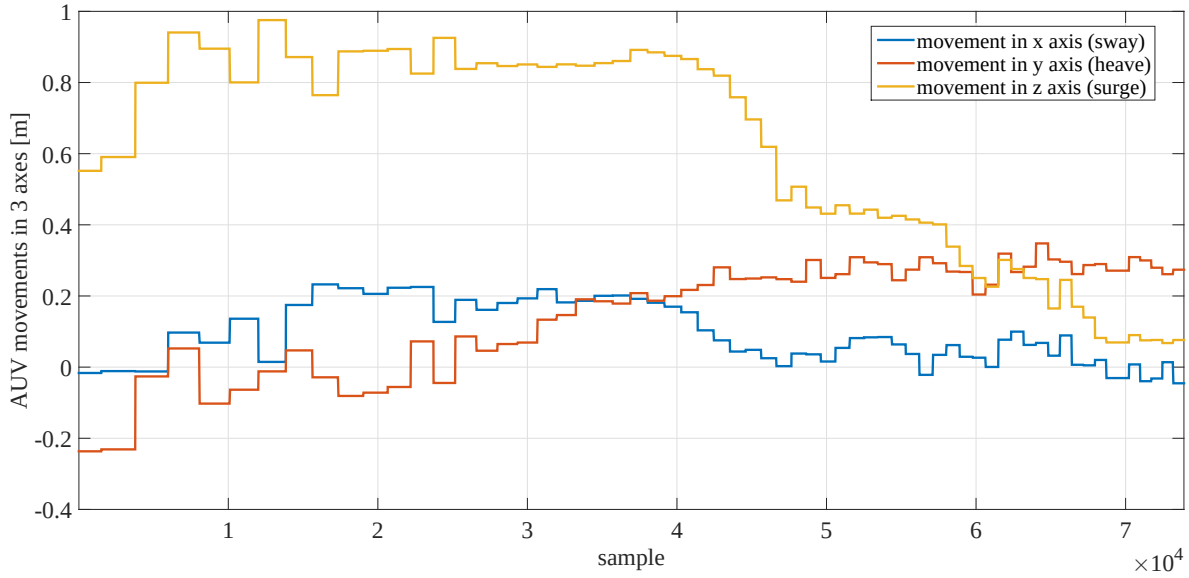


Figure 2-41: Movements of the AUV in 3 axes recorded during the execution of the valve turning task. For more information, see Section 2.11.4

Figure 2-42 illustrates the results of the experiment including the inputs and output of the RFDM/3 during the execution of the task. The first input is the distance between the gripper and the valve. It can be seen that the AUV gradually moves towards the valve, but in several situations due to the increase of uncertainty the reactive system commands the AUV to go backward. The second input represents the relative movements between the gripper and the valve which is remained in a normal condition during the execution of the task. The third input represents the sensor delay which in several situation is increased by occluding the valve. The output shows the reactive decisions generated by the reactive system, RFDM/3. It can be seen that the reactive decisions become negative when the sensor delay grows and becomes positive when the delay decreases.

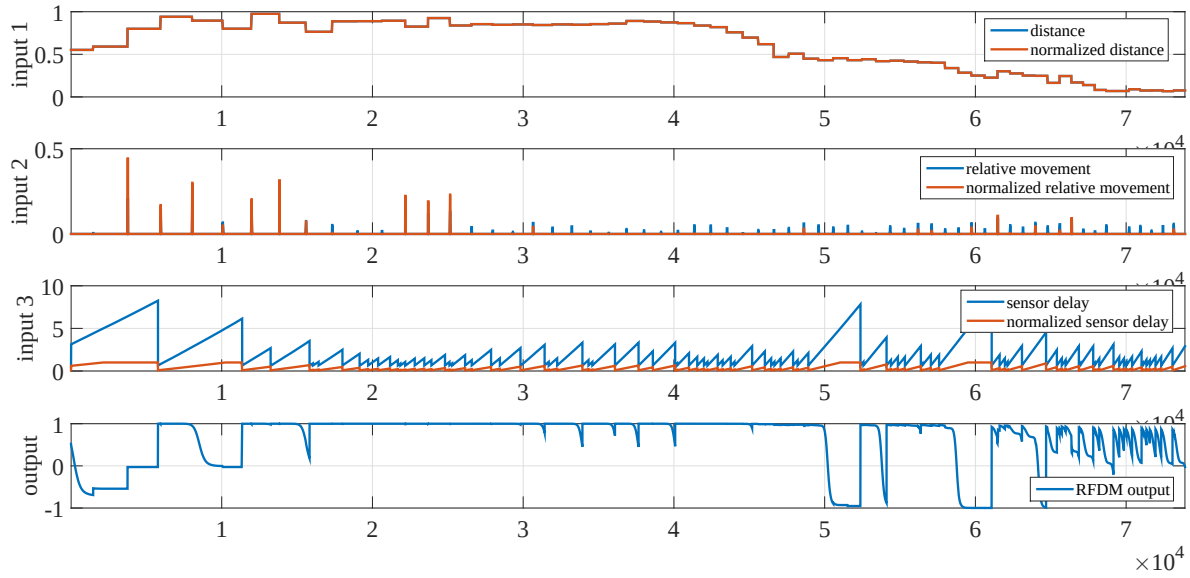


Figure 2-42: Covering the valve using another panel provides the opportunity to investigate the behavior of the RFDM/3 system under uncertainties caused by sensor delay. For more information, see Section 2.11.4

2.12 Chapter Summary

In this chapter, a hierarchical learning approach has been proposed that enables a robot to acquire a reactive behavior for coping with uncertainties. The proposed approach has been devised for performing the challenging task of autonomous robotic valve turning. The proposed hierarchical architecture consists of three main layers each of which realizes specific subtasks to improve the autonomy of the system. In the first layer, the robot acquires novel motor skills for approaching and grasping the valve through kinesthetic teaching based on imitation learning. In addition, a hybrid force/motion control strategy has been utilized in this layer to dissipate undesired forces and torques during the turning phase. A Reactive Fuzzy Decision Maker (RFDM) has been devised in the second layer which reacts to uncertainties caused by disturbances, noise, and sudden movements in the environment. The reactive layer observes the relative movements between the robot and the valve, sensor delays and applied forces/torques to the valve according to the distance between the valve and the robot. RFDM evaluates the dynamic behavior of the system and modulates the robot's movements accordingly by changing the direction

and rate of the movement. Three variants of the proposed RFDM system have been developed throughout this chapter. Each variant utilizes a different set of inputs and a various structure according to the provided source of information in the experiments. The third layer tunes the RFDM system using optimization algorithms based on expert knowledge. Conscious knowledge of a human expert is utilized to devise the main rules of the system. Whereas, the subconscious knowledge of the human expert is used to tune the rules. The validity and performance of the proposed approach have been successfully demonstrated through several real-world valve turning experiments both in the laboratory and in underwater environments. It has been shown that learning such reactive behavior effectively improves the efficiency and autonomy of the system specially in underwater valve turning scenario.

"It is not the strongest or the most intelligent who will survive but those who can best manage change."

— Charles Darwin

3

Learning Reactive Behavior: A Fault-Tolerant Approach

3.1 Motivation

An autonomous system should be able to react not only to the external changes of the environment, but also to the internal changes in the system's states. In the previous chapter, a learning approach has been proposed that allows a robot to react to the disturbances, noises and sudden movements in the environment while executing a desired task. In this chapter[1], on the other hand, a learning framework is proposed that enables a robot to react to the internal changes during the mission. Since the main focus of this research is in the scope of the PANDORA project, in this chapter the subject of failure

[1] The main results of this chapter were previously published in [57–60].

recovery is investigated.

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. Although the failure can happen in all subsystems of the AUV, this chapter places the focus on the case of thruster failure. Thrusters may fail during deployment, thereby jeopardizing the mission. Thruster blocking, rotor failure, and flooded thrusters are some of the factors that can lead to a thruster failure in real missions [61].

A fault-tolerant strategy enables a system to continue its intended operation, possibly at a reduced level, rather than failing completely. The fault-tolerant 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 [61–63]. After the failure is detected and isolated a fault-tolerant strategy must be considered to rescue the vehicle safely. 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 [61,64–67].

While the problem has been comprehensively studied in the case of actuator-redundant vehicles, the literature is still lacking a unifying approach if a broken thruster makes the AUV under-actuated [68]. In such case, more sophisticated techniques than allocation control is required. A few works are targeted at AUV controlled with surfaces [69–71]. Those methods are specific to the kinematics and dynamics models of the considered AUV.

Geometric control theory is used to generate trajectories for an AUV in order to overcome an actuator failure [72]. The method is an open-loop control strategy without

considering external disturbances and un-modeled dynamics of the system. Some methods provide an analysis of the thruster failure combinations for an AUV [73]. However, these method require a trajectory which may not be optimal for the faulty AUV.

In this chapter a learning-based framework for discovery of control policies to overcome thruster failure is proposed. Most existing methods have been particularly designed either for the case that the failure makes the AUV under-actuated or the AUV remains over-actuated. One of the advantages of this framework is that it is applicable in both cases. In addition, in most existing fault-tolerant control approaches, the trajectory generation is either ignored or the trajectory is given to the AUV manually. The proposed framework, on the other hand, includes the capability of generating optimal trajectories that can navigate the AUV towards a target with minimum cost while satisfying a number of objectives. In the case that the objectives do not conflict a Single-Objective Reinforcement Learning (SORL) approach based on linear scalarization of objectives is devised. The framework is then extended to find optimal solutions for a task including conflicting objectives using Multi-Objective Reinforcement Learning (MORL) and a vector objective function. Furthermore, a state-dependent policy is computed on-board and the fault-tolerant control loop is closed with the state feedbacks. So, contrary to many existing methods, the proposed framework can be evaluated in a real-world experiment as described in Section 3.4.10.

In many situations, a thruster is not fully broken but its functionality is reduced say by a percentage. Such thrusters are called partially broken. Another advantage of the proposed framework is that, it can utilize the remaining functionality of partially broken thrusters and find an optimal solution for such situations. The discovered solution can be more optimized than solutions in which a partially broken thruster is totally ignored.

Finally, the presented framework 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 framework is applicable.

The main advantages of the proposed framework can be listed as follows:

- Applicable on both under-actuated and over-actuated systems.

- Independent of an external trajectory generation module, because the discovered optimal policy itself includes a trajectory.
- Deals with independent, complementary and conflicting objectives through linear scalarization and objective vectorization.
- Computing the policy on-board and using a closed-loop control feedback, the approach is utilized in real-world experiments.
- Deals with both fully broken and partially broken thrusters.
- Can be implemented to various underwater robots, if a dynamic model of the system is available.

The rest of the chapter is organized as follows. Related work is reviewed in Section 3.2. The methodology and main components of the proposed framework based on Single-Objective Reinforcement Learning (SORL) and linear scalarization techniques are described in Section 3.3. The simulated and real-world experiments for a scalarized objective function are reported in Section 3.4. An extension of the presented framework is developed in Section 3.5 which utilizes Multi-Objective Reinforcement Learning (MORL) in order to deal with conflicting multiple objectives. The experimental results in simulation for MORL are reported in Section 3.6. And finally, conclusions of the research are drawn in Section 3.7.

3.2 Related Work

Most of the existing fault-tolerant schemes consider actuator redundancies, such that the vehicle remains over-actuated even if a fault occurs in one of the thrusters. For this category of problems a general solution has been found that is reallocating the desired forces on the vehicle over the working thrusters [61, 65, 66]. This method is also called control allocation or thruster reallocation. The existing solutions for control allocation problem can be divided in two main categories: solutions for linear effector models and solutions for nonlinear effector models. Generalize inverse technique through Singular

Value Decomposition (SVD) is used to find a solution for unconstrained linear control allocation [74, 75]. Simple saturation is an easy technique for dealing with constrained linear control allocation problem. However, finding a solution is not guaranteed. Redistributed pseudo-inverse method [76], direct allocation [75], and error minimizing using linear programming [77] or quadratic programming [78] are some existing solutions. As previously stated, such methods are applicable in case that even in the presence of the failure, the AUV remains over-actuated.

Reallocation of over-actuated vehicles is also formulated as optimization problems in order to use the remaining assets of the system. For instance Enns proposed a solution for controlling the allocation problem by approximating a system of linear equations subject to constraints for redundant actuators [79]. For each controlled axis there is one equation with actuation rate and position limit constraints. Linear programming is used to find a feasible solution and quadratic programming is employed to approximate a solution. Other groups also used constrained optimization techniques for the problems that are not easy to solve using iterative numerical optimization in real-time with high sampling rate [75, 80]. To address this issue some real-time iterative optimization techniques have been proposed [81, 82]. Several implicit and explicit methods for controlling an over-actuated underwater vehicle has been discussed by Fossen and Johansen [67, 83]. They address the problem of optimizing the used power subject to actuator rate and position constraints.

On the contrary, if a broken thruster makes the AUV under-actuated more sophisticated techniques than control allocation or thruster reallocation are required. In order to control the AUV, some groups use surfaces [69–71]. Those methods depends on the kinematics and dynamics models of the considered AUV. The proposed method in this chapter, on the other hand, employs the dynamics model for simulation, but not for derivation of the controller. It also utilizes a linear function approximator to represent a control policy, whose parameters are learned according to the model of the system and the specified task. Pettersen and Egeland proposed a continuous time-varying feedback law for stabilizing the position and orientation of an under-actuated surface vessel having only two thrusters [84]. The feedback law does not depend on parameters of the model and is

robust to model parameter uncertainty.

Leonard proposed an algorithm that provides motion control laws that rely only on the remaining assets of the system [85, 86]. The algorithm constructs open-loop motion control laws for point-to-point maneuvers that are parameterized by the vehicle's actuation capability. So when the failure makes the AUV under-actuated, the system switches to the new motion control law. One assumption is that each actuator can be independently actuated. The algorithm relies on kinematics and dynamics equations of the vehicle in order to reposition and reorient the under-actuated system using small-amplitude periodically time-varying force and torque inputs. This method is considered as a geometric approach to analyse specific motion properties of underwater vehicles. However, the algorithm does not generate a trajectory.

A time optimal approach and a motion planning approach proposed by Cyba *et al.* address the problem of motion planning in under-actuated scenario [87]. These approaches utilize concatenation of kinematic motions. They linked the optimization problem to the geometric framework using singular extremals.

The problem of position-tracking control of under-actuated AUVs in the presence of unknown ocean currents in a horizontal plane is studied by Bi *et al.* [88]. The approach includes a position controller and a current observer based on the Lyapunov stability theory. In order to design the controller for the nonlinear dynamics system the back-stepping technique [89] is used. Although the presented approach is not dealing with thruster failure problem, it presents a control system similar to many controllers devised for failure scenario.

Choi and Kondo, provide an analysis of the thruster failure combinations for a vehicle similar to Girona500 [73]. Their method is simple and in some circumstances applicable to the failure recovery problem under investigation in this chapter. 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 matter. 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. The proposed method in this chapter generates optimal trajectories, that can be used to accomplish the given task achieving

the lowest cost in the presence of the fault.

Andonian *et al.,* used geometric control theory to design trajectories for an AUV for performing its recovery while experiencing a thruster failure [72]. The AUV is modeled as a forced affine connection control system, and the control strategies through the use of integral curves is developed. During the failure recovery, the forced affine connection control system is adjusted to the new situation and the vehicle computes Decoupling Vector Fields (DVFs)[2]. The DVFs required for the kinematic control of the under-actuated AUV are presented. The conducted experiment includes an AUV equipped with eight thrusters following a trajectory in a simulated environment. The trajectory is designed manually by an expert. At the end of the path four of the thrusters fail which make the AUV under-actuated. The goal of the introduced failure recovery system is to recover the AUV back to the surface. The presented approach tries to find a control system for the new kinematics of the system that finally should follow the previously navigated trajectory. Despite the fact that the AUV cannot follow the trajectory precisely, the simulation showed that it can reach the surface safely. The presented geometric control is an open-loop control strategy, therefore its validity is mainly theoretical, because of the inevitable presence of unpredictable dynamics and external disturbances. The proposed framework in this chapter, on the other hand, uses feedbacks from states of the system in order to cope with external disturbances and water currents. Also it is independent from an external trajectory generation module by generating an optimal trajectory for the damaged vehicle.

Koos *et al.* proposed T-Resilience algorithm that aims at making the optimization process aware of the limits of the simulation [90]. It looks for the behaviors that only use reliable parts of robots and avoids behaviors that it is unable to achieve in the real-world. On the contrary, the model used in this proposed method includes the dynamics equation of the AUV together with identified hydrodynamic parameters of the system and the presented approach generates an optimal trajectory that takes the AUV to the target with minimum cost. Nonetheless, their method could be combined with the proposed open-loop policy in this chapter, providing an alternative way of having feedback control on a trajectory designed for the faulty AUV.

[2]A DVF is a kinematic reduction of rank one [72].

3.3 Methodology

Thruster failures reduce the mobility of underwater vehicles severely, and hence they cannot maneuver as previously prescribed. However, AUVs are supposed to be autonomous systems that can react to changes independently and learn new behaviors discretely. This chapter proposes a learning framework that enables a robot to react to thruster failures reactively, thereby improving the autonomy of the system. The proposed framework discovers fault-tolerance policies to overcome thruster failures using the functional assets of the system. The main goal of the proposed framework is to bring the AUV safely to a station where it can be rescued.

In order to deal with unstructured environments the robot should be able to learn through interacting with its surrounding environment. In such case, supervised learning approaches are not an adequate choice, because they require a set of labeled examples provided by an expert which is practically impossible. Instead, Reinforcement Learning (RL) enables an agent to learn the desired task through interaction. In other words, the robot senses the state of the environment and selects proper actions to achieve a desired goal. In RL, the goal which is provided by an expert, is to maximize a long-term reward over a specific horizon.

The proposed fault recovery module is framed in the context of model-based direct policy search for reinforcement learning. This framework, which is illustrated in Figure 3-1, comprises a fault detection module, a dynamic model of the vehicle, a parameterized representation for the control policy, a reward function, and an optimization algorithm. The fault detection module detects and isolates the faulty thruster and sends a signal to a higher level planner. The planner then decides whether to switch from the original controller of the system to fault-tolerant controller. The fault-detection module is explained in Section 3.3.2.

The dynamics model of the system is then reconfigured according to the current situation of the system. In other words, the generated commands (i.e. torques) by the approach are sent to the functional thrusters and faulty thruster do not receive any commands. Consequently, there is no need for changing the model. The dynamics

equations of the system are explained in Section 3.3.1. 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. The policy representation and the reward function are described in Sections 3.3.3 and 3.3.4 respectively. In addition, all the components of the fault recovery module are depicted in Figure 3-1.

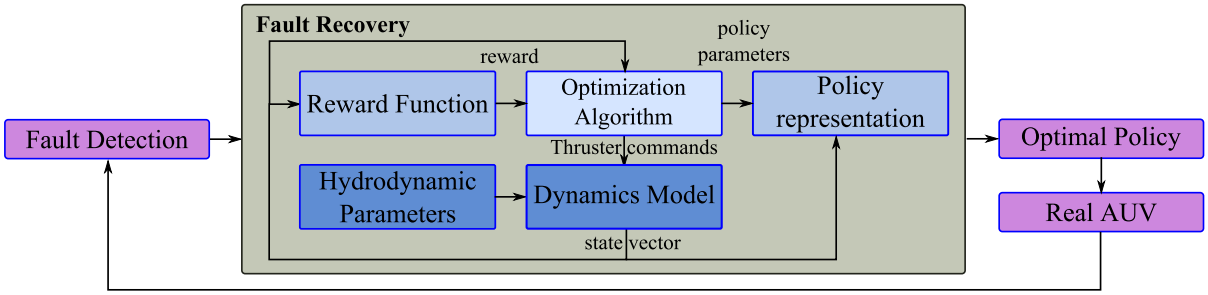


Figure 3-1: A diagram illustrating fault detection and fault recovery modules. The fault recovery module includes a number of elements such as policy representation, reward function and dynamics model of the system. For more information, see Section 3.3.

3.3.1 AUV Model

In this section a dynamic model of the 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. According to a classic reference in marine technology, an underwater vehicle can be modeled as a rigid body subject to external forces and torques while moving in a fluid environment [91]. The 6 DoF equations of motion for an underwater vehicle are given in a compact form as follows:

$$\dot{\eta} = \mathbf{J}(\eta)\nu \tag{3.1}$$

$$\mathbf{B}\tau = \mathbf{M}\dot{\nu} + (\mathbf{C}(\nu) + \mathbf{D}(\nu))\nu + \mathbf{g}(\eta) \tag{3.2}$$

$$\mathbf{M} = \mathbf{M}_{RB} + \mathbf{M}_A \tag{3.3}$$

$$\mathbf{C} = \mathbf{C}_{RB}(\nu) + \mathbf{C}_A(\nu), \tag{3.4}$$

where $\eta = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T$ is the generalized coordinates for a vehicle moving in 6 DoF with respect to the inertial frame and $\nu = \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^T$ 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 the control input vector or command vector. More details about the equations of motion can be found in [91].

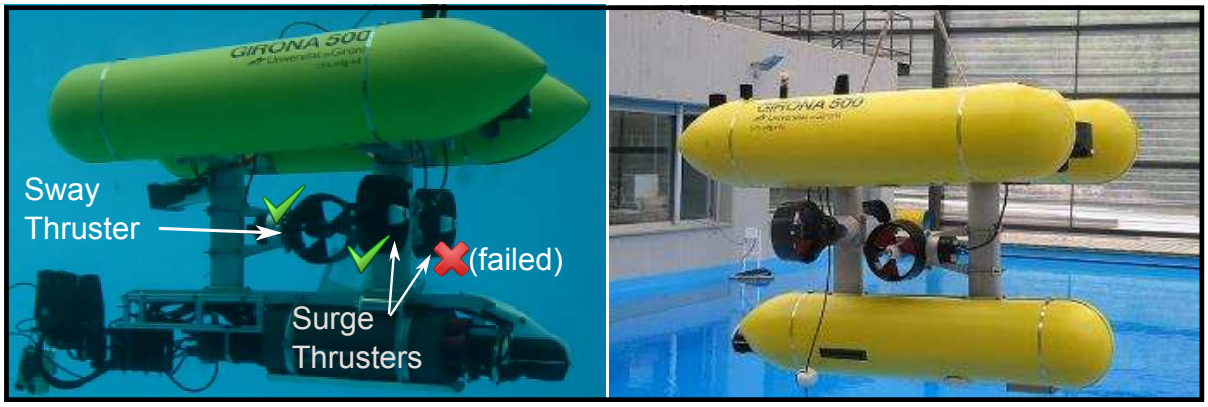


Figure 3-2: The Girona500 AUV equipped with 5 thrusters (3 are visible in the photo). In the conducted experiment, one of the surge thrusters is broken.

The experiments presented in this chapter are conducted by employing the Girona500 [55] AUV (see Figure 3-2). Girona500 is a reconfigurable AUV equipped with typical navigation sensors (e.g. DVL[3]), survey equipments (e.g. stereo camera) and various thruster layouts. As depicted in Figure 3-3, the selected thruster layout in this work consists of five thrusters:

[3]Doppler Velocity Log

two in heave direction, two in surge direction, and one 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 (3.4). The hydrodynamic parameters are extracted using an online identification method and are reported in [92]. In addition, in the modeling of the system it should be considered that Girona500 was designed to have passive stability in roll and pitch.

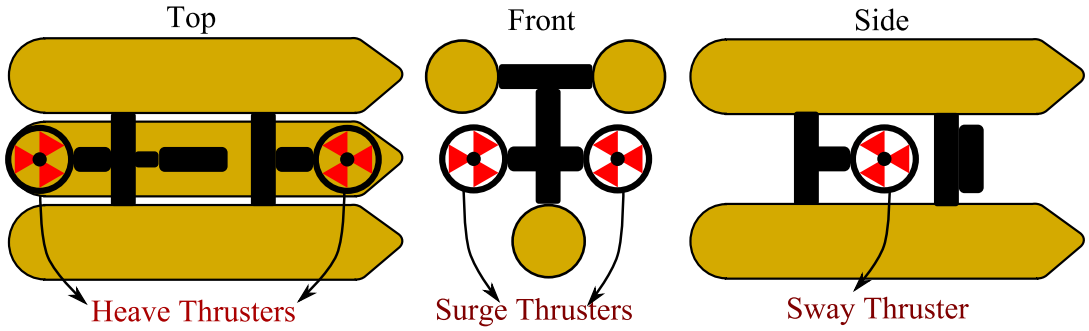


Figure 3-3: The selected thruster layout for Girona500 including five thrusters.

It is worth noting that for over-actuated systems, since the vehicle remains redundant even after thruster failure, most approaches operate on the matrix **B**, to obtain the required forces/torques by reallocating the command on the functional thrusters. This method is also known as thruster reallocation or control allocation which has been mentioned in Section 3.2. The simplest way to modify **B** is ignoring the columns correspondent to the faulty thrusters. In this chapter, on the other hand, a different approach is proposed that computes a new command function τ to reach a given target without modifying the matrix **B**. One of the advantages of this approach is that it is applicable in both cases of over-actuated and under-actuated vehicles.

3.3.2 Fault Detection Module

The process of monitoring a system in order to recognize the presence of a failure is called fault detection. As mentioned before, failure detection in underwater vehicles has been extensively studied [61–63], and so, it is out of scope of this thesis. Therefore, the assumption is made that, as depicted in Figure 3-1, the fault detection module continuously monitors all the thrusters and when a thruster is deemed faulty, the module

detects and isolates the faulty thruster and sends a signal to the fault recovery module. The task of the fault recovery module is then 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.

3.3.3 Policy Representation

Reinforcement learning (RL) in continuous state-space requires function approximation. In this technique a parameterized representation of the final solution is formed. The goal of the RL algorithm is to find a set of parameters that leads to an optimal solution. In direct policy search, a policy which is a representation of the desired solution is learned. The policy maps state vectors to actions in order to maximize the long-term reward. For the AUV the state vector is formed by concatenating $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$ vectors and the action is the control input vector $\boldsymbol{\tau}$.

Using linear function approximation, a policy Π can be represented as a weighted linear sum of a set of features known as basis functions as follows:

$$\Pi = \sum_{i=1}^{n_f} \omega_i \varphi_i = \boldsymbol{\omega}^T \boldsymbol{\varphi}, \tag{3.5}$$

where $\boldsymbol{\varphi} = [\varphi_1, \ldots, \varphi_{n_f}]^{\mathrm{T}}$ is a vector of n_f basis functions and $\varphi_i \in \boldsymbol{\varphi}$ is the i^{th} basis function, $\boldsymbol{\omega} \in \mathbb{R}^{n_f}$ is a parameter vector and $\omega_i \in \boldsymbol{\omega}$ is the i^{th} parameter. The most common choices for basis functions include polynomial basis, radial basis, Proto-value, and Fourier basis schemes. In this chapter, Fourier basis scheme is utilized 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 [93].

In this section, univariate and multivariate Fourier bases are defined. The n^{th} degree Fourier expansion for approximating a periodic function f with period T is:

$$\bar{f}(x) = \frac{a_0}{2} + \sum_{k=1}^{n}\left[a_k \cos\left(k\frac{2\pi}{T}x\right) + b_k \sin\left(k\frac{2\pi}{T}x\right)\right] \tag{3.6}$$

$$a_k = \frac{2}{T}\int_0^T f(x)\cos\left(\frac{2\pi k x}{T}\right)dx \tag{3.7}$$

$$b_k = \frac{2}{T}\int_0^T f(x)\sin\left(\frac{2\pi k x}{T}\right)dx, \tag{3.8}$$

where x is a variable (i.e. a state variable) and $\bar{f}$ denotes the corresponding approximate f. The n^{th} order Fourier expansion of a multivariate function $F(\mathbf{x})$ with period T in d dimensions is:

$$\bar{F}(\mathbf{x}) = \sum_{\boldsymbol{\varsigma}}\left[a_{\boldsymbol{\varsigma}} \cos\left(\frac{2\pi}{T}\boldsymbol{\varsigma}\mathbf{x}\right) + a_{\boldsymbol{\varsigma}} \sin\left(\frac{2\pi}{T}\boldsymbol{\varsigma}\mathbf{x}\right)\right], \tag{3.9}$$

where $\boldsymbol{\varsigma} = [\varsigma_1, \ldots, \varsigma_d]$, $\varsigma_j \in [0, \ldots, n]$, $1 \leq j \leq d$, and $\bar{F}$ denotes the corresponding approximate F. Since a full Fourier expansion includes both sin and cos terms, the number of basis functions for the n^{th} order expansion with d variables is $2(n+1)^d$. This number can be reduced to $(n+1)^d$ by dropping either of terms for each variable [93]. Thus the n^{th}order Fourier expansion of d variables can be formulated as follows:

$$\varphi_i(\mathbf{x}) = \cos\left(\pi \boldsymbol{\varsigma}^i \mathbf{x}\right), \tag{3.10}$$

where $\boldsymbol{\varsigma}^i = [1, ..., \varsigma_d]$, $\varsigma_j \in [0, ..., n]$, $1 \leq j \leq d$. Since cos is bounded in $[-1, 1]$, the defined basis in (3.10) is easy to compute accurately.

The approximated policy which is a function of parameter vector $\boldsymbol{\omega}$ and state variables $\{\boldsymbol{\eta}, \boldsymbol{\nu}\}$, represents the control input vector $\boldsymbol{\tau}$ of the AUV. However, employing all the state variables (i.e. $\begin{bmatrix} x & y & z & \phi & \theta & \psi & u & v & w & p & q & r \end{bmatrix}$) is not necessary and the proposed method utilizes a subset of state variables called observation vector $\mathbf{o}$ (in this chapter $\mathbf{o} = [x \; y \; \psi \; u \; v]$). For more details about the function approximation using Fourier basis refer to [93].

3.3.4 Reward Function

The performance of the vehicle is measured through a reward function R, formed as:

$$R = \sum_{t=0}^{T} r_t(\mathbf{o}_t)\Big|_{\Pi}, \tag{3.11}$$

where r_t is the immediate scalar reward at time t which depends on the observed state vector $\mathbf{o}$. The reward function primarily represents a single objective. However, linear scalarization technique provides the advantage of dealing with multiple objectives using different weighting factors as follows:

$$r_t = \sum_{i=1}^{n_o} \xi^i r_t^i, \tag{3.12}$$

where n_o is the number of objectives, r_t^i is the i^{th} immediate scalar reward for a single objective, ξ^i weighs the i^{th} reward, r_t is the total immediate scalar reward, and depends on the current observed state vectors $\mathbf{o}_t$, 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 maximize the cumulative reward R over horizon T.

Equation (3.12) suggests that various rewards affect the behavior of the system differently according to their weights. Consequently, choosing a proper weight for each objective is difficult because they can affect the final solution unpredictably. To address this problem and since linear scalarization can be used only for not conflicting objectives, in Section 3.5, a framework utilizing multiple conflicting objectives is presented.

Many different definitions of the immediate reward are possible. The following definition is used in this section and the experiments in Section 3.4:

$$r_t = \xi^1 r_t^1 + \xi^2 r_t^2 \tag{3.13}$$

$$r_t^1 = \frac{1}{\| \mathbf{p}_t - \mathbf{p}_d \| + \epsilon} \tag{3.14}$$

$$r_t^2 = \frac{1}{\| \mathbf{v}_t - \mathbf{v}_d \| + \epsilon}, \tag{3.15}$$

where $\|\,.\,\|$ indicates 2-norm Euclidean distance, $\mathbf{p}_t$ and $\mathbf{p}_d$ are the current and the desired position vectors, $[x, y]$, respectively. $\mathbf{v}_t$ and $\mathbf{v}_d$ are the current and the desired linear velocity vectors, $[u, v]$, respectively. The first part of the reward is defined to navigate the AUV towards the target. The second part, however, ensures that the AUV reaches the target with minimum final velocity.

3.3.5 Optimization Algorithms

The proposed framework employs a model-based policy search approach in which 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 reward function is the other degree of freedom of the proposed approach which was discussed in the previous section. Another element of the proposed fault-tolerant framework is the algorithm required for finding an optimal solution. 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 ω.

Three different optimization algorithms are implemented to compare the quality and the computational feasibility of the proposed solution for online discovery of the fault-tolerant policy. The chosen algorithms consist of a derivative-free optimization algorithm introduced by Leonetti *et al.* [94], the well-known Simulated Annealing [95], and the powerful stochastic evolutionary algorithm, Differential Evolution [96]. The first algorithm was used for online identification of Girona500 as well [92]. 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. The mentioned optimization algorithms are briefly described in this section.

Modified Price's Algorithm

Modified Price (MP)[4] 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 [94]. MP is a combination of a global and a local derivative-free method, designed for optimization of non-linear, multi-modal, and multivariate functions. The global part of the MP algorithm which has been introduced by Brachetti *et al.* is a population-based method [47]. Recently, Leonetti *et al.* combined this global search with a deterministic local search [94]. 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 section, the initial population size for the algorithm is set to 20 times the number of the parameters and other parameters are set according to [94]. MP has been used in previous chapter in Section 2.5.2 for tuning the proposed reactive system. Also, for more information about MP, see Appendix E.

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 [95]. 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 [97]. Although there are different variants of this algorithm [98], in this chapter the standard SA algorithm is utilized. The initial temperature, re-annealing interval, and temperature function options are set to 100, 100, and *'fast annealing'* scheme respectively. For more information about SA, see Appendix D.

Differential Evolution

The Differential evolution (DE) algorithm, proposed by Storn and Price is a simple yet powerful population-based, stochastic, heuristic evolutionary algorithm, which is an

[4] The original algorithm is named after one of its creators, Kenneth Price [96].

efficient and effective global optimizer in the continuous search domain [96]. DE has been successfully applied to optimization problems including non-linear, non-differentiable, non-convex, and multimodal functions. 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 (two crossover schemes and five mutation strategies) [96]. According to the classic notation `DE/x/y/z`, the particular variant of DE that is used in this chapter 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 section, the standard implementation of the DE algorithm is utilized [99]. The number of populations N_p is set to 10 times the number of the parameters, the weighting factor is considered as $F \in [0.8, 0.9]$ and the crossover constant is set $C_r \in [0.9, 1]$ according to [100, 101]. DE is explained in more details in Section 3.5.5. For more information about DE, see Appendix A.

3.3.6 Online Procedure

In the presented scenario, when a thruster is deemed faulty, a function R is created to represent the reward for 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 R. As will be seen subsequently, the secondary objective enforces the robot to reach the desired point with a given velocity.

3.4 Experiments

As described previously, the presented framework includes a dynamic model of the AUV, a policy representation, a reward function and an optimization algorithm. The dynamics model of the system which has been described in Section 3.3.1 is implemented as an on-board module for performing the simulations on the robot. The policy is represented with a linear function approximator using Fourier basis functions as explained in Section 3.3.3. When a thruster is deemed faulty, a function R is created to represent the reward for a path to the target location. The reward function can contain more than one objective using linear scalarization technique as explained in Section 3.3.4. Finally, an optimization algorithm is selected and employed to find an optimal policy from simulation. To validate the reliability and performance of the proposed framework, in this section, a set of experiments in simulated and real-world environments is conducted.

3.4.1 Setup

The experiments are conducted on the dynamic model of Girona500 as presented in (3.4), whose parameters have been identified in [92]. In girona500 the heave thrusters not only control the vehicle in heave direction, but also compensates the buoyancy force and keep the vehicle submerged. Since a broken heave thruster makes the AUV float, all of the experiments are designed so that the thruster failure occurs in the horizontal plane, while the heave movements of the AUV are always controlled by its original controller. Therefore the 6 DoF model is used for simulating the dynamics behavior of the AUV, but the devised experiments utilize only 3 DoF in horizontal plane. In this chapter the case is studied where the right surge thruster of the AUV is deemed faulty. However, the proposed framework can be easily applied in the case where instead other thrusters are broken. By such consideration, the Girona500 AUV can only navigate using the left surge and the sway thrusters. Thus the vehicle becomes under-actuated and any attempt to change the allocation matrix $\mathbf{B}$ would be ineffective. In all of the experiments in this section, the immediate reward is defined according to (3.15). In addition, $T = 60$ s is used, since all the target destinations are reachable in 60 seconds. Also, when the AUV reaches an area

close enough to the desired position, $\| \mathbf{p}_t - \mathbf{p}_d \| < 0.2$ m, the optimization algorithm is terminated. In addition, both ξ^1 and ξ^2 are initially set to 1.0 when $t < T$. The secondary objective is considered important only at the final state, $t = T$. In order to weigh the velocity objective with respect to the positional objective, ξ^2 is increased towards 100 at the final state.

3.4.2 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 Figure 3-4 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, the AUV is commanded to move 3.0 m in the surge direction (x-axis) while the thruster commands for all five thrusters of the robot are being recorded. Secondly, the right surge thruster is turned off and the experiment is repeated.

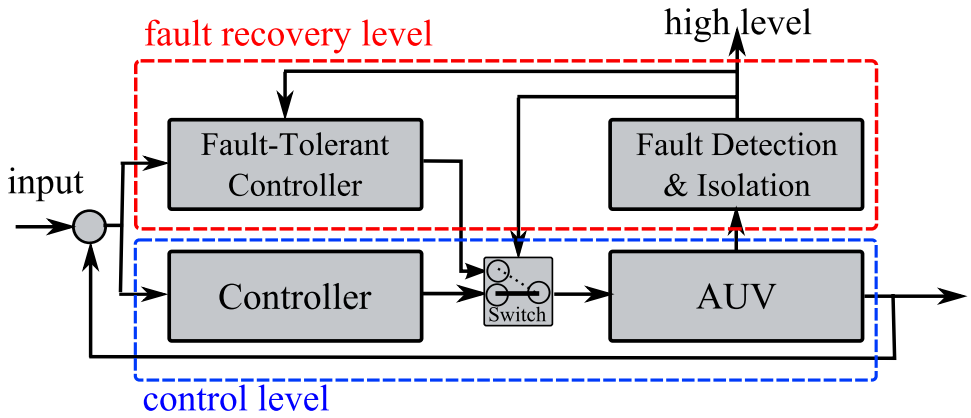


Figure 3-4: Control architecture of the AUV including the controller level and the fault recovery level. Once a failure is detected and isolated, the system employs the proposed fault-tolerant framework. For more information, see Section 3.4.2 and 3.4.4.

The recorded data is depicted in Figure 3-5. It can be seen that in the second experiment the governing controller of the system tries to use the same configuration of the thrusters that was used under normal condition. 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 Figure 3-4) needs to be concatenated to the control level architecture of the AUV (the dashed red box in Figure 3-4). 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.

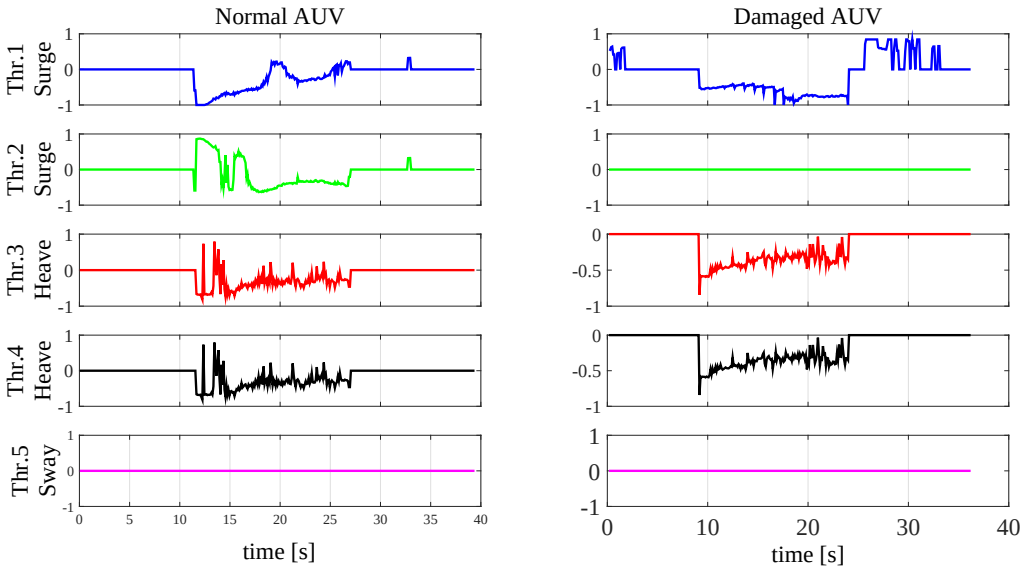


Figure 3-5: The recorded thruster commands for a normal AUV (left column) and a damaged AUV (right column - the right surge thruster is broken) while using the original controller scheme without failure recovery level. For more information, see Section 3.4.2.

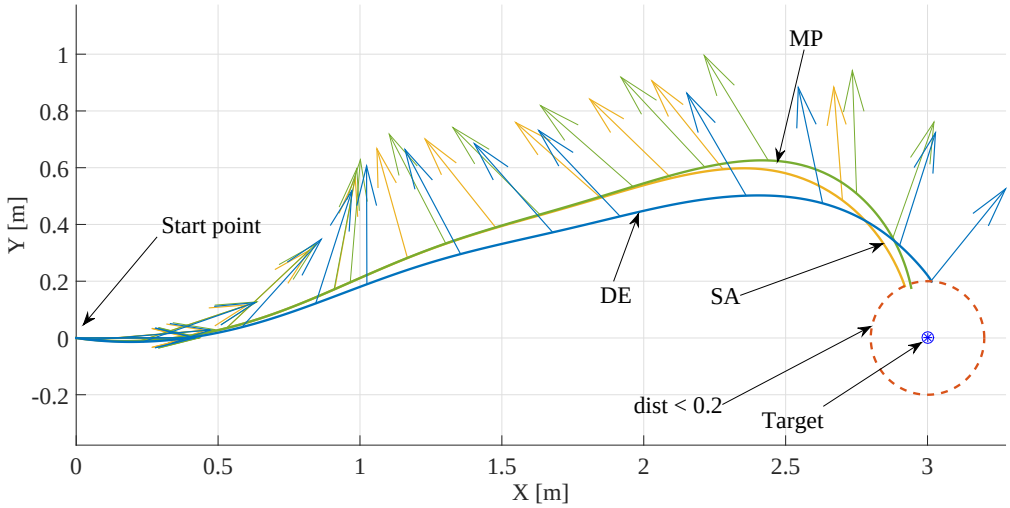
3.4.3 Time-dependent Policy

In this experiment, the policy is represented as a linear function approximator which depends only on time t,

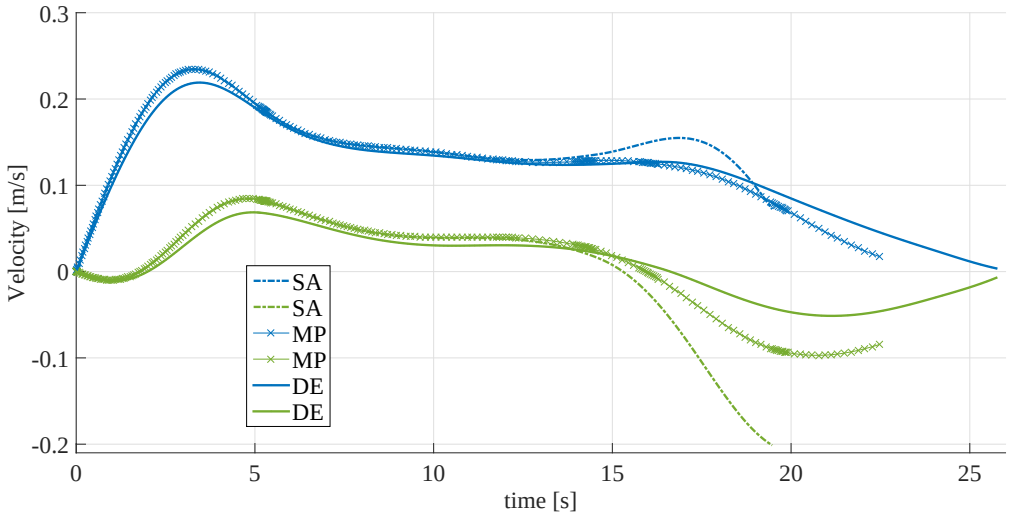
$$\Pi = \omega^{\mathrm{T}} \varphi(t). \tag{3.16}$$

In other words, the policy changes only with respect to time independent of other state variables. In this representation ω is the parameter vector and to represent $\varphi(t)$ a 3$^{\mathrm{rd}}$ order Fourier basis is employed [93]. The number of optimization parameters equals to eight (i.e. four parameters for each functional thruster). Applying the optimal time-

dependent policies computed by the three optimization algorithms, the trajectories and velocity profiles are depicted in Figures 3-6a and 3-6b. As it can be seen, the obtained velocity profiles are varied, whereas, the acquired trajectories are similar.



(a) Trajectories in 2D plane.



(b) Velocity profiles along X and Y axes.

Figure 3-6: Acquired results for the experiments with time-dependent policy using Simulated Annealing, Modified Price and Differential Evolution algorithms. The discovered solutions by the employed algorithms are similar. For more information, see Section 3.4.3.

Since all algorithms are stochastic, the results may converge to various solutions in different runs. So, the optimization process was repeated 50 times for each optimization

algorithm. The extracted statistical results in terms of number of function evaluations and best objective values are depicted in Figure 3-7. The results show that in this case, the Differential Evolution algorithm outperforms the others.

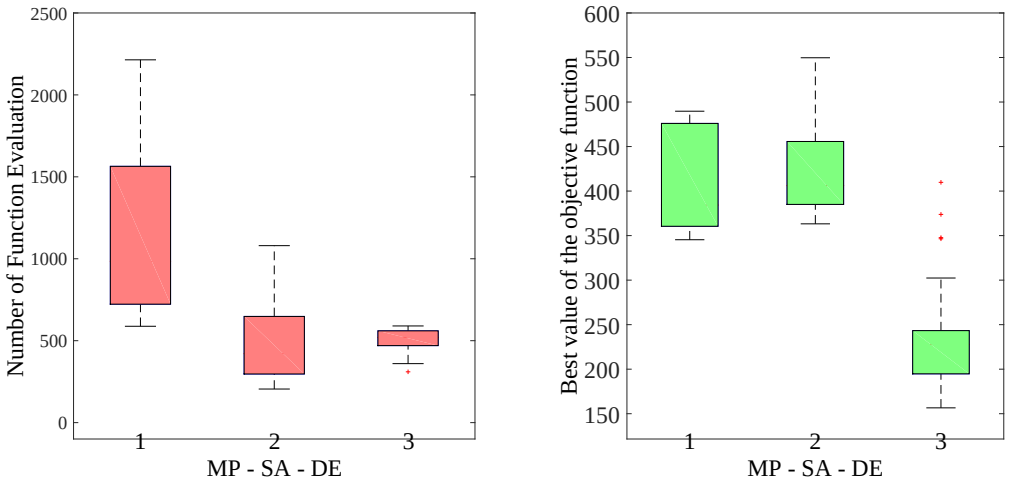


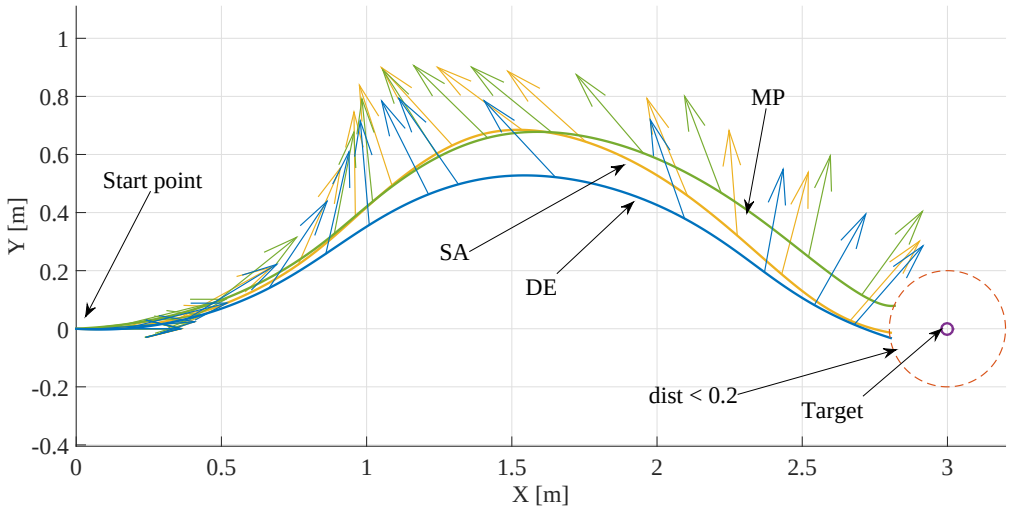
Figure 3-7: Acquired statistical results for the experiments with time-dependent policy representation over 50 runs using Simulated Annealing, Modified Price, and Differential Evolution algorithms. For more information, see Section 3.4.3.

3.4.4 State-dependent Policy

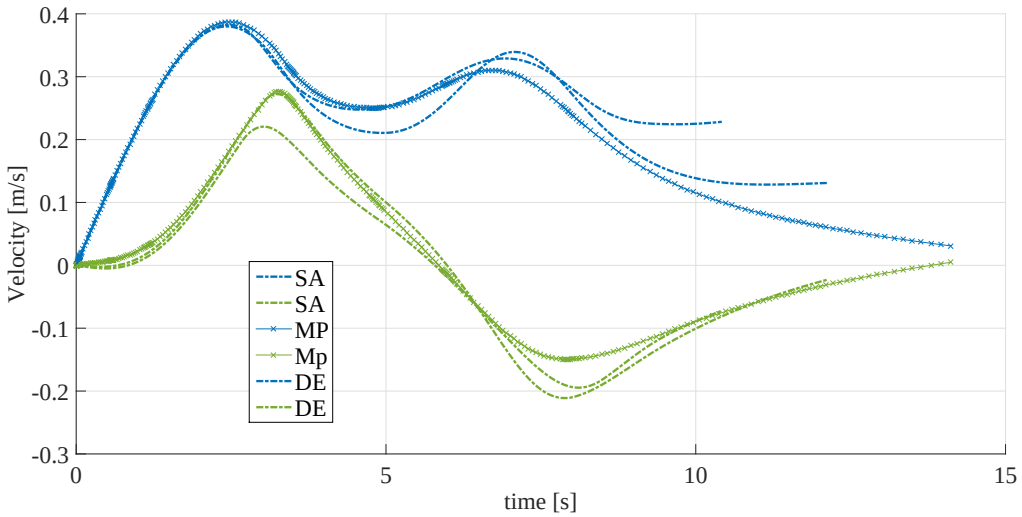
In this experiment, the control loop is closed by including feedback from the state variables (i.e. observation vector consisting of position, orientation, linear and angular velocities, and yaw angle, $\mathbf{o} = [x \; y \; \psi \; u \; v]$). In this case, the policy depends on the observation vector $\mathbf{o}$, as follows:

$$\Pi = \boldsymbol{\omega}^{\mathrm{T}} \boldsymbol{\varphi}(\mathbf{o}), \tag{3.17}$$

where $\boldsymbol{\omega}$ is the parameter vector and $\boldsymbol{\varphi}$ denotes the basis functions. Employing a 3^{rd} order Fourier basis to represent $\boldsymbol{\varphi}(\mathbf{o})$, the number of optimization parameters for each thruster is equal to 16. So, for navigation in 2D plane including two undamaged and one broken thrusters, the total number of optimization parameters equals to 32. As it can be seen in Figure 3-8a and 3-8b the acquired velocity profiles vary but finally converge towards $[u, v] = [0, 0]$ more smoothly, whereas, the acquired trajectories are similar.



(a) Trajectories in 2D plane.



(b) Velocity profiles along X and Y axes.

Figure 3-8: Acquired results for the experiments with state-dependent policy using Simulated Annealing, Modified Price and Differential Evolution algorithms. The discovered solutions by the employed algorithms are similar. For more information, see Section 3.4.4.

Once again, the optimization process was repeated 50 times for each optimization algorithm. The extracted statistical results are depicted in Figure 3-9. Also in this case, the DE algorithm shows a better performance in terms of the number of function evaluation and the best objective values.

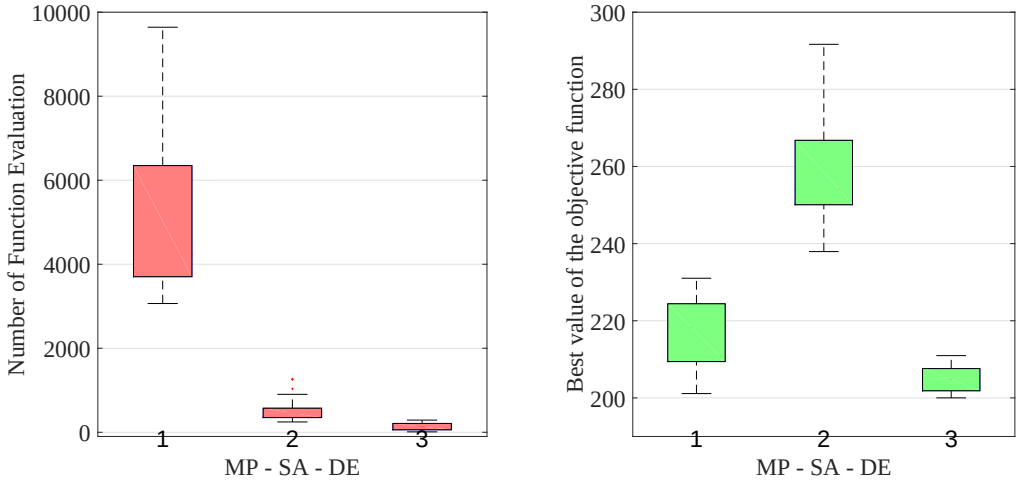


Figure 3-9: Acquired statistical results for the experiments with state-dependent policy representation over 50 runs using Simulated Annealing, Modified Price, and Differential Evolution algorithms. For more information, see Section 3.4.4.

3.4.5 Point-to-Point Navigation

In the previous experiments the complexity of the policy and the corresponding accuracy in reaching a target position and velocities were investigated. In this experiment, however, the feasibility of the approach to reach different target locations is examined using a time-dependent policy representation with 3^{rd} order Fourier basis functions. A grid of target points with coordinates in $[-10, 10]$ m at 1.0 m distance with respect to each other, and null target velocity is generated. Some of the generated trajectories are depicted in Figure 3-10. The AUV was able to reach each point at a distance of less then 0.2 m and stop there, since the target velocity was the null vector.

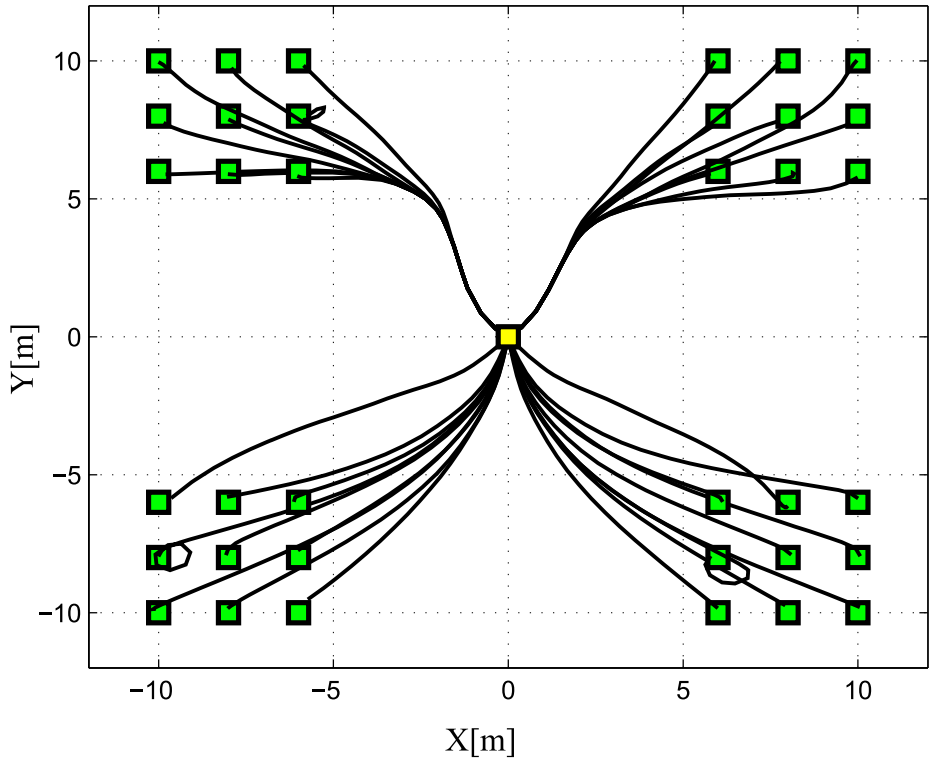


Figure 3-10: The result trajectories for the point-to-point navigation experiment using time-dependent policy representation. The yellow square is the initial position of the AUV, and the green squares show the desired final positions. For more information, see Section 3.4.5.

3.4.6 Navigation through Waypoints

In this experiment, the target is placed 50 m far from the current position of the AUV. Two desired path are generated to navigate the robot, one of which is a straight line and the other is an arc of circumference. Along each path a waypoint is generated every 5.0 m. The problem of reaching the next waypoint from the current state is iteratively posed with a target velocity pointing towards the subsequent waypoint and norm equal to 0.7, the highest linear velocity of Girona500. The acquired trajectories and the orientation of the AUV are shown in Figure 3-11. 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 real-world using the state-dependent policy representation and closing the loop. But in fact,

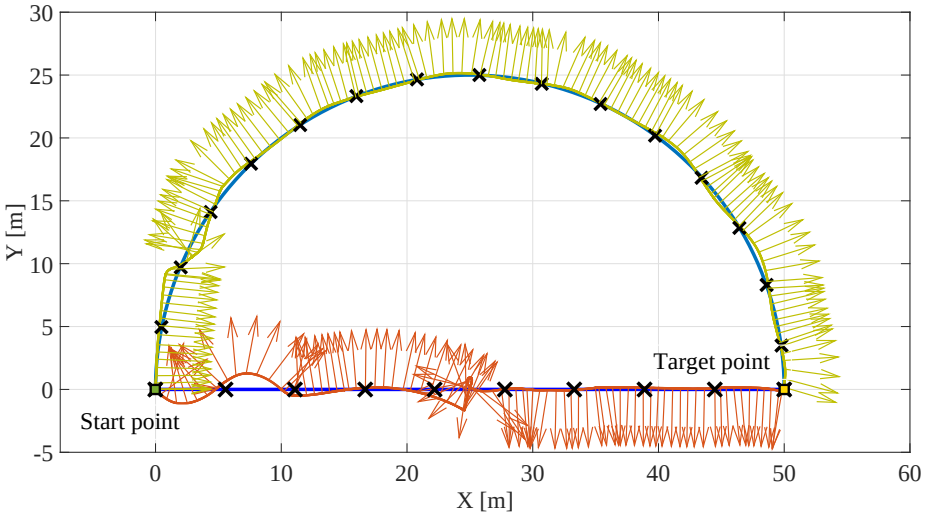


Figure 3-11: The desired path and acquired trajectories in the case that the AUV navigates through waypoints. The dark blue and the light blue profiles show the desired straight and curved paths respectively. The red and the light green profiles illustrate the acquired corresponding trajectories. The arrows show the orientation of the AUV at each point along the path. For more information, see Section 3.4.6.

either a larger test facility or an onshore tank is required to validate the experiment.

3.4.7 Covariance Analysis

The policies in the previous experiments were learned in a noiseless environment. Often, the real world has noises that have not been accounted for in the simulation. As a result a policy must be tested for robustness against such noises. The policy with the highest reward is not necessarily the most robust policy. This scenario is used as an example to show how covariance analysis can be used to select a robust policy. In order to perform covariance analysis, the robot picks the top policies from the noiseless simulations and runs them in a noisy simulation. In this experiment, the noise is introduced by adding a Gaussian noise to the input of the thrusters. Each policy is run 30 times and scored using a scoring metric. Firstly, a covariance matrix is calculated using the 30 samples with two variables: u and v position of the AUV. As denoted in (3.18), for each policy a single value, C_{policy} is extracted from the covariance matrix $\mathbf{Q} \in \mathbb{R}^{k \times k}$, by taking the product of the diagonal of the matrix. q_{ii} denotes the diagonal elements and k denotes number of variables which in this case is equal to two. Any other metric such as summation of the

diagonals can also work.

$$C_{\text{policy}} = \prod_{i=1}^{k} q_{ii}^{\text{policy}}. \tag{3.18}$$

The reward for the scoring metric is calculated using (3.19). It takes into account two factors: the time, t_{target}, it takes to reach the target, and the distance, d_{target}, from the target.

$$R_{\text{policy}} = \mathbb{E}[d_{\text{target}}] + \mathbb{E}[t_{\text{target}}], \tag{3.19}$$

To ensure that a policy that cannot reach, or takes a long time to reach the target is penalized, the formula in (3.20) is used to score the policies.

$$\text{score} = \beta \hat{R}_{\text{policy}} + (1-\beta)\hat{C}_{\text{policy}}, \tag{3.20}$$

where $\hat{R}_{\text{policy}}$ and $\hat{C}_{\text{policy}}$ are normalized reward and covariance of a policy respectively, and β determines the compromise between accuracy and robustness.

Figure 3-12 depicts the results of running the top five polices in a noisy environment. The final destination of the robot and the target position are illustrated. The area around the target ($\| \, . \, \| < 0.2$ m) is also marked. It is evident that the robustness to noise is not necessarily correlated to the ranking of a policy in the noiseless simulation. It can be noticed that the highest ranking policy is performing worse than lower ranking policies.

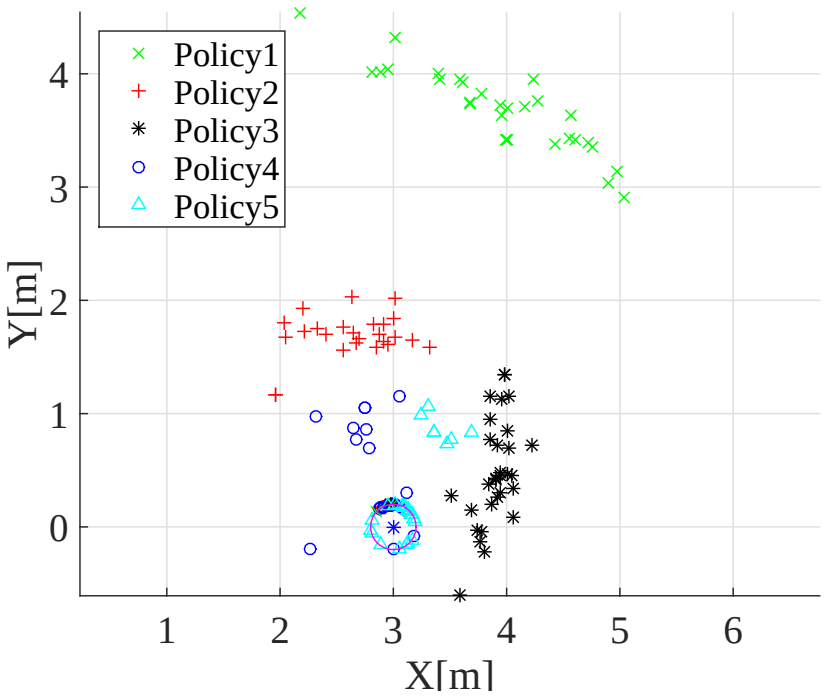
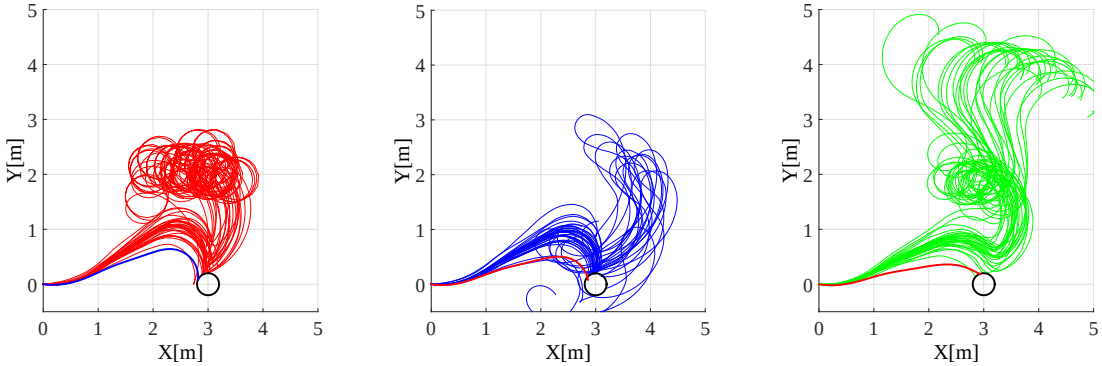


Figure 3-12: The positions reached by AUV during a noisy simulation. Policy 1 had the highest score, while Policy 5 was the fifth ranking policy. The colors correlate with policies in Figure 3-13. For more information, see Section 3.4.7.

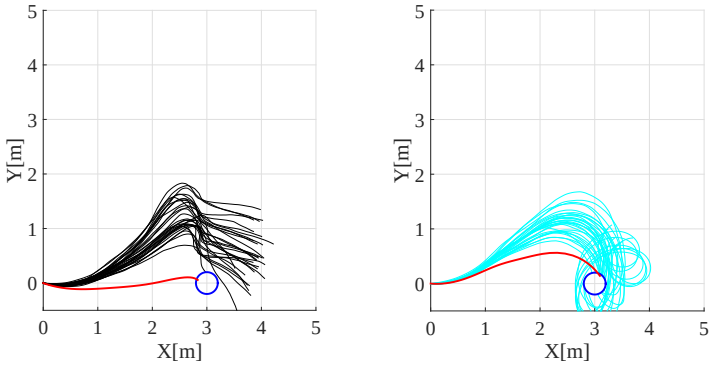
Figure 3-13 illustrates the path taken by each policy when executed in a noisy environment. The figures display paths for both noiseless and noisy environments. It can be seen that the top policy exhibits an erratic behavior when it fails, taking longer paths and performing loops. Other policies are also affected by noise. However, Policy 2 and Policy 5 are still able to reach the target position. While the two policies perform similarly when only target position is considered, Policy 5 performs better on the distance measure. The effect of β is also evaluated on the scoring. Table 3.1 shows the results of the scoring policy. Not surprisingly, for a β value of up to 0.5, the scoring selects Policy 5, which reflects visual analysis of the policies.

Table 3.1: Policy selection as the value of β is changed.

β	0.0	0.25	0.50	0.75	1
Policy	5	5	5	2	2



(a) Policy 1 (b) Policy 2 (c) Policy 3



(d) Policy 4 (e) Policy 5

Figure 3-13: Simulation results showing the effect of noise on selected five policies. The policies are ranked according their reward. The figures illustrate that policies with higher ranking are not necessarily robust to noise in the environment. For more information, see Section 3.4.7.

An unexpected behavior in the data was also noticed. Since a Gaussian noise is used, the paths in the noisy environment is expected to have a Gaussian distribution around the path in the noiseless environment. Further investigations revealed that this is due to the hardware limitations. Figure 3-14 shows the commands sent to the thrusters. The blue curve is the command for the surge thruster and the red curve is the command for the sway thruster. The thruster commands are produced by the learned policy. However, the thrusters only operate in the range $[-1, 1]$ and $[-0.5, 0.5]$ for surge and sway thrusters, respectively. As shown in Figure 3-14, in this experiment, the sway thruster sometimes reaches saturation. Therefore, in the robustness test simulations, adding noise to the already saturated thruster has no effect unless the addition of the noise moves the thruster

away from saturation. It also can be observed that the curve does not have a zero mean, hence, the saturation affects the negative thruster values more. Thereby, the AUV has a tendency to sway more towards the positive direction, which explains the bias noticed in the data.

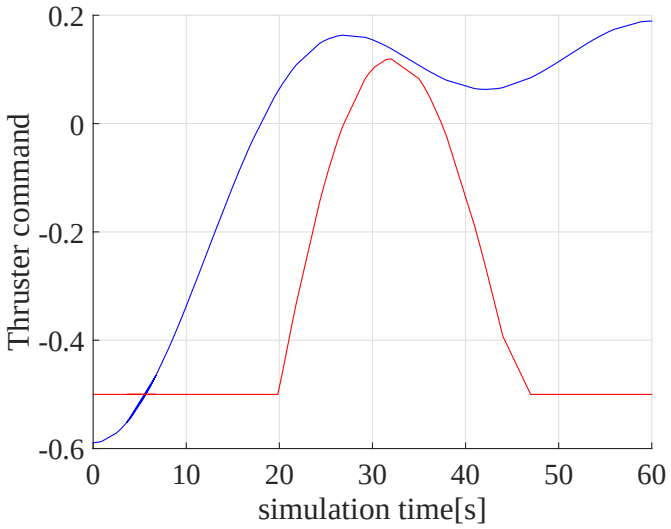


Figure 3-14: The thruster commands. The blue curve is the command for the surge thruster and the red curve is the command for the sway thruster. The figure shows that in this experiment, the sway thruster reaches saturation. For more information, see Section 3.4.7.

3.4.8 Learning Distributions

All the applied optimization algorithms generate the initial population randomly distributed over bounded regions. The bounded regions are extracted from the upper and lower bounds of the parameters. All the different solutions obtained from the 50 runs of the experiments in sections 3.4.3 and 3.4.4 are collected and a normal distribution is fitted to each parameter ω (i.e. a mean and a standard deviation for each parameter is calculated). The standard uniform distribution in the algorithms are then replaced by the estimated normal distribution for each parameter separately. Consequently, the algorithm generates the initial population from the learned distributions extracted from the previous experiments expecting better fitness or better objective values. Using this simple method, the best solutions in each experiment can be collected and added 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. As depicted in Figure 3-15, the optimization process shows better performance while using learned distribution, because it starts from better solutions. Statistically, 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 which is outside the scope of this thesis. Also, it is not been considered in the discussion about computational cost of the proposed method in the next section.

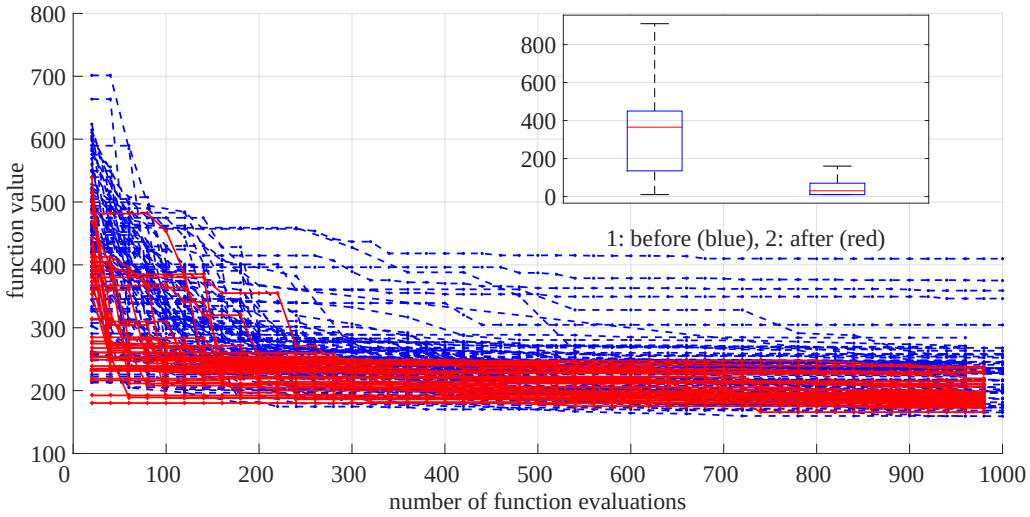


Figure 3-15: The comparison before and after learning the parameter distributions. The algorithm needs less number of function evaluations to reach the same results after learning the parameter distributions. Blue lines and red lines show the results before and after learning the distributions respectively. For more information, see Section 3.4.8.

3.4.9 Computational Cost

In order to check the feasibility of the presented framework, we need to assure that the policy optimization can be performed on-board in a short time. So, the quality of the solution (i.e. distance from the target) over time is computed for a waypoint 5.0 m away from the initial position of the AUV. The optimization process was repeated 20 times and

the average result is illustrated in Figure 3-16. The result shows that it took 12 seconds to find a solution able to take the AUV only 0.5 m from the target and the approach can find a satisfactory solution in less than 2 minutes. All experiments took place on a single thread on an Intel Core i3 CPU 2.30GHz.

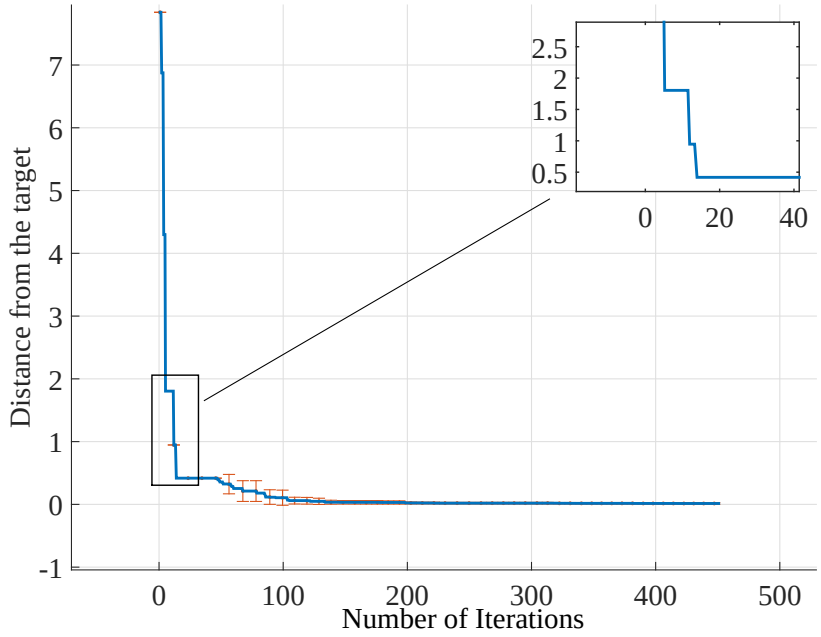


Figure 3-16: Computational cost of the learning method to discover a fault-tolerant policy. For more information, see Section 3.4.9.

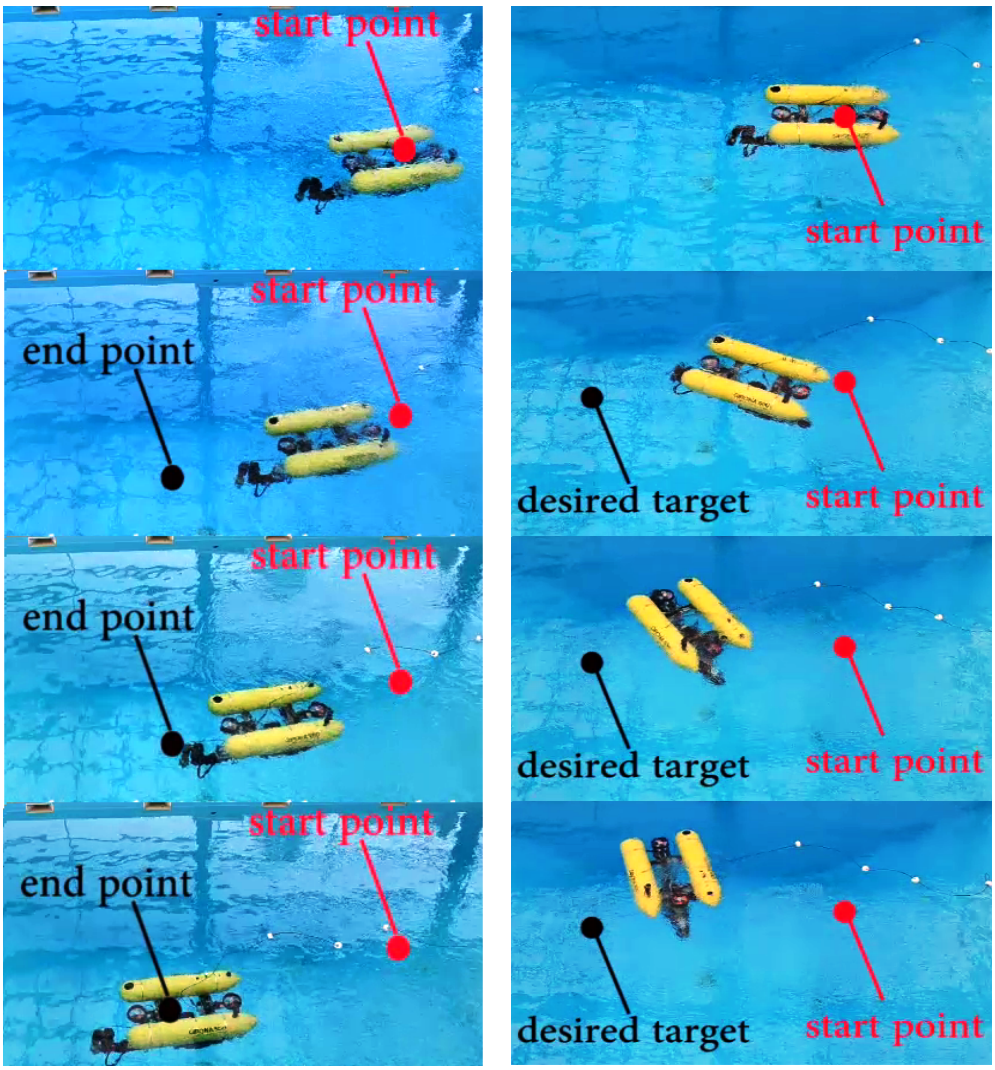
3.4.10 Real-world Experiment

After conducting several simulated experiments, in this experiment, the proposed framework is tested on Girona500 in real-world[5]. As it is depicted in Figure 3-18, firstly the robot is commanded to move 3.0 m along the surge direction while the original controller of the system is navigating the AUV; a number of snapshots in Figure 3-17a and the light blue trajectory in Figure 3-18 show the result. Naturally, the original control system is capable of navigating the robot towards the target.

Secondly, the right surge thruster is intentionally turned off and the same experiment is repeated. The behavior of the controller is plotted as the yellow trajectory in Figure 3-18

[5]A video of the real-world experiments reported in this section is available online at my personal website http://www.ahmadzadeh.info/videos.

and a number of snapshots in Figure 3-17b. As previously stated in Section 3.4.2, the result shows that the original controller of the system cannot recover the AUV from the failure, and the position error is increasing gradually.



(a) Normal condition.

(b) A failed thruster.

Figure 3-17: Two sets of snapshots illustrating the behavior of the AUV in normal condition and in the presence of thruster failure in one of the surge thrusters. The target is set 3.0 m in front of the AUV in surge direction. For more information, see Section 3.4.10.

Furthermore, to find an optimal policy for this thruster failure situation, an on-board simulation is performed using the state-dependent policy representation. The light green trajectory in Figure 3-18 shows the result obtained from the on-board simulation. Finally, the discovered optimal solution is applied to the real robot for navigating it towards the

target and the recorded trajectory is plotted as the black profile in Figure 3-18. In addition, a set of snapshots in Figure 3-19 illustrate the movement of the AUV while executing the discovered policy. The behavior of the robot in the real-world is very similar to the behavior of the simulator. Although the presented framework is using the model of the AUV, the main factors that make the real and simulated data slightly different can be enumerated as: (i) 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. (ii) unmodeled disturbances from the dynamic environment (e.g. currents, eddies and other sources of noise.)

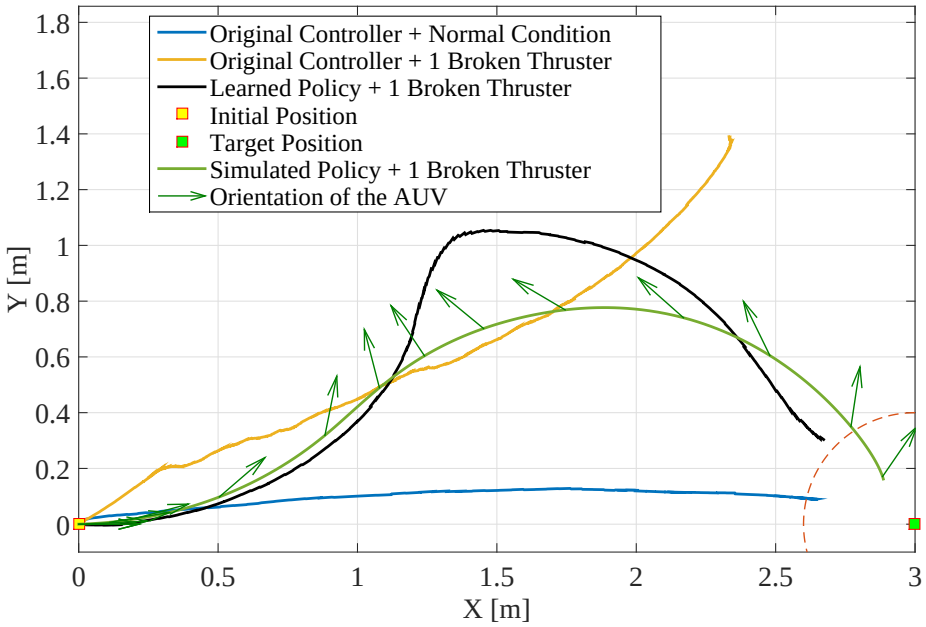


Figure 3-18: The trajectories recorded in different scenarios during the real-world experiments. For more information, see Section 3.4.10.

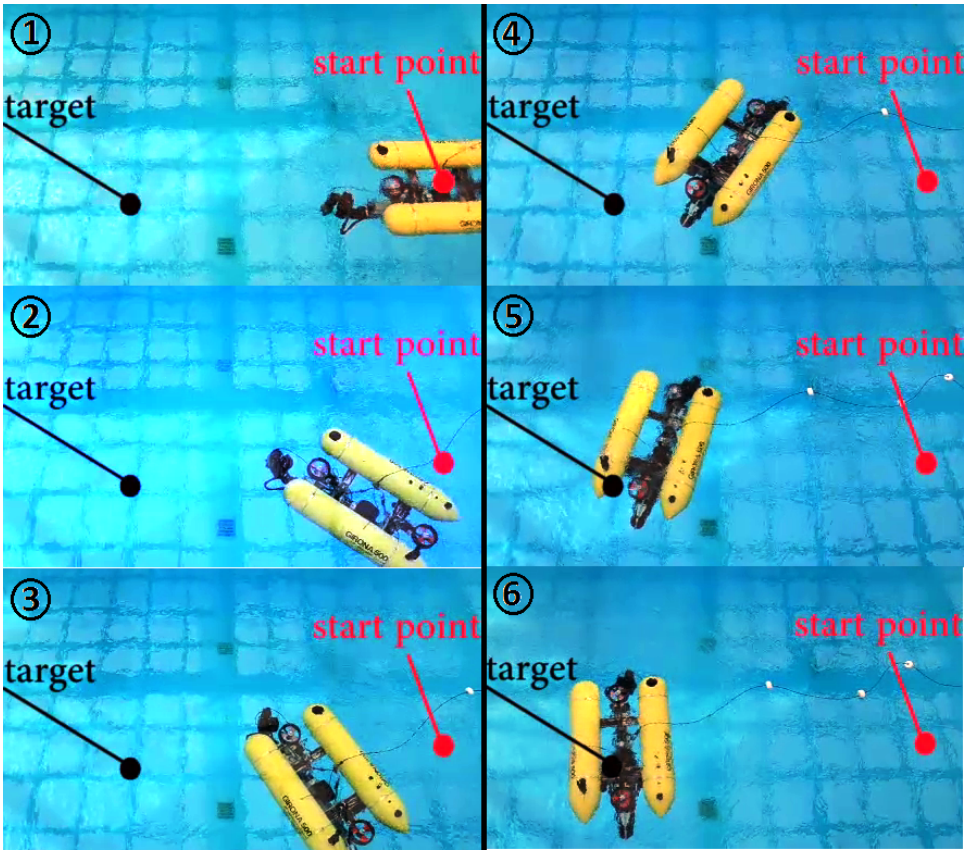


Figure 3-19: Snapshots illustrating the movement of the AUV while executing the discovered fault-tolerant policy. For more information, see Section 3.4.10.

3.5 Failure Recovery using Multi-Objective Reinforcement Learning

Up to this point, 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. The fact is that thrusters are barely totaly broken. One of the contributions of this section is taking advantage of the remaining functionality of a partially broken thruster. Therefore, the proposed method can deal with partially broken thrusters and use them to reach the desired goal. Furthermore, in the previous sections, a single-objective optimization approach was used and a scalarized objective function was employed, even though the defined objectives were conflicting. In other words, the approach can discover a policy that satisfies the

shortest path objective without satisfying the final velocity objective. In such case, although, the objectives are prioritized by the associated weights, the approach finally discovers a single optimal solution. Another issue is that the relationship between the weights and the discovered policy may be unpredictable and small changes in weights may produce large changes in the policy, or vice versa [102]. In this section, a Multi-Objective Reinforcement Learning (MORL) algorithm is employed that is capable of dealing with multiple conflicting objectives and discovering multiple optimal solutions. Each optimal solution can be used to generate a trajectory that is able to navigate the AUV towards a specified target.

3.5.1 Related Work on Multi-Objective Reinforcement Learning

Multi-objective reinforcement learning is the process of maximizing multiple rewards which can be complementary, conflicting, or independent. The existing methods can be categorized into two groups: single-policy and multiple-policy.

The first group aims at finding an optimal policy among many *Pareto* optimal policies that may exist. So the algorithm must be provided with some guidance as to which of these policies is to be preferred. To address this problem, Gabor *et al.* assume a fixed ordering between different rewards [103]. Mannor and Shimkin proposed a geometric approach that performs learning on multiple reward components [104]. The approach results in expected rewards lying in a particular region of reward space. An alternative approach to specifying preferences is linear scalarization [59, 105, 106], in which the objective vector is scalarized according to a weight vector. Linear scalarization has been used in Section 3.4 by utilizing single-objective reinforcement learning approach and dealing with more than one objective. Varying the weights allows the user to express the relative importance of the objectives. For instance, increasing an objective's weight will bias the learning towards that objective. One of the issues is that the relationship between the weights and the discovered policy may be unpredictable. Depending on the nature of the *Pareto* Front, small changes in weights may produce large changes in the learned policy, or vice versa [102].

The goal of the second group is to learn multiple policies that form an approximate

to the *Pareto* Front. Shelton calculates a gradient in the parameter-policy space for each objective and then these gradients are combined into a weighted gradient form [107]. By varying the weighting of the objective gradients a range of policies can be discovered. Barrett and Narayanan proposed a value-iteration convex hull algorithm that in parallel learns all deterministic policies which define the convex hull of the *Pareto* Front. It finally forms mixture policies which lie along the boundaries of this hull [108]. An alternative approach is employing evolutionary algorithms [109]. By evolving a population of solutions, multi-objective evolutionary algorithms are able to approximate the *Pareto* optimal set in a single run. Differential Evolution (DE) [96] was originally designed for scalar objective optimization. However because of its efficiency and simplicity, several multi-objective DE algorithms have been developed in recent years [110–112]. DE uses weighted difference between solutions to perturb the population and to create candidate solutions. The new trial solutions are partly from the candidate solutions and partly from the old population. DE has been used in the previous section that outperformed the others candidate algorithms. The Multi-Objective Differential Evolution (MODE) algorithm proposed in [113] is used in this section.

3.5.2 A More Generic Fault Detection Module

In a more realistic scenario, 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, a generic case is considered in which a thruster can be fully functional, partially broken or totally nonfunctional. Therefore, the assumption is made that the fault detection 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. The coefficient of functionality for a thruster can be estimated using different methods, for instance using energy measurement [114].

3.5.3 Vector Reward Function

In order to construct a framework including a multi-objective optimization algorithm and a set of conflicting objectives, a vector reward function is formed as follows:

$$R = \sum_{t=0}^{T} \mathbf{r}_t(\mathbf{o}_t)\Big|_{\Pi}, \tag{3.21}$$

where $\mathbf{r}_t$ is the immediate reward vector at time t that depends on the current observed states $\mathbf{o}_t$, 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 maximize the cumulative reward R over a horizon T. One of the advantages of vector reward function is that it can deal with independent, complementary, and conflicting objectives. Various definitions of the immediate vector reward are possible. The following definition of the immediate vector reward including three reward components r_t^1, r_t^2, and r_t^3 is used in this section and the experiments in Section 3.6:

$$\mathbf{r}_t = [r_t^1 \; r_t^2 \; r_t^3]^{\mathrm{T}} \tag{3.22}$$

$$r_t^1 = \frac{1}{\|P_t - P_d\| + \epsilon} \tag{3.23}$$

$$r_t^2 = \frac{1}{\|V_t - V_d\| + \epsilon} \tag{3.24}$$

$$r_t^3 = \frac{1}{|1 - \cos(\Delta\psi)| + \epsilon}, \tag{3.25}$$

where $\|\,.\,\|$ indicates 2-norm Euclidean distance, $\mathbf{p}_t$ and $\mathbf{p}_d$ are the current and the desired position vectors, $[x \;\; y]$, respectively. $\mathbf{v}_t$ and $\mathbf{v}_d$ are the current and the desired linear velocity vectors, $[u \;\; v]$, respectively. $\Delta\psi = (\psi_t - \psi_d)$ in which ψ_t and ψ_d are the current and the desired yaw angles of the AUV respectively. The first reward component, r_t^1, is defined to navigate the AUV towards the target. The second component, r_t^2, ensures the AUV reaches the target with minimum final velocity. Consider a scenario similar to the experiment in Section 3.4.6 that the path to reach a final target is segmented into multiple waypoints. In such case, it is important to pass through all the waypoints one by one

with the minimum final velocity and then execute the next discovered policy. Otherwise, the AUV needs to replan and discover new policies at each waypoint. Finally, the third component, r_t^3, keeps the orientation of the AUV always heading towards the target. The selected reward components are conflicting, for instance the AUV can reach the target with a different orientation far from the desired one.

3.5.4 Multi-Objective Optimization Problem

In this section, the problem of multi-objective optimization is formulated to provide a background for the next sections. A multi-objective optimization problem can be formulated as:

$$\begin{aligned} \textbf{Minimize} \quad & \mathbf{F}(\mathbf{x}) = [f_1(\mathbf{x}), ..., f_m(\mathbf{x})]^T \\ s.t. \quad & \mathbf{x} \in \Omega, \end{aligned} \tag{3.26}$$

where Ω is the decision space and $\mathbf{x} = [x_1, x_2, ..., x_n]^T$ is a decision vector, and $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, ..., k$ are the objective functions.

Single-objective optimization problems may consist of a number of objective functions, as far as the objectives are not conflicting with each other. In this case a single solution exists, which can optimize all the objectives. On the other hand, if the objectives conflict, improvement of one may lead to deterioration of another. The best solutions in such a case are called *Pareto* optimal solutions. The following definitions are used in the concept of *Pareto* optimality:

Definition 1. A vector $\mathbf{u} = [u_1, ..., u_m]^T$ is said to dominate another vector $\mathbf{v} = [v_1, ..., v_m]^T$, denoted as $\mathbf{u} \prec \mathbf{v}$, iff $\forall i \in \{1, ..., m\}$, $u_i \leq v_i$ and $\mathbf{u} \neq \mathbf{v}$.

Definition 2. A feasible solution $\mathbf{x}^* \in \Omega$ is called a *Pareto* optimal solution, iff $\nexists \mathbf{y} \in \Omega$ such that $\mathbf{F}(\mathbf{y}) \prec \mathbf{F}(\mathbf{x}^*)$.

Definition 3. *Pareto* Set (PS) is the set of all the *Pareto* optimal solutions and can be denoted as:

$$\text{PS} = \{\mathbf{x} \in \Omega | \nexists \mathbf{y} \in \Omega, \mathbf{F}(\mathbf{y}) \prec \mathbf{F}(\mathbf{x})\}, \tag{3.27}$$

Pareto Front (PF) is the image of the PS in the objective space that can be denoted as:

$$\mathrm{PF} = \{\mathbf{F}(\mathbf{x}) | \mathbf{x} \in \mathrm{PS}\}. \tag{3.28}$$

3.5.5 Multi-Objective Optimization Algorithms

Among several existing optimization algorithms, evolutionary algorithms are considered as a powerful alternative. These algorithms are very effective in solving complex search problems, including single-objective and multi-objective optimization problems [109]. Dealing with a group of candidate solutions, makes them effective to find a group of optimal solutions. Because of its simplicity and high efficiency, Differential Evolution [96] is considered as one of the most popular evolutionary algorithms over continuous domains. The backbone of the algorithm is based on weighted difference between solutions to perturb the population and to create candidate solutions. Previously in Section 3.4, the performance of single-objective DE was compared with two other population based algorithms, namely, Simulated Annealing and Modified Price. The obtained results suggest that DE outperforms the other candidates. Consequently, to solve the described multi-objective problem in this chapter, a multi-objective differential evolution algorithm is employed.

In this section, firstly the standard differential evolution algorithm is briefly explained. Subsequently, a multi-objective extension of the differential evolution is discussed in more details.

Differential Evolution

DE is a population-based stochastic method for global optimization and its recombination and mutation operators are the variation operators used to generate new solutions. Unlike, Genetic Algorithm (GA) and several ES approaches, solutions in DE are encoded with real values. Moreover, DE does not use a fixed distribution as the Gaussian distribution adopted in ES approaches; instead, the current distribution of the solutions in the search space determines the search direction and even the stepsize for each individual.

DE utilizes N_P D-dimensional parameter vector

$$\mathbf{x}_{i,G} = [x_{1,i,G}, x_{2,i,G}, \ldots, x_{D,i,G}]^{\mathrm{T}}, \quad i = \{1,2,...,N_P\}, \tag{3.29}$$

as a population for each generation G, where N_p denotes the number of population, D is the dimension of the optimization parameter vector, $x_{j,i,G}$ is j^{th} parameter in i^{th} population in generation G. In order to cover the entire parameter space, the initial population is sampled using a uniform probability distribution. Thereby, for each parameter a minimum and a maximum bound are specified by the user as follows:

$$\mathbf{x}_{\min} = [x_{1,\min}, x_{2,\min}, \ldots, x_{d,\min}]^{\mathrm{T}} \tag{3.30}$$

$$\mathbf{x}_{\max} = [x_{1,\max}, x_{2,\max}, \ldots, x_{d,\max}]^{\mathrm{T}} \tag{3.31}$$

$$\mathbf{x}_{i,G} \in [\mathbf{x}_{\min}, \mathbf{x}_{\max}]. \tag{3.32}$$

The mutation operator works based on differences between pairs of solutions with the aim of finding a search direction using the distribution of the solutions in the current population. DE generates new parameter vectors by adding the weighted difference between two population vectors to a third vector. The mutant vector is generated as:

$$\mathbf{v}_{i,G+1} = \mathbf{x}_{\rho_1,G} + F(\mathbf{x}_{\rho_2,G} - \mathbf{x}_{\rho_3,G}), \tag{3.33}$$

where $\rho_1, \rho_2, \rho_3 \in \{1,2,...,N_p\}$ are random indices, and $F \in (0.4,1]$ is a real and constant factor which controls the amplification of the differential variation. In the next phase, the crossover operation is introduced to increase the diversity of the perturbed parameter vector. The trial vector

$$\mathbf{u}_{i,G+1} = (u_{1,i,G+1}, u_{2,i,G+1}, ..., u_{D,i,G+1}), \tag{3.34}$$

is formed according to

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & \text{if } \texttt{rand}(j) \leq C_r \text{ or } j = \texttt{rnbr}(i) \\ x_{j,i,G} & \text{otherwise} \end{cases} \tag{3.35}$$

where, $\texttt{rand}(j)$ is the j^{th} evaluation of a uniform random number. $C_r \in [0,1]$ is the crossover constant. $\texttt{rnbr}(i)$ is a randomly chosen index which ensures that $\mathbf{u}_{i,G+1}$ gets at least one parameter from $\mathbf{v}_{i,G+1}$. The selection operator is used to decide whether or not the trial vector should become a member of the next generation. By comparing the trial vector to the target vector, the selection operator selects the vector with smaller cost.

In order to classify the different variants of DE the notation `DE/a/b/c` is proposed by Storn and Price [96]. `a` specifies the vector to be mutated. Two examples are `rand` and `best`. `b` is the number of difference vectors used in the strategy. `c` denotes the crossover scheme (e.g. `bin`). The explained DE strategy can be written as: `DE/rand/1/bin`. For more information about the DE algorithm, see Appendix A.

Multi-Objective Differential Evolution

Because of its efficiency for solving problems, several multi-objective DE algorithms have been developed in recent years [110–112]. In this chapter, the Multi-Objective Differential Evolution (MODE) algorithm proposed in [113] is utilized. This algorithm is inspired from Elitist non-dominated Sorting Genetic Algorithm (NSGA-II). In a multi-objective domain, the goal is to identify the *Pareto* Set (PS). In MODE, a population of size N_p is generated randomly and the fitness functions are evaluated. The population is then sorted based on dominance concept. In the next step, the operations of DE are carried out over the individuals of the population. Then the fitness of the trial vectors are evaluated. Unlike DE, in MODE the trial vectors are not compared with the corresponding parent vectors. Instead, both the parent vectors and the trial vectors are combined to form a global population of size, $2N_p$. The global population is then ranked by the distance calculation. The best N_p individuals are selected based on their ranking and distance and act as the parent vector for the next generation. Algorithm 1 shows a simplified pseudo code of the MODE

algorithm.

Input : $\{D, N_p, G_{\max}, F, C_r\}$

$P_{\text{parent}} \leftarrow$ Initialize random population
$J_{\text{parent}} = \text{eval}(P_{\text{parent}})$
for $i = 1$ *to* $G_{\max}$ **do**
    for $j = 1$ *to* N_P **do**
        $P_{\text{mutant}} = \text{Mutation}\,(P_{\text{parent}})$
        $P_{\text{child}} = \text{Crossover}\,(P_{\text{parent}}, P_{\text{mutant}})$
    end
    $J_{\text{child}} = \text{eval}(P_{\text{child}})$
    for $j = 1$ *to* N_P **do**
        $[P_{\text{parent}}, J_{\text{parent}}] = \text{Selection}\,(J_{\text{parent}}, J_{\text{child}})$
    end
    $\text{PF} = J_{\text{parent}}$
    $\text{PS} = P_{\text{parent}}$
end
$[\text{PF}_{\text{out}}, \text{PS}_{\text{out}}] = \text{DominanceFilter}\,(\text{PF}, \text{PS})$

Algorithm 1: Multi-Objective Differential Evolution (MODE)

3.6 Experiments with Multi-Objective Reinforcement Learning

Similar to the previous section, the extended framework includes the dynamic model of the AUV from Section 3.3.1, the policy representation from Section 3.4.4, a vector reward function from Section 3.5.3 and a multi-objective optimization algorithm as described in Section 3.5.5. The dynamics model of the system is implemented as an on-board module for performing the simulations on the robot. The policy is represented with a linear function approximator using Fourier basis functions. When a thruster is deemed faulty, the fault-detection module sends a signal including a vector indicating the coefficient

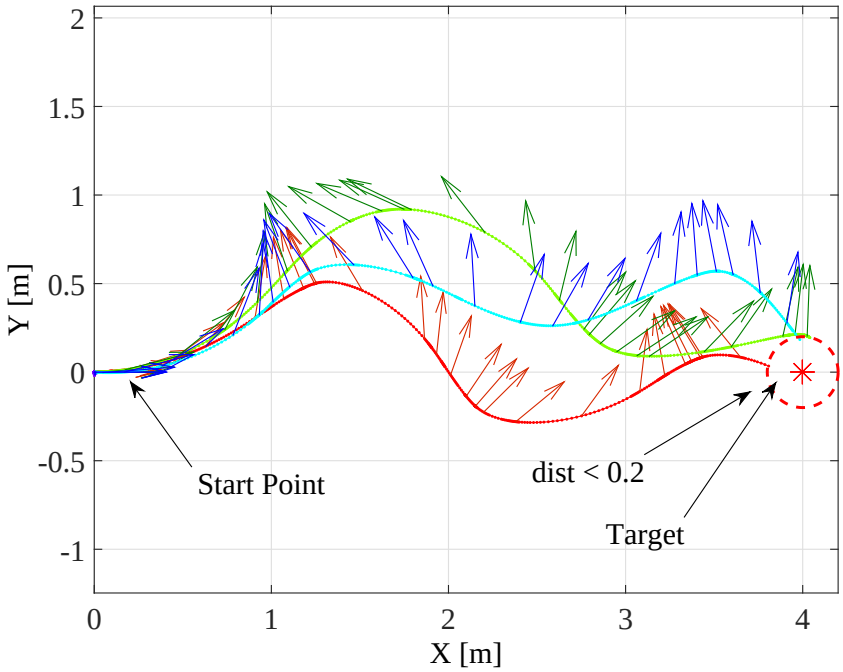
of functionality for each thruster to a higher planning layer. The higher layer decides whether or not switch the failure recovery layer on. A vector reward function R consisting of multiple objectives is created. Finally, the optimization algorithm discovers a set of optimal policies from the simulation to recover the AUV. To validate the reliability and performance of the proposed framework, in this section, a set of experiments is conducted in simulation.

3.6.1 Setup

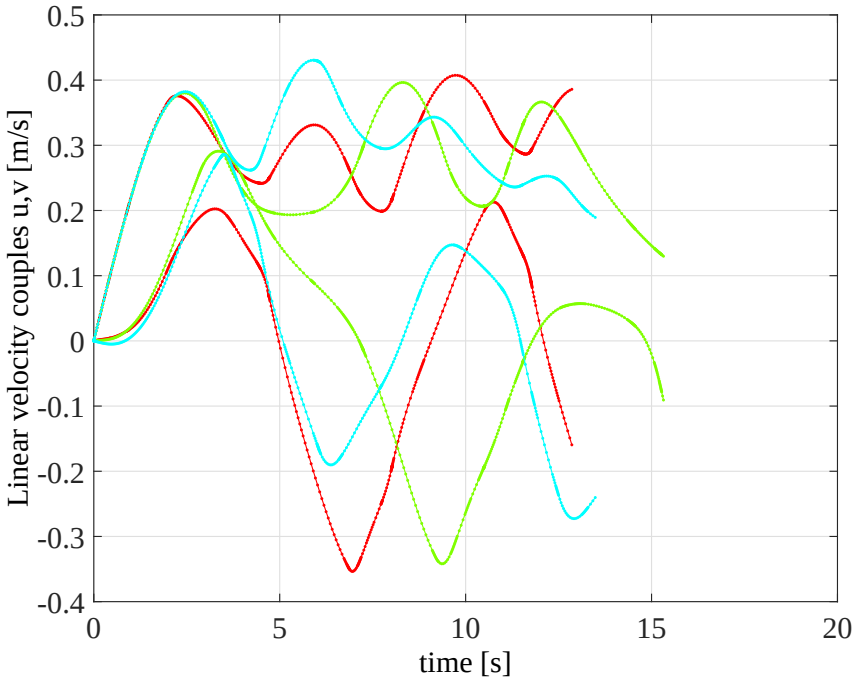
The conducted experiments in this section are based on 6 DoF dynamics model of Girona500 formulated in (3.4) which its hydrodynamics parameters have been identified in [92]. As mentioned before, the heave thrusters in Girona500 control the vehicle in heave direction and also compensate the buoyancy force and keep the vehicle submerged. Consequently, the experiments are designed so that the thruster failure occurs in the horizontal plane, while the heave movements of the AUV are always controlled by its original controller. In all the experiments, the right surge thruster is deemed faulty. As described in Section 3.5.2, the fault detection module not only detects the faulty thruster but also estimates its coefficient of functionality ($\alpha \in [0,1]$). So, in different experiments α_2 indicates the measure of the functionality of the second surge thruster. However, the other two thrusters are considered fully functional ($\alpha_1 = \alpha_3 = 1$). Unlike previous experiments in Section 3.4, the proposed approach benefits from the remaining functionality of the faulty thruster together with the other healthy thrusters. For each experiment a final time, T, is specified for each episode (e.g. $T = 60$ s). The final time is selected somehow that the target is reachable in T seconds. The vector reward function is designed such that when the AUV reaches an area close enough to the desired position, $\|\mathbf{p}_t - \mathbf{p}_d\| < 0.2$ m, the current episode is terminated. In all the experiments, the policy depends on five state variables of the system, $\mathbf{o} = [x\ y\ \psi\ u\ v]$. Employing a 3$^{\text{rd}}$ order Fourier basis to represent the features of the policy, the number of optimization parameters is equal to 16 for each thruster. So, for navigating in 2D plane including three thrusters (totally and partially functional), the total number of the optimization parameters is 48.

3.6.2 A Fully Broken Thruster

In the first multi-objective experiment, a case is considered in which the right surge thruster is fully broken ($\alpha_2 = 0$). In such situation the AUV becomes under-actuated and any attempt to reallocate the actuator configuration matrix would be ineffective. This scenario is similar to the conducted experiments in Section 3.4 using scalar rewards. The goal of this experiment is to analyze the capability of the multi-objective approach for dealing with the under-actuated AUV. In the current scenario, the target is located 4.0 m in front of the AUV along the surge direction and the algorithm tries to find a set of policies to satisfy the multiple objectives defined in section 3.5.3. The set of optimal trajectories acquired by employing the discovered *Pareto* optimal policies are depicted in Figure 3-20a. The related velocities are illustrated in Figure 3-20b. The effect of the multiple conflicting objectives can be seen in the figures. For instance, in Figure 3-20a the final heading error of the AUV with respect to the target in the light blue trajectory is less than the other trajectories. On the other hand, in Figure 3-20b the light green profile shows a minimum final velocity.



(a) Optimal trajectories.



(b) Velocity profiles.

Figure 3-20: Three optimal trajectories and corresponding velocity profiles acquired by employing the discovered *Pareto* optimal solutions for a fully broken thruster. The arrows show the orientation of the AUV. For each trajectory two linear velocity profiles exist (u in x direction and v in y direction). For more information, see Section 3.6.2.

3.6.3 A Partially Broken Thruster

In the second multi-objective experiment, a case is considered in which the right surge thruster is partially broken ($\alpha_2 \neq 0$). In such situation the AUV is considered as redundant, but the dynamic behavior of the system is different. Similar to the previous experiment, the target is located 4 m in front of the AUV and the algorithm tries to find a set of optimal policies to satisfy the multiple-objectives defined in section 3.5.3. To check the performance of the approach for various values of α_2 (the coefficient of functionality for the faulty thruster), the experiment was repeated 11 times and each time α_2 is increased by 0.1 starting from 0 to 1. The optimal trajectories generated by the discovered *Pareto* optimal policies in each case are depicted in Figure 3-21. Since in each experiment, the maximum number of episodes is fixed to 50, in some cases the algorithm discovered more *Pareto* optimal policies, while in some others it found less solutions. In addition, the maximum number of function evaluations in each case is set to 1000.

In order to investigate the optimization process in more details, one of the experiments ($\alpha_2 = 0.4$) is repeated with $N_p = 100, C_r = 0.9, F = 0.85, G_{max} = 50$. The results are reported in Figure 3-22. Three optimal solutions are reported in separate plots (3-22a, 3-22b and 3-22c). As it can be seen, Figure 3-22a illustrates a trajectory that satisfies the shortest path objective. The trajectory in Figure 3-22b satisfies the minimum final velocity and the third trajectory in Figure 3-22c keeps the heading of the AUV towards the target, to minimize the heading error. In Figure 3-23, the reward-space *Pareto* Front is plotted over the first and third objectives, and the convergence of the populations in each generation is depicted by a color gradient.

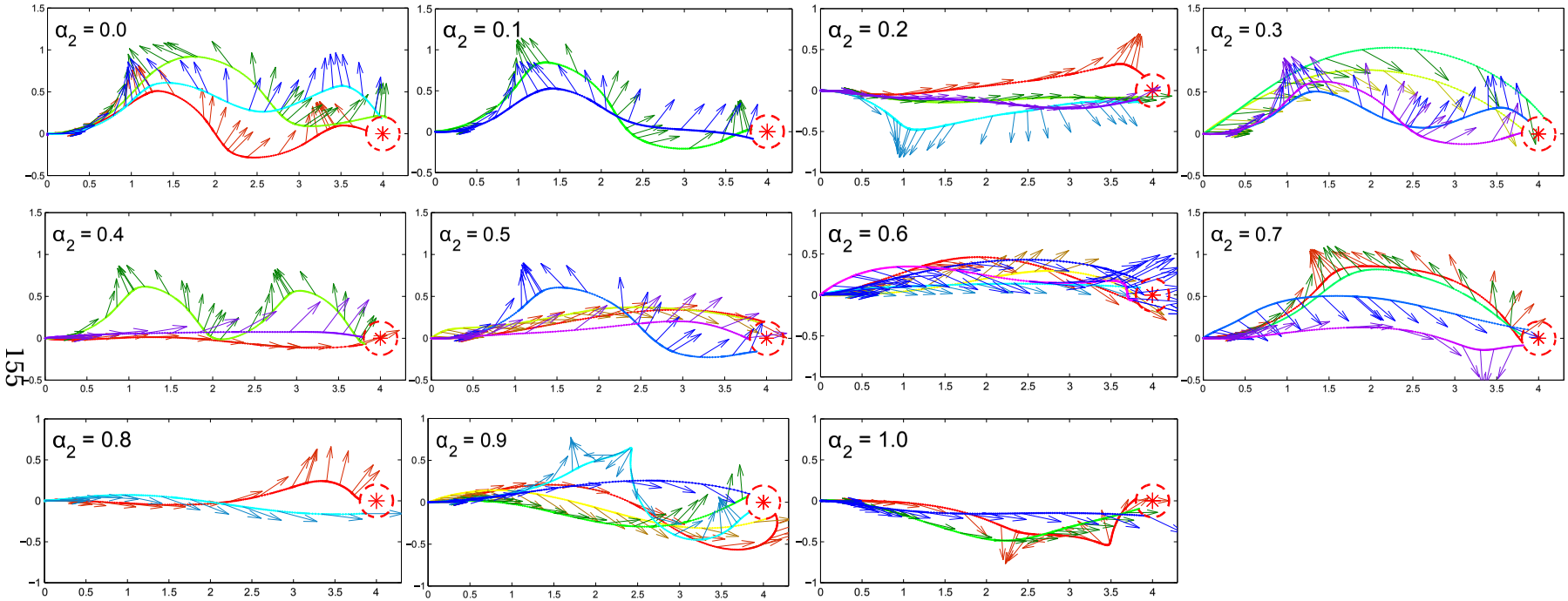
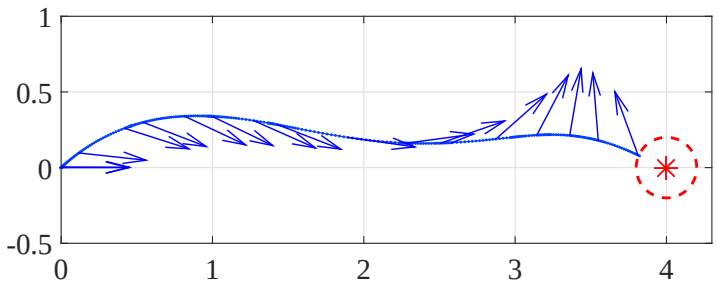
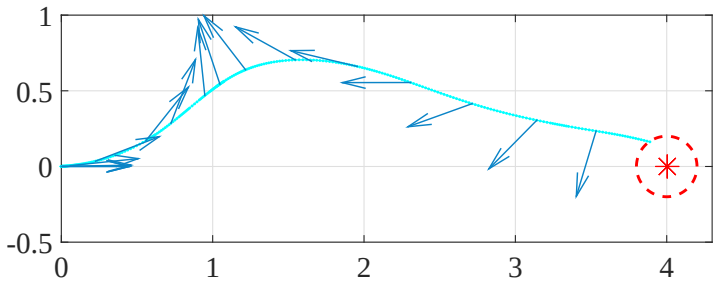


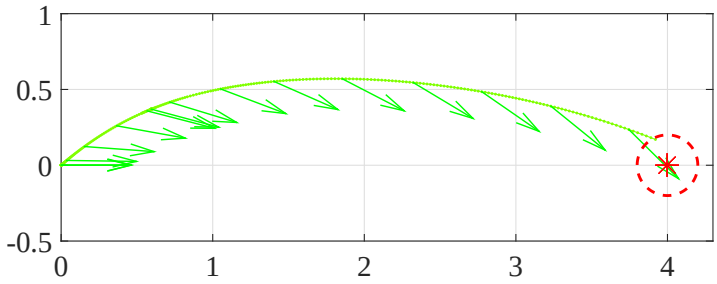
Figure 3-21: Acquired results for the 2nd multi-objective experiment with various coefficient of functionality for the right surge thruster which is a dimensionless quantity between $[0,1]$. In all sub-figures the x and y axes show the surge and sway directions respectively in the horizontal plane of AUV and the dimensions are in meters. For more information, see Section 3.6.3.



(a) The solution satisfying the 1^{st} objective, shortest path.



(b) The solution satisfying the 2^{nd} objective, minimum final velocity.



(c) The solution satisfying the 3^{rd} objective, minimum heading error.

Figure 3-22: Acquired results for the 2^{nd} multi-objective experiment with $\alpha_2 = 0.4$. In sub-figures 3-22a, 3-22b, 3-22c the x and y axes show the surge and sway respectively and the dimensions are in meters. For more information, see Section 3-22.

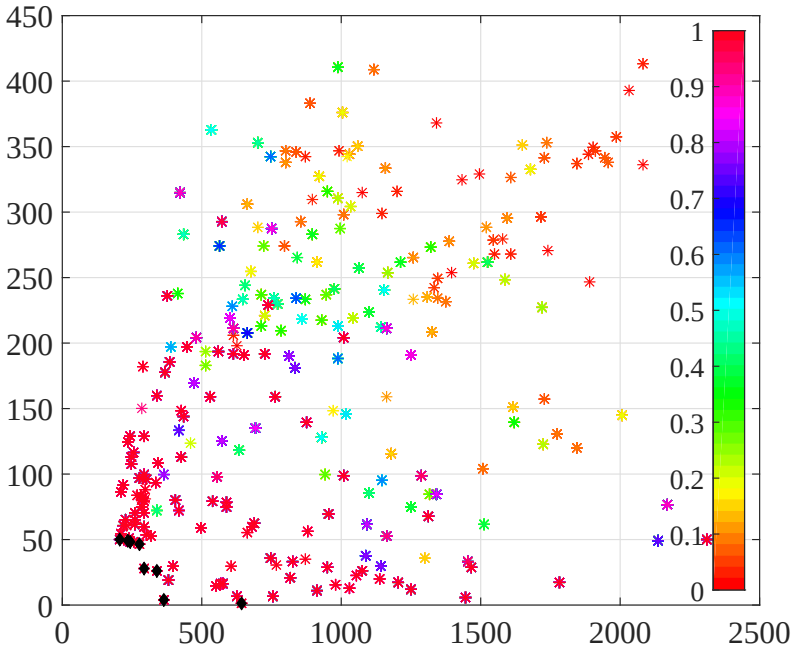


Figure 3-23: The reward-space *Pareto* Front is plotted over two objectives, in which the dominant solutions are marked as black. The movement of populations towards minimum rewards is shown with a color gradient. The axis are dimensionless. For more information, see Section 3-22.

3.6.4 Improving the Performance

In order to apply the proposed multi-objective approach in real-world experiments more efficiently, the performance of the approach can be improved. The idea behind that is to benefit from the previous experience for increasing the performance of the learning approach by decreasing the computational cost. Subsequently, all the optimal policies discovered for different values of α_2 which are extracted in the conducted experiment in Section 3.6.3, are stored in a lookup table. The extracted data is also depicted in Figure 3-24.

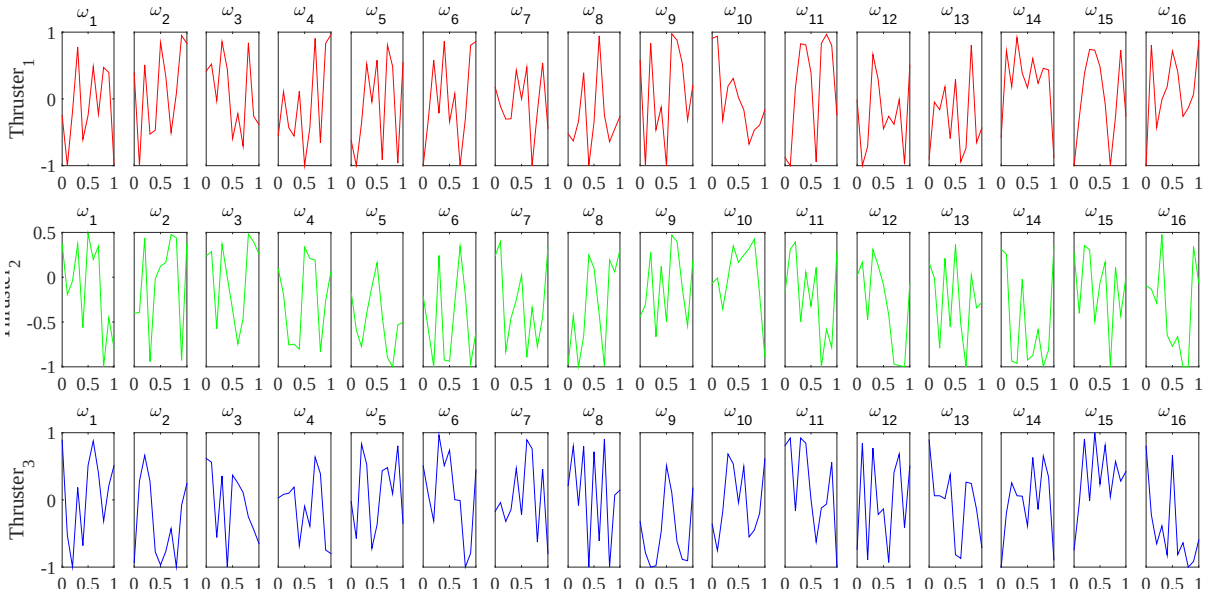


Figure 3-24: The discovered policy parameters from the 2nd experiment in Section 3.6.3. Each row is related to a thruster. In each row 16 parameters of the policy representation, ω, are depicted. Each parameter is re-scaled to range $[-1,1]$ and is plotted versus α. Both axes illustrate dimensionless quantities. For more information, see Section 3.6.4.

In Figure 3-24, each row is related to one of the thrusters and in each row 16 policy parameters, ω, are plotted. Each ω is plotted versus α ($\alpha \in [0, 0.1, \ldots, 1]$). Since the behavior of the parameters seems to be nonlinear, model fitting techniques are ineffective. For instance, using regression techniques to find a model for each parameter would be inadequate. Instead, the previous experience is stored as a lookup table for initializing the first generation of parents in the optimization algorithm. The effect of using previous knowledge is further investigated by employing the nearest existing solution as initial population.

A new experiment is performed where α_2 is not exactly one of the previously experienced values, $\alpha_2 = 0.45$. The experiment is conducted twice. First the algorithm is initialized with a random population to discover an optimal control policy similar to previous experiments. Then the same simulation is repeated, this time starting from the existing optimal solution for $\alpha_2 = 0.4$. In other words, the algorithm initializes the first generation of parents with the nearest existing solution from the lookup table. All the other assumptions are similar to the previous experiments. To be consistent, The whole experiment is repeated 20 times. The statistical results are depicted in Figure 3-25. In the

left, the number of function evaluations and in the right, the required time to reach the first optimal solution are compared between two cases. All experiments took place on a single thread on an Intel Core i3 CPU 2.30GHz. The result suggests that starting from a neighbour solution can increase the performance of the approach. It is worth noting that, in a case that the selected initial solution is not improving the performance of the approach due to the nonlinear behavior of the parameters with respect to α, it would act such as a neutral initial solution. The algorithm naturally replaces this solution with a better individual in the next step. This fact suggests that using such lookup table does not increase the computational cost and it is highly likely that the selected initial solution from the table augments the efficiency when running the approach on-line.

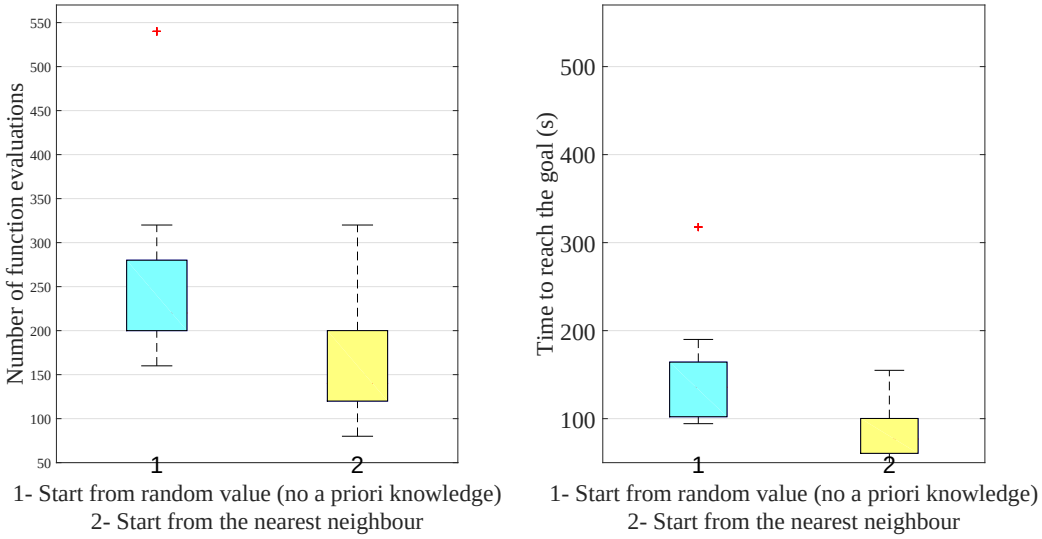


Figure 3-25: The box-plots show the effect of employing previously experienced knowledge in the performance of the approach. The first experiment starts from random values, whereas the second experiment uses the nearest neighbor solution. The left plot shows the number of function evaluations and the right plot depicts the time needed to reach the goal. Both experiments were averaged over 20 runs. For more information, see Section 3.6.4.

3.7 Chapter Summary

In this chapter a learning framework has been proposed that enables a robot to learn a reactive behavior for dealing with internal failures. The proposed framework focuses

on elevating fault-tolerance in underwater robotic domain to increase the reliability and autonomy of AUVs. Using a model-based direct policy search, new control policies are discovered to overcome thruster failures as they happen. The policies are learned on an on-board simulated model of the AUV. The model is adapted according to the new condition when a fault is detected and isolated. Since the learning framework generates an optimal trajectory, the learned fault-tolerant policy is able to navigate the AUV towards a specified target while satisfying desired objectives. In most other methods, on the other hand, trajectory generation is ignored in the fault-tolerant control problem. Finally, the learned policy is executed on the real robot in a closed-loop using the state feedback of the AUV.

One of the advantages of the proposed framework is that it is applicable in both cases when the AUV becomes under-actuated in the presence of a fault or remains over-actuated. Unlike most existing methods which disregard the faulty thruster, the proposed framework can also deal with partially broken thrusters to increase the autonomy of the AUV. Furthermore, 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.

To deal with multiple objectives which can be complementary, conflicting, or independent, two variants of the proposed frameworks have been developed throughout this chapter. The first variant utilizes linear scalarization technique which is suitable for independent and complementary objectives. However, it has been used for conflicting objectives by employing prioritization technique. This variant includes a policy representation, a reward function, a single-objective reinforcement learning algorithm and a dynamics model of the AUV. It discovers a single fault-tolerant control policy that can recover the AUV from thruster failures. The second variant, on the other hand, utilizes a multi-objective algorithm that can deal with multiple conflicting objectives by discovering multiple optimal solutions. This variant includes a dynamic model of the AUV, a policy representation, a vector reward function, and a multi-objective reinforcement learning algorithm. It discovers multiple fault-tolerant policies each of which satisfies the

objectives differently. In linear scalarization technique the objective vector is scalarized according to a weight vector. The weight vector specifies the relative importance of the objectives. Varying weight of an objective will bias the learning towards that objective. One of the disadvantages of scalarization is that the relationship between the policy and the weights may be unpredictable and small changes in weights may produce large changes in the policy. On the other hand, multi-objective optimization techniques with vector objectives address this issue by investigating the nature of *Pareto* Front for the specified scenario. However, after all they require a decision making algorithm or a human expert to select an optimal solution.

The feasibility and efficiency of the proposed learning framework have been validated through simulated and real-world experiments.

"No army can withstand the strength of an idea whose time has come."

— Victor Hugo

4

Visuospatial Skill Learning

4.1 Motivation

DURING the past two decades several robot skill learning approaches based on human demonstrations have been proposed [5, 115–118]. Many of them address motor skill learning in which new motor skills are transferred to the robot using policy derivation techniques such as mapping function [119], system model [120], etc. Motor skill learning approaches can be categorized in two main groups: trajectory-based and goal-based. The former group put the focus on recording and regenerating trajectories [117] or intermittently forces [121] to emulate the demonstrated skill. For instance, in [122] a humanoid robot learns to play air hockey by learning primitives, or when combined with reinforcement learning in [123] a robot learns to flip a pancake by observing demonstrations.

In many cases, however, it is not the trajectory that is of central importance, but the goal of the task (e.g. solving a jigsaw puzzle [124]). Learning every single trajectory in such tasks actually increases the complexity of the learning process unnecessarily [125]. To address this drawback, several goal-based approaches have been proposed [126–128]. There is a large body of literature on grammars from the linguistic and computer science communities, with a number of applications related to robotics [125, 127]. Furthermore, a number of symbolic learning approaches exist that focus on goal configuration rather than action execution [128]. However, in order to ground the symbols, they comprise many steps inherently, namely segmentation, clustering, object recognition, structure recognition, symbol generation, syntactic task modeling, motion grammar, rule generation, etc. Another drawback of such approaches is that they require a significant amount of *a priori* knowledge to be manually engineered into the system [125, 127]. In addition, most above-mentioned approaches assume the availability of the information on the internal state of a human tutor such as joint angles, while humans usually cannot directly access to imitate the observed behavior.

An alternative to motor skill approaches are visual learning approaches [6, 8]. These approaches are based on observing the human demonstration and using human-like *visuospatial skills* to replicate the task. *Visuospatial skill* is the capability to visually perceive the spatial relationship between objects.

This chapter[1] proposes a novel visuospatial skill learning approach for interactive robot learning tasks. Unlike the motor skill learning approaches, the proposed approach utilizes visual perception as the main source of information for learning new skills from demonstrations. The proposed Visuospatial Skill Learning (VSL) approach, which is a significant contribution of this thesis, uses a simple algorithm and minimum *a priori* knowledge to learn a sequence of operations from a single demonstration. In contrast to many previous approaches, VSL leverages simplicity, efficiency , and user-friendly human-robot interaction. Rather than relying on complicated models of human actions, labeled human data, or object recognition, the VSL approach allows the robot to learn a variety of complex tasks effortlessly, simply by observing and reproducing the visual

[1] The main results of this chapter were previously published in [129–131].

relationship among objects. The feasibility of the proposed approach is demonstrated in several simulated and real-world experiments in which the robot learns to organize objects of different shape and color on a tabletop workspace to accomplish a goal configuration. In the conducted experiments, the robot acquires and reproduces six main capabilities:

- Absolute object reconfiguration
- Relative object reconfiguration
- Classification
- Turn-taking
- User intervention to modify the reproduction
- Multiple operation performed on the same object

To learn a new skill from demonstrations, the proposed approach uses 2D observations as the main source of information. Since using 2D observations limits VSL to planar tasks, the steps required for extending VSL for dealing with 3D applications are also investigated in this chapter. The result show that extending the approach to 3D space requires more sophisticated image processing techniques.

Furthermore, to augment the capabilities of VSL, an integrated learning framework is developed in this chapter. The integrated framework comprises a motor skill learning layer, VSL, and a symbolic planner. The robot learns the primitive actions (e.g. pull, pick) using the motor skill learning layer which is a trajectory-based approach. It also learns the sequence of actions through VSL. The symbolic planner enables the robot to learn the maintain a generalized representation of each action which can be planed and performed independently. The integration of VSL and a symbolic planner is a substantial contribution of this chapter, which has a potential for the future. The coupling between VSL and symbolic planner is natural and intuitive and brings significant advantages over using VSL separately. Learning preconditions and effects of the actions in VSL approach and making symbolic plans based on the learned knowledge deals with the long-standing important problem of Artificial Intelligence, namely Symbol Grounding. The feasibility and capability of the proposed framework are experimentally validated. In the

experiments it has been shown that the symbols and rules are grounded through learning in the problem domain.

The rest of the chapter is organized as follows. Related work is reviewed in Section 4.2. The terminology and methodology of the VSL approach are explained in details in Section 4.3. Implementation steps for the VSL approach are described in Section 4.4. Results for simulated and real-world experiments are reported in Section 4.5. The VSL approach is extended to 3D in Section 4.6. Experimental results for 3D real-world experiments are reported in Section 4.6.3. In Section 4.7, VSL is used for learning symbolic representation of actions and a novel learning framework is proposed. The implementation steps for the proposed framework are discussed in Section 4.7.6. Experimental results using the proposed framework are reported in Section 4.7.7. And finally, conclusions of the chapter are drawn in Section 4.8.

4.2 Related Work

Visual skill learning or learning by watching is one of the most powerful mechanisms of learning in humans. It has been shown that even infants can imitate both facial and manual gestures [132, 133]. In cognitive science, learning by watching has been investigated as a source of higher order intelligence and fast acquisition of knowledge [134–136]. In the rest of this section, related and most recent approaches that engage the topic of robot learning using visual perception are discussed. The mentioned works in this section, include those that can be categorized as goal-based, utilize visual perception as the main source of information, and especially those which focus on the learning of object manipulation tasks.

One of the most influential works on the problem of plan extraction from observation is that by Ikeuchi and Suehiro [137, 138]. To extract assembly plans from observations, the system obtains a continuous sequence of images. The segmentation is accomplished using level change in the brightness differences. The objects are detected and recognized using background subtraction and feature match finding respectively. A set of simple features (e.g. edge, face) and a pre-defined geometric model are used in the match

finding process. By comparing two sets of object relations, the transition between them are extracted. The system determines the assembly relations by analyzing contact directions. All existing assembly relation transitions are extracted by analyzing each assembly relation and unnecessary relations transitions are pruned. The system also determines manipulator operations from the assembly relations. The reproduction phase in the conducted experiments were neglected and the authors just focused on the extracting of task plans from observation. One of the drawbacks of their approach is that it requires a geometric model and a predefined coordinate system for each object. The defined coordinate systems are used to determine the grasping configuration and orientation which are predetermined for each object in the system.

The early works of Kuniyoshi *et al.* focus on acquiring reusable high-level task knowledge by watching a demonstration [6, 139]. In their *learning by watching* approach multiple vision sensors are used to monitor the execution of the task. The focus of the paper is on the *seeing* (i.e. demonstration) and *understanding* (i.e. learning) parts and the paper lacks a reproduction or *doing* section. By extracting some basic visual features from the observation, the object recognition system finds a match between the observation and a 3D model of the environment. The system assigns a symbol for each action and the executed actions are recognized from a pre-defined action database. From the set of recognized executed actions a high-level task plan is extracted. To figure out which object in a real scene is moving, human-hand-recognition based on temporal image subtraction is used. Since tracking the hand is not sufficient for classifying assembly operations, the meaningful changes have also been tracked. The meaningful changes are detected by subtracting pre-action and post-action observations based on area thresholding. In addition, the direction to move the search window is detected from the movement of the hand. The approach can only deal with rectangular objects and translational movements and the system cannot detect rotations. The other disadvantage of the approach is that the model-based shape recognition is computationally expensive and after each operation and before the reproduction phase the model of the environment has to be modified accordingly. This modification increases the complexity of the approach. Finally, the method cannot detect and recognize objects in contact as separate objects.

Asada *et al.* proposed a method for learning by observation (teaching by showing) based on the demonstrator's view recovery and adaptive visual servoing [7]. Assuming that coordinate transformation is a time-consuming and error-prone method, their proposed approach is based on the consideration that both the robot and the tutor have the same body structure. The method utilizes two sets of stereo cameras, one for observing the robot's motions and the other for observing the tutor's motions. The optic-geometrical constraint, called *epipolar constraint*, is used to reconstruct the view of the agent, on which adaptive visual servoing is applied to imitate the observed motion. In this chapter instead, coordinate transformation between the sensor and the robot is used.

Ehrenmann *et al.* proposed a learning from observation system using multiple sensors in a kitchen environment with typical household tasks [140]. They focused on pick-and-place operations including techniques for grasping. A data glove, a magnetic field based tracking system and an active trinocular camera head were used in their experiment. Object recognition is done using fast view-based vision approaches. Also, finger joint movements and hand position in 3D space are extracted from the data glove. The method is based on pre-trained neural networks to detect hand configurations and to search in a predefined symbol database. However, there was no real-world reproduction with a robot. A similar research that focuses on extracting and classifying subtasks for grasping tasks using visual data from demonstration, generating trajectory and extracting subtasks was proposed in [141]. The proposed approach uses color markers to capture data from the tutor's hand. The proposed approach in this chapter, neither uses neural networks nor symbol abstraction techniques.

A visual learning by Imitation approach was presented by Lopes and Santos [8]. In order to map visual perception to motor skills (visuo-motor) neural networks together with viewpoint transformation are utilized. For gesture imitation a Bayesian formulation is adopted. Their approach uses a single camera in the experiments.

The method proposed by Pardowitz *et al.* extracts knowledge about a task from sequence of actions [142]. The target task is setting a table using pick-and-place actions. The robot extracts the proper primitive actions and the constraints of the the task from demonstrations.

Pastor *et al.* presented an approach using which the learned motor skills from demonstrations are encoded into a non-linear differential equation that can reproduce those skills afterwards [40]. In order to provide primitive actions with semantics, the concept of Object-Action Complex [143] is used. However, the resulting symbolic actions are never used in a high-level planner.

A symbolic learning approach was proposed by Ekvall and Kragic in which a logical model for a STRIPS planner from multiple human demonstrations is learned [144, 145]. In their works, a task planning approach is used in combination with robot learning from demonstration. The robot generates states and identifies constraints of a task incrementally according to the order of the action execution In addition, In their method a demonstration is an image of the target configuration and they do not utilize observations of each action. Differently from the proposed approach in this chapter, the objects are first modeled geometrically and a set of SIFT features for each object is extracted in off-line mode and used during the learning phase. The method can only deal with polyhedral objects and a limited number of predefined grasp strategies are defined. A grasp strategy is selected based on the result from a collision detection module. The collision detection module works based on the model of the environment in 2D and it becomes slow by increasing the number of objects. Furthermore, when dealing with multiple demonstrations in object manipulation tasks, one problem is how to automatically decide the precision of the positioning from observations. To address this issue, K-means clustering is utilized to quantize the position and orientation for a specific object into a number of subgroups.

The spatial relations model, presented in [146] has several limitations that prevents it from being implemented on a real robot. For instance, they assume a perfect and noiseless visual information and a perfect object segmentation. Also a small grammar is used which means that the system just works for a few carefully constructed expressions.

A symbolic goal-based learning approach was presented by Chao *et al.* that can ground discrete concept from continuous perceptual data using unsupervised learning [128]. In addition, Bayesian inference is used in order to learn task goals. In the conducted experiments, five non-expert tutors have been employed. Each tutor performs multiple

demonstrations for five pick-and-place tasks. Each task consists of a single pick-and-place operation. The experiments were conducted using the 'Simon' humanoid robot. Before performing each demonstration, the initial configuration of the workspace is randomized by the tutor. Background subtraction is used for segmentation and the shelf is detected using a marker. The visual features extracted from each segmented object include, width and length of the best fitting ellipsoid, the area of the simplified polygon, and the hue histogram. During each operation, the approach detects the object that changes most significantly. The starting and ending time of each demonstration is provided using graphical or speech commands. VSL solely relies on the captured observations to learn the sequence of operations and there is no need to perform any of those steps.

A room tidying task in a simplified environment was performed and presented in [147]. The tutor instructs the robot using verbal commands. The low-level commands are pre-programmed into the system. The location of the objects, discrete states of the world, and configuration of the objects are encoded as features. A database of features for all possible configurations of the objects is created and labeled manually by the tutor. A beta regression classifier is used to learn the features to detect 'good' and 'bad' configurations. Since the system relies on a large database of features, the state-space is discretized and simulated annealing is used to find an optimal solution. In the demonstration phase, the tutor provides the robot with a sequence of actions to reach a desired goal. The robot, however, considers only the result of the task not the actions. Using a planner, it can later plan the task and generate proper actions. The main assumption is that the robot can solve the task without asking for help. This assumption actually limits the approach to be able to deal with very simple tasks. The conducted experiments in [147] are interesting because they can be achieved easily using VSL by demonstrating the task to the robot.

A framework for manipulation tasks has been introduced by Kroemer and Peters that includes two main layers. The former optimizes the parameters of the motor primitives, whereas the latter relies on pre- and post-conditions to model actions [148]. In contrast to VSL, the pre- and post-conditions are manually engineered into the system. Aksoy *et al.*, showed how symbolic representations can be linked to the trajectory level using spatiotemporal relations between objects and hands in the workspace [149]. In a similar

way, Aein *et al.*, showed how such grounded symbolic representations can be employed together with low level trajectory information to create action libraries for imitating observed actions [150].

There is a large body of literature on grammars from the linguistic and computer science communities, with a number of applications related to robotics [125,127]. Niekum *et al.* proposed a method to learn from continuous demonstrations by segmenting the trajectories and recognizing the skills [125]. Segmentation and recognition are achieved using a Beta-Process Autoregressive HMM, while Dynamic Movement Primitives are used to reproduce the skill. Their framework can be categorized as a trajectory-based technique that has been used to learn a goal-based task. The framework has been applied in simulation and in the real-world on a PR2 mobile manipulator. Joint data, gripper data, and stereo vision data are recorded during the demonstration. The method has been applied to rectangular objects which are manipulated without rotation. Object detection is done just in the simulation. Instead, in the real-world experiment, AR markers are utilized. In each experiment, for each known object a coordinate frame is assigned manually. These coordinate frames are used to relate the objects to the demonstrated skills. One of the drawbacks of their method is that sometimes the segmentation process (i.e. the BP-AR-HMM technique) extracts an extra skill from the continuous trajectories. Since a coordinate frame has to be identified from each extracted segment, for the detected extra skill the system fails to identify a coordinate frame. Another disadvantage of this framework is that, increasing the number of skills in the demonstrations makes the segmentation process increasingly complicated.

In order to planning and control of the operation of a robot through a context-free grammar, Dantam *et al.* proposed Motion Grammar [151]. The motion grammar is a linguistic method in which the grammar is used to generate motions for the robot. The method was implemented to the problem of human-robot chess and the experiments were conducted using a Schunk LWA3 7 DoF robot arm with a Schunk SDH 7 DoF, 3-fingered hand. A point cloud of the chessboard is used to detect the pieces. In another work, the motion grammar approach has been applied to the interactive game of Yamakuzushi [152]. To detect the pieces in the workspace, first a point cloud is extracted from which a plane is

built for each object based on its angle with respect to the angle of the table. The highest precedence plane is selected as the target to move. One of the drawbacks is that the grammars and the parsers have to be constructed manually.

Visual analysis of demonstrations and automatic policy extraction for an assembly task have been presented in [127]. To convert a demonstration of the desired task into a string of connected events, this approach uses a set of different techniques such as image segmentation, clustering, object recognition, object tracking, structure recognition, symbol generation, transformation of symbolic abstraction, and trajectory generation. The proposed approach in this chapter does not use symbol generation and symbol abstraction techniques.

A 3D object recognition and pose estimation for grasping in occluded environments has been proposed by Papazov *et al.* [153]. The method relies on 3D point cloud information and includes preprocessing and online phases. In the first phase the existing models are given to the off-line phase of the method in order to extract their features. The extracted features are used in the on-line phase for object recognition.

To deal with the spatial relations models presented in their previous research [146], Guadarrama *et al.* proposed a natural language interface for grounding nouns and spatial relations [154]. The data used for the training phase has been acquired via a virtual world. In the conducted experiments, a PR2 robot equipped with a laser scanner and an RGB-D sensor was used. The laser scanner creates the map and the RGB-D sensor is employed for segmentation and recognition of the objects. Their method learns how to classify a database of modeled objects. In contrast to VSL which uses minimum *a priori* knowledge, there is a huge database of collected images for each object which is used to train a classifier. Also, they apply a language module to learn related spatial prepositions. And finally, the presented applications and the spatial representations are limited to 2D representations.

Niekum *et al.* proposed an automatic method for segmenting continuous task demonstrations into primitives and finally grounding them semantically [155]. A Beta Process Autoregressive Hidden Markov Model is used for segmentation. A finite state representation of the task is incrementally build and improved. Additional data collected

during the interactive replay of the task is used for improving the primitives. A furniture assembly task using PR2 mobile manipulator was demonstrated.

By introducing a stack-based domain specific language for describing object repositioning tasks, Feniello *et al.* built a framework [156] based on the VSL approach proposed in this chapter. By performing demonstrations on a tablet interface, the time required for teaching is reduced and the reproduction phase can be validated before execution of the task in the real-world. Various types of real-world experiments were conducted including sorting, kitting, and packaging tasks.

There is a long history of research in the area of interpreting spatial relations [157–160]. In most of these works a model of spatial relations is build by hard-coding the meaning of the spatial relations. Instead, some recent works conclude that a learned model of spatial relations can outperform the hard-coded models [146, 154].

4.3 Introduction to Visuospatial Skill Learning

As previously stated, VSL is a goal-based robot learning approach consisting of two main phases, demonstration and reproduction. These phases are shown in Figure 4-1 which is a high-level flow diagram. VSL enables a robot to acquire new visuospatial skills for object manipulation by observing a demonstration. To explain different phases of the proposed approach, a virtual experimental setup for performing an object manipulation task is depicted in Figure 4-2. The virtual experimental setup consists of a robot manipulator equipped with a gripper, a tabletop workspace, a set of objects, and a vision sensor. A human tutor demonstrates a sequence of operations on the set of objects. Each operation consists of a set of primitive actions [2], e.g. pick, place. This section places the focus on object manipulation tasks, including pick-and-place operations [3], in which achieving the goal of the task and retaining the sequence of operations are particularly important. The robot acquires a set of observations from the demonstration using the vision sensor. The robot learns new skills using VSL. Afterwards, starting from a random initial configuration

[2] Throughout this chapter the terms action and primitive action are used equivalently.

[3] An operation is a sequence of actions e.g. pick-and-place operation. An action may consist of a number of sub-actions, for instance the pick action consists of two sub-actions, reach and grasp.

of the objects, the robot can perform a new sequence of operations which ultimately results in reaching the same goal as the one demonstrated by the tutor.

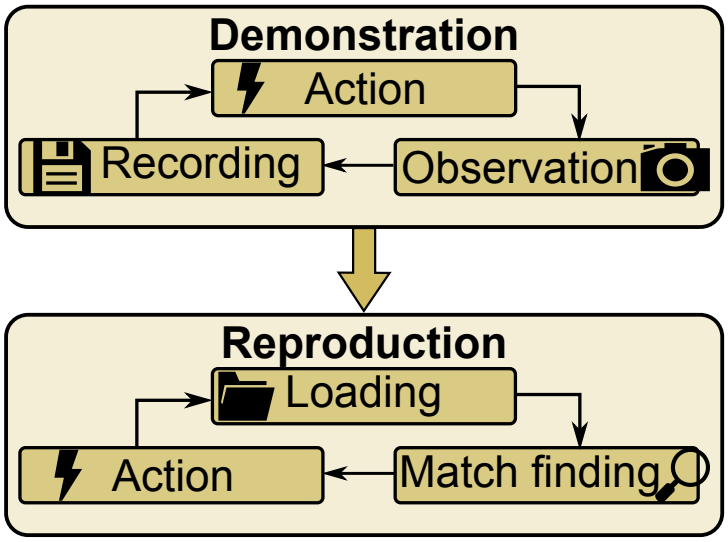


Figure 4-1: A high-level flow diagram illustrating the demonstration and reproduction phases in the VSL approach. For more information, see Section 4.3.

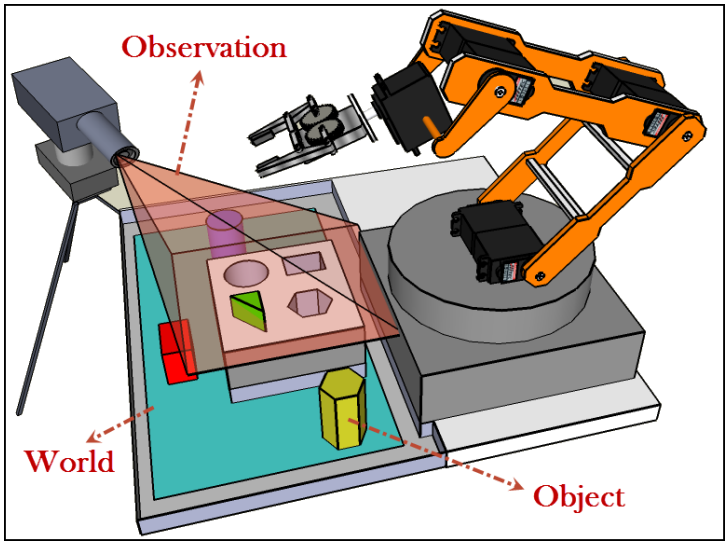


Figure 4-2: A schematic illustrating the virtual experimental setup for an object manipulation task. For more information, see Section 4.3.

This section introduces the Visuospatial Skill Learning (VSL) approach by defining the basic terms, describing the problem statement, and explaining the VSL algorithm in details.

4.3.1 Terminology

The basic terms that are used to describe VSL consist of:

- ***World*:** the workspace of the robot which is observable by the vision sensor. The *world* includes objects which are being used during the learning task, and can be reconfigured both by the human tutor and the robot.
- ***Frame*:** a bounding box which defines a cuboid in 3D space or a rectangle in 2D space. The size of the *frame* can be fixed or variable. The maximum size of the *frame* is equal to the size of the *world*.
- ***Observation*:** the captured context of the *world* from a predefined viewpoint using a specific *frame*. An *observation* can be a 2D image or a cloud of 3D points.
- ***Pre-action observation*:** an *observation* which is captured just before the action is executed. The robot searches for preconditions in the *pre-action observations* before selecting and executing an action.
- ***Post-action observation*:** an *observation* which is captured just after the action is executed. The robot perceives the effects of the executed actions in the *post-action observations*.

In order to reproduce a learned task, VSL should be able to perform operations. Because of that, VSL contains a library of primitive actions such as pick, place, push, etc. Therefore, the robot is able to execute each primitive action when required. In this section and in all of the experiments that are conducted solely using VSL (Sections 4.5 and 4.6.3), a trajectory generation module together with a torque-close grasping strategy are used for generating and executing primitive actions (For more information, see Section 4.4). In Section 4.7, on the other hand, VSL is integrated with a conventional trajectory-based technique that enables the robot to learn the primitive actions from demonstration.

4.3.2 Problem Formulation

Formally, a process of Visuospatial Skill Learning is defined as a tuple

$$\mathcal{V} = \left\{\mathcal{W}, \mathcal{O}, \mathcal{F}, \mathcal{A}, \mathcal{C}, \Pi, \phi\right\},$$

where $\mathcal{W} \in \mathbb{R}^{m \times n}$ is a matrix which represents the context of the *world* including the workspace and all objects. $\mathcal{W}_D$ and $\mathcal{W}_R$ indicate the *world* during the demonstration and reproduction phases respectively; $\mathcal{O}$ is a set of *observation* dictionaries $\mathcal{O} = \{\mathcal{O}^{pre}, \mathcal{O}^{post}\}$; $\mathcal{O}^{pre}$ and $\mathcal{O}^{post}$ are *observation* dictionaries comprising a sequence of *pre-action* and *post-action observations* respectively. $\mathcal{O}^{pre} = \langle \mathcal{O}^{pre}(1), \mathcal{O}^{pre}(2), \ldots, \mathcal{O}^{pre}(\eta) \rangle$, and $\mathcal{O}^{post} = \langle \mathcal{O}^{post}(1), \mathcal{O}^{post}(2), \ldots, \mathcal{O}^{post}(\eta) \rangle$. η is the number of operations performed by the tutor during the demonstration phase. Thereby, for example, $\mathcal{O}^{pre}(i)$ represents the *pre-action observation* captured during the i^{th} operation.

$\mathcal{F} \in \mathbb{R}^{m \times n}$ is an *observation frame* which is used for capturing the *observations*. $\mathcal{A}$ is a set of primitive actions defined in the learning task (e.g. pick). $\mathcal{C}$ is a set of *constraint* dictionaries $\mathcal{C} = \{\mathcal{C}^{pre}, \mathcal{C}^{post}\}$; $\mathcal{C}^{pre}$ and $\mathcal{C}^{post}$ are *constraint* dictionaries comprising a sequence of *pre-action*, and *post-action constraints* respectively. Π is a policy or an ordered action sequence extracted from demonstrations. ϕ is a vector containing extracted features from *observations* (e.g. SIFT features). Pseudo-code of VSL is given in Algorithm 2.

4.3.3 Methodology

At the beginning of the demonstration, the objects are randomly placed in the *world* ($\mathcal{W}_D$). The *world*, the size of the *frame*, and the set of primitive actions $\mathcal{A}$ are known to the robot. In the demonstration phase, the size of the *frame* ($\mathcal{F}_D$) is equal to the size of the *world* ($\mathcal{W}_D$). Either using a trajectory generation module (Section 4.4) or through learning from demonstration (Section 4.7.6), the robot is capable of selecting and executing the primitive actions from $\mathcal{A}$. For instance, the robot is capable of moving towards a desired given pose and execute a pick action. Implementation details can be found in Section 4.4.

In some tasks the tutor utilizes a landmark in order to specify different concepts.

Input : $\{\mathcal{W}, \mathcal{F}, \mathcal{A}\}$
Output: $\{\mathcal{O}, \mathcal{P}, \Pi, \mathcal{C}, \mathcal{B}, \phi\}$
1 **Initialization:** detect absolute landmarks if defined
2 $\mathcal{L}$=detectLandmarks($\mathcal{W}$)
3 $i = 1$, $j = 1$
// Part I : Demonstration
4 **for** *each Action* **do**
5 $\mathcal{O}_i^{pre} = \text{getPreActionObs}(\mathcal{W}_D, \mathcal{F}_D)$
6 $\mathcal{O}_i^{post} = \text{getPostActionObs}(\mathcal{W}_D, \mathcal{F}_D)$
7 $[\mathcal{B}_i, \mathcal{P}_i^{pre}, \mathcal{P}_i^{post}, \phi_i] = \text{getObject}(\mathcal{O}_i^{pre}, \mathcal{O}_i^{post})$
8 $[\mathcal{C}_i^{pre}, \mathcal{C}_i^{post}] = \text{getConstraint}(\mathcal{B}_i, \mathcal{P}_i^{pre}, \mathcal{P}_i^{post}, \mathcal{L})$
9 $\Pi_i = \text{getAction}(\mathcal{A}, \mathcal{C}_i^{pre}, \mathcal{C}_i^{post})$
10 $i = i + 1$
11 **end**
// Part II : Reproduction (this part is used in absence of symbolic planner.)
12 **for** $j = 1$ ***to*** i **do**
13 $\mathcal{P}_j^{*pre} = \text{findBestMatch}\,(\mathcal{W}_R, \mathcal{O}_j^{pre}, \phi_j, \mathcal{C}_j^{pre}, \mathcal{L})$
14 $\mathcal{P}_j^{*post} = \text{findBestMatch}\,(\mathcal{W}_R, \mathcal{O}_j^{post}, \phi_j, \mathcal{C}_j^{post}, \mathcal{L})$
15 executeAction($\mathcal{P}_j^{*pre}, \mathcal{P}_j^{*post}, \Pi_j$)
16 **end**

Algorithm 2: Pseudo-code for VSL. For more information, see Section 4.3.3.

For instance, a vertical borderline can be used to divide the workspace into two areas illustrating right and left zones. Another example is a horizontal borderline that can be used to specify far and near concepts. A landmark can be either static or dynamic. A static landmark is fixed with respect to the *world* during both phases. A dynamic landmark can be repositioned or transformed before reproduction phase. Both types of landmarks are used in the conducted experiments. In the case that, a landmark, e.g. label, borderline, is being used during the learning process, the robot should be able to detect it in the *world* (line 2 in Algorithm 2). The difference between a landmark and an object is that unlike an object, a landmark cannot be manipulated during the demonstration and reproduction phase. During the demonstration phase, VSL captures one *pre-action observation* ($\mathcal{O}^{pre}$) and one *post-action observation* ($\mathcal{O}^{post}$) for each action executed by the tutor using the specified *frame* ($\mathcal{F}_D$) (lines 5,6). The *pre-action* and *post-action observations* are used to detect the object on which the action is executed. The *observations* are also used to

detect the place where the object is repositioned at. For each detected object, a symbolic representation ($\mathcal{B}$) is created. The symbolic object can be used in the case that the reproduction phase of the algorithm is replaced with a high-level symbolic planner (For more information, see Section 4.7). In addition, for each detected object, VSL extracts a feature vector (ϕ). In order to extract ϕ, any feature extracting method can be employed (e.g. SIFT). The extracted features are used by VSL for detecting and recognizing the objects during the reproduction phase.

VSL also extracts the pose of the object before and after action execution ($\mathcal{P}^{pre}, \mathcal{P}^{post}$). The pose vectors together with the detected landmarks ($\mathcal{L}$) are used to identify preconditions and effects of the executed action through spatial reasoning (line 8). For instance, if $\mathcal{P}^{pre}$ is above a horizontal borderline and $\mathcal{P}^{post}$ is below the line, the precondition of the action is that the object is above the line and the effect of the execution of the action is that the object is below the line. In the other word, by observing the predicates of an executed action, the action can be identified from the set of actions $\mathcal{A}$ (line 9). The sequence of identified actions are then stored in a policy vector Π.

The second part of the algorithm is able to execute the learned sequence of actions independently [129, 130]. In such case, VSL observes the new *world* ($\mathcal{W}_R$) in which the objects are reconfigured randomly. Comparing the recorded *pre-* and *post-action observations*, VSL detects the best match for each object and executes the related primitive action from the learned policy. Although, the findBestMatch function can use any metric to find the best matching *observation*, to be consistent, in all of the experiments conducted in Section 4.5, the same metric is used (For more information, see Section 4.4.2). After finding the best match, the algorithm extracts the pose of the object before and after action execution, $\mathcal{P}_j^{*pre}$, $\mathcal{P}_j^{*post}$ (lines 13 and 14). Finally, an action is selected from the policy Π and is sent together with the pre and post poses to the executeAction function. This function selects the $\mathcal{A}_j$ primitive action. As previously stated, the robot knows how to perform a primitive action, for instance it uses a trajectory generation module and a grasping strategy to perform a pick action. More details about the action execution can be found in Section 4.4. It is worth noting that to reproduce the task with more general capabilities, in Section 4.7, a generic symbolic planner is utilized.

4.4 Implementation of VSL

In this section, the steps required to implement the VSL approach for real-world experiments are described.

4.4.1 Coordinate Transformation

In order to transform points between the coordinate frame of the sensor (image plane) and the coordinate frame of the robot (workspace plane) and vice versa a coordinate transformation is required (see Figure 4-3). The coordinate transformation between the sensor's frame of reference and the robot's frame of reference is done using coordinate transformation matrix **T**. This matrix is calculated using Singular Value Decomposition (SVD) by collecting two sets of points, one from the workspace and the other from the captured image [161]. Whenever the sensor's frame of reference is changed with respect to the robot's frame of reference, **T** has to be recalculated. Using the updated coordinate transformation matrix, VSL can reproduce the learned skill even if the experimental setup is altered. It means that VSL is a view-invariant approach. For more details on coordinate transformation check [161].

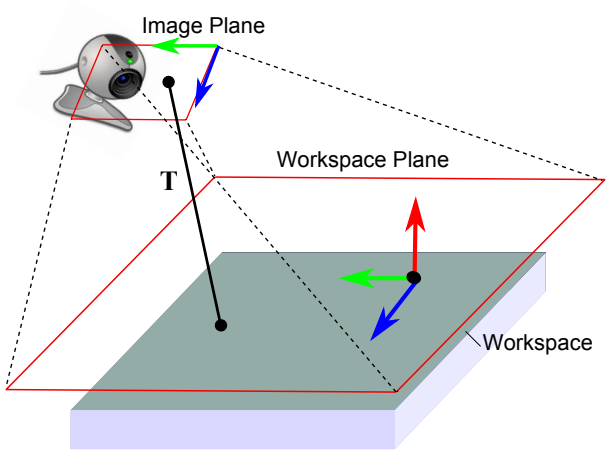


Figure 4-3: Coordinate transformation from the image plane to the workspace plane (**T** represents the homography matrix). For more information, see Section 4.4.1.

4.4.2 Image Processing

Image processing methods have been employed in both demonstration and reproduction phases of VSL. In the demonstration phase, for each operation the algorithm captures a set of raw images consist of *pre-action* and *post-action observations*. Firstly, the captured raw images are rectified using the homography matrix $\mathbf{T}$. Secondly, background subtraction and thresholding techniques are applied on the couple of images to generate *pre-action* and *post-action observations*. The extracted *observations* are centered around the *frame*. In the reproduction phase, for each operation the algorithm rectifies the captured *world observation*. Then, the corresponding recorded *observations* are loaded from the demonstration phase and a metric is applied to find the best match (the findBestMatch function in the Algorithm 2). Although any metric can be used in this function (e.g. window search method), in the first set of experiments in Section 4.5, Scale Invariant Feature Transform (SIFT) algorithm [162] is used. SIFT is one of the most popular feature-based methods which is able to detect and describe local features that are invariant to scaling and rotation. Afterwards, in order to estimate the transformation matrix $\mathbf{T}_{\text{sift}}$ from the set of matches, RANSAC technique is applied. Since the calculated transformation matrix $\mathbf{T}_{\text{sift}}$ has 8 degrees of freedom, with 9 elements in the matrix, to have a unique normalized representation, $\mathbf{T}_{\text{sift}}$ is pre-multiplied with a normalization constant:

$$\alpha = \frac{\text{sign}(\mathbf{T}_{\text{sift}}(3,3))}{\sqrt{(\mathbf{T}_{\text{sift}}(3,1)^2 + \mathbf{T}_{\text{sift}}(3,2)^2 + \mathbf{T}_{\text{sift}}(3,3)^2)}}. \tag{4.1}$$

This normalization constant is selected to make the decomposed projective matrix have a vanishing line vector of unit magnitude and that avoids unnatural interpolation results. The normalized matrix $\alpha\mathbf{T}_{\text{sift}}$ can be decomposed into simple transformation elements,

$$\alpha\mathbf{T}_{\text{sift}} = \mathbf{T}\mathbf{R}_{\theta}\mathbf{R}_{-\phi}\mathbf{S}_{\nu}\mathbf{R}_{\phi}\mathbf{P}, \tag{4.2}$$

where $\mathbf{R}_{\pm\phi}$ are rotation matrices to align the axis for horizontal and vertical scaling of $\mathbf{S}_{\nu}$, $\mathbf{R}_{\theta}$ is another rotation matrix to orientate the shape into its final orientation; $\mathbf{T}$ is a translation matrix; and lastly $\mathbf{P}$ is a pure projective matrix:

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \alpha\mathbf{T}(3,1) & \alpha\mathbf{T}(3,2) & \alpha\mathbf{T}(3,3) \end{bmatrix} \tag{4.3}$$

An affine matrix, $\mathbf{T}_A$, is the remainder of $\alpha\mathbf{T}$ by extracting $\mathbf{P}$; $\mathbf{T}_A = \alpha\mathbf{T}\mathbf{P}^{-1}$. $\mathbf{T}$ is extracted by taking the 3$^{\text{rd}}$ column of $\mathbf{T}_A$ and $\mathbf{A} \in \mathbb{R}^{2\times 2}$, is the remainder of $\mathbf{T}_A$. $\mathbf{A}$ can be further decomposed using SVD such that,

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{4.4}$$

where $\mathbf{D}$ is a diagonal matrix, and $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices. Finally we can calculate

$$\mathbf{S}_v = \begin{bmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}, \mathbf{R}_\theta = \begin{bmatrix} \mathbf{U}\mathbf{V}^T & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}, \mathbf{R}_\phi = \begin{bmatrix} \mathbf{V}^T & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.5}$$

Since the conducted experiments in Section 4.5 include two primitive actions, pick and place, $\mathbf{R}_\theta$ is calculated for both pick and place operations ($\mathbf{R}_\theta^{pick}, \mathbf{R}_\theta^{place}$). The pick and place rotation angles of the objects are extracted as follows:

$$\begin{aligned} \theta_{pick} &= \arctan\left(\frac{\mathbf{R}_\theta^{pick}(2,2)}{\mathbf{R}_\theta^{pick}(2,1)}\right) \\ \theta_{place} &= \arctan\left(\frac{\mathbf{R}_\theta^{place}(2,2)}{\mathbf{R}_\theta^{place}(2,1)}\right) \end{aligned} \tag{4.6}$$

Note that projective transformation is position-dependent compared to the position-independent affine transformation. More details about homography estimation and decomposition can be found in [163].

Finally, it should be mentioned that the image processing part is not the focus of this thesis. Thus, some efficient and robust image processing techniques are employed for the real-world experiments. For instance in the real-world experiments in Section 4.5.2 the SIFT-RANSAC algorithms are used because of their popularity and the capability of fast and robust match finding. Figure 4-4 shows the result of applying background subtraction

and thresholding techniques to a couple of *observations*. It also illustrates the result of match finding using SIFT in a real-world experiment.

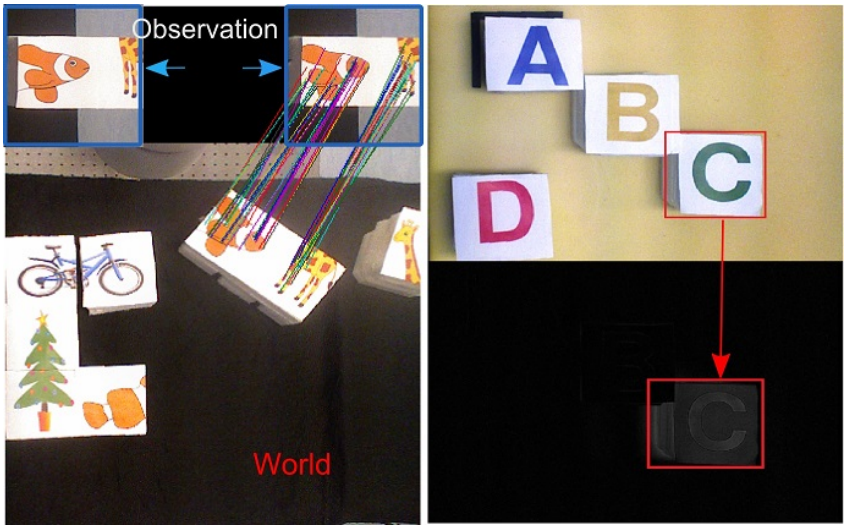


Figure 4-4: Image processing techniques utilized in VSL. The result of applying the background subtraction and thresholding for a place action (right). Match finding result between the 4^{th} observation and the *world* in the 4^{th} operation during reproduction of the Domino task using SIFT (left). For more information, see Section 4.4.2.

4.4.3 Segmentation

The temporal segmentation of human activities and gestures is a difficult issue which is still under investigation [164, 165]. In VSL, recognition of each individual operation during the continuous task performance requires segmentation. Many simplified solutions to the problem of segmentation in object manipulation tasks have been proposed [6, 138, 139, 165]. Since the focus of this thesis is not placed on solving the problem of automatic segmentation, a simple yet effective approach is utilized. During the demonstration phase, the vision sensor continuously captures and records the workspace of the robot at a specific frame rate. The frame rate is set somehow that there is enough time for the tutor to operate between each capture till the next one (e.g. $5-10$sec). This delay helps to avoid capturing a snapshot including unwanted effects (e.g. the tutor's hand). However, the tutor can later manually check and discard such snapshots from the set. When the demonstration phase is completed, the segmentation process is applied to the set of recorded snapshots. By looking at a couple of consecutive snapshots, meaningful

changes are detected. The meaningful changes are those including operations performed on the objects by the tutor. Three simple techniques are applied to the set of raw images including absolute difference, histogram, and correlation.

Segmentation using Absolute Difference

The first technique is based on element-wise absolute differences of a couple of images. After applying the absolute difference technique, the difference vector for the sequence of images is re-scaled in range $[0, 1]$ and then a threshold is used to separate the boundary of the meaningful changes from noise. Figure 4-5 illustrates a set of captured snapshots and the result of applying the absolute difference segmentation technique. The observations containing meaningful changes are used in VSL and the rest of the observations are discarded.

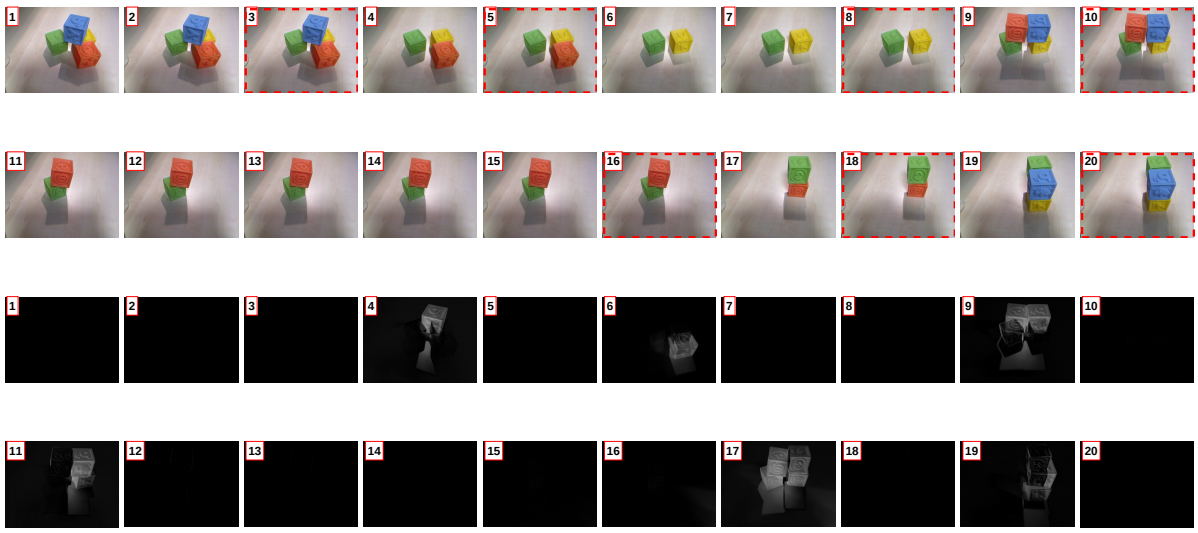


Figure 4-5: Segmentation process applied to a set of captured images from the scene during the demonstration. The meaningful changes in the scene are detected by calculating the absolute difference in each step. The snapshots including a meaningful change are highlighted with a red rectangle frame. The rest of the snapshots are discarded. The bottom row shows the absolute difference between two consecutive snapshots. For more information, see Section 4.4.3.

Segmentation using Histogram

The second techniques is based on the changes in the histogram counts from one image to the next. The Euclidean distance between estimated counts, determines the changes. After applying the histogram technique, the difference vector for the sequence of images is re-scaled in range $[0, 1]$ and then a threshold is used to decide the boundary of meaningful changes from noise. The segmentation technique based on histogram counts is applied to the same set of images and the result is shown in Figure 4-6.

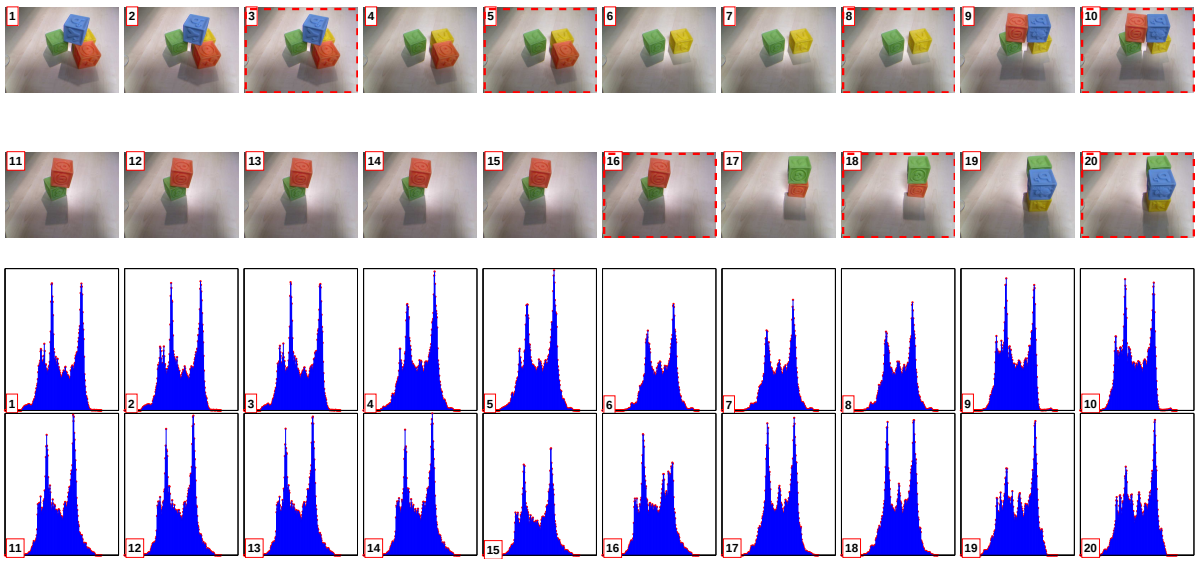


Figure 4-6: Segmentation process applied to a set of captured images from the scene during the demonstration. The meaningful changes in the scene are detected by calculating the histogram counts in each step. The snapshots including a meaningful change are highlighted with a red rectangle frame. The bottom row shows the histogram in each step. For more information, see Section 4.4.3.

Segmentation using Correlation

The third technique is based on the 2D correlation coefficient. The correlation coefficient between two identical image is equal to one. After applying the correlation technique, the difference vector for the sequence of images is re-scaled in range $[0, 1]$ and then a threshold is used to decide the boundary of meaningful changes from noise. Figure 4-7 illustrates the results of applying the segmentation technique based on 2D correlation coefficients. In all three experiments, the meaningful snapshots are recorded and the rest are discarded.

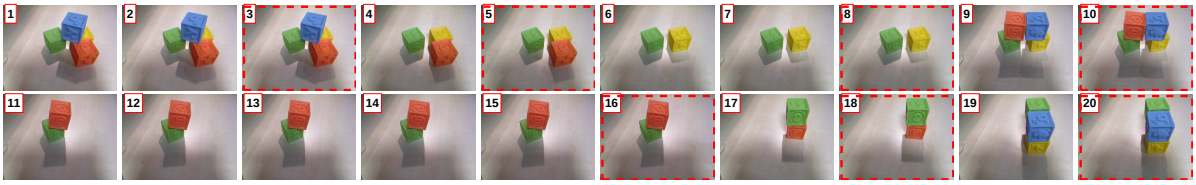


Figure 4-7: Segmentation process applied to a set of captured images from the scene during the demonstration. The meaningful changes in the scene are detected by calculating the 2D correlation coefficient in each step. The snapshots including a meaningful change are highlighted with a red rectangle frame. The rest of the snapshots are discarded. For more information, see Section 4.4.3.

Figure 4-8 depicts the re-scaled difference results obtained from the mentioned segmentation techniques. The red bars highlights the snapshots containing meaningful operations.

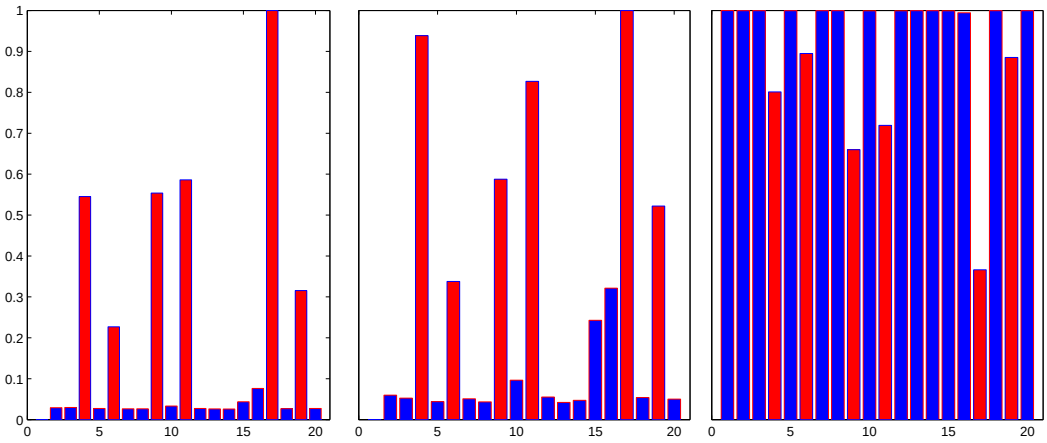


Figure 4-8: From left to right, the re-scaled absolute difference, the re-scaled histogram difference, and the re-scaled 2D correlation coefficient for the set of images shown in Figure 4-7. The red bars highlights the snapshots containing the meaningful operations. For more information, see Section 4.4.3.

4.4.4 Trajectory Generation

In order to perform an operation, for instance a pick-and-place operation, the robot must execute a set of primitive actions consisting of reaching, grasping, relocating, and releasing. Both reaching and relocating primitive actions correspond to a trajectory. These trajectories can either be manually programmed into the system or a tutor can teach them to the robot for instance through Learning by Demonstration [32]. In this section, a simple trajectory generation strategy has been used. The pick and place points together with the

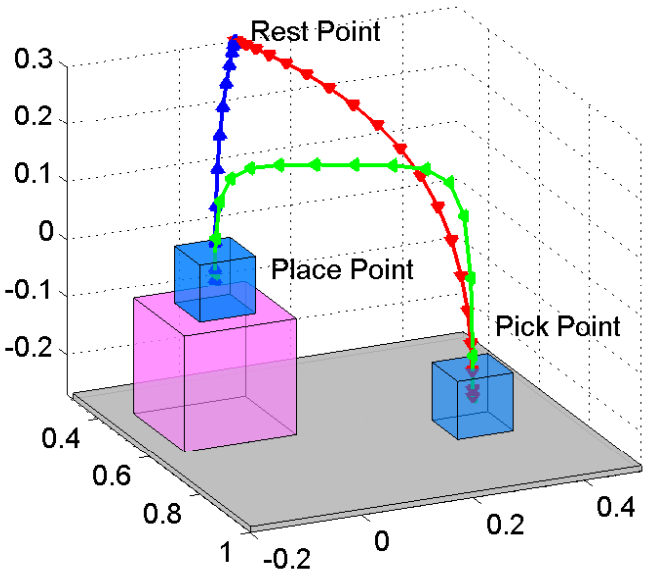
pick and place rotation angles extracted from the image processing section, are used to generate a trajectory for the corresponding operation. For each pick-and-place operation the desired Cartesian trajectory of the end-effector is a cyclic movement between three key points: rest point, pick point, and place point. Fig 4-9a illustrates a complete trajectory generated for a pick-and-place operation. Also, four different profiles of rotation angles are depicted in Fig 4-9b. The robot starts from the rest point while the rotation angle is equal to zero. It moves smoothly along the red curve towards the pick point. During this movement the robot's hand rotates to satisfy the pick rotation angle according to the rotation angle profile. Then the robot picks up an object, relocates it along the green curve to the place-point, while the hand is rotating to meet the place rotation angle. Then, the robot places the object in the place-point, and finally moves back along the blue curve to the rest-point. The initial and final rotation angles are considered to be zero. In order to form each trajectory, initial conditions (i.e., initial positions and velocities) and a specific duration are defined. Thereby, a geometric path is defined which can be expressed in the parametric form of (4.7)-(4.9),

$$p_x = a_3 s^3 + a_2 s^2 + a_1 s + a_0, \tag{4.7}$$

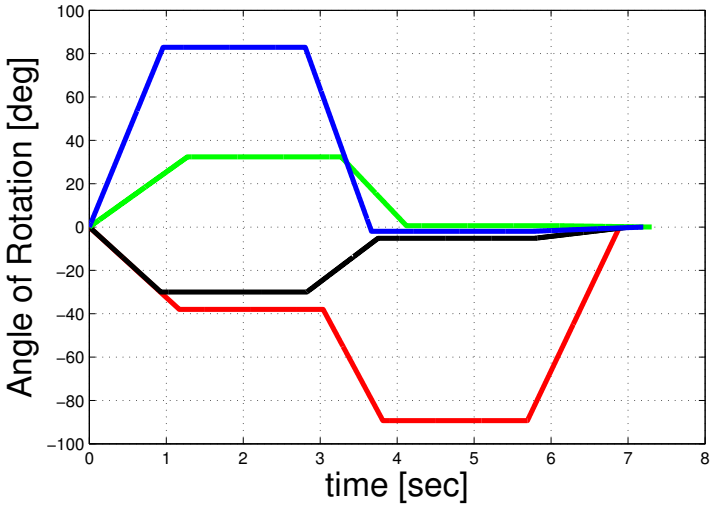
$$p_y = b_3 s^3 + b_2 s^2 + b_1 s + b_0, \tag{4.8}$$

$$p_z = h[1 - |(\tanh^{-1}(h_0(s-0.5)))^{\kappa}|], \tag{4.9}$$

where, s is a function of time t, $(s = s(t))$, $p_x = p_x(s)$, $p_y = p_y(s)$, and $p_z = p_z(s)$ are the 3D elements of the geometric spatial path. κ is the curvature, h_0 and h are initial height and height of the curve in the middle point of the path respectively. h and h_0 can be either provided by the tutor or detected through depth information provided by an RGB-D sensor. The a_i and b_i coefficients are calculated using the initial and final conditions. In addition, the time is smoothly distributed with a 3$^{\text{rd}}$ order polynomial between the t_{start} and t_{final} which both are instructed by the tutor.



(a) A full cycle of spatial trajectory generated for a pick-and-place operation.



(b) The generated angle of rotation, θ, for the robot's hand.

Figure 4-9: The generated trajectory including position and orientation profiles for a spatial pick-and-place. For more information, see Section 4.4.4.

Moreover, a trapezoidal profile is used together with the extracted θ_{pick} and θ_{place} to generate a rotation angle trajectory for the robot's hand. As shown in Figure 4-9a, the trajectory generation module can also deal with objects placed in different heights (different z-axis levels). The grasp strategy is discussed in the next section.

4.4.5 Grasp Synthesis

There are many elaborate ways to do grasp synthesis for known or unknown objects [166, 167]. Since the problem of grasping is not the main focus of our research, a simple yet effective grasping method using the torque sensor of the Barrett Hand is implemented. In the 2D real-world experiments in Section 4.5, the grasp position is calculated at the center of the corresponding *pre-action* observation. In the 3D real-world experiment using 3D VSL, in Section 4.6.3, the grasp pose is calculated using the centroid of the detected point cloud (i.e. the object to grasp). The grasping module which controls the hand firstly opens all the fingers and after the hand is located above the desired object, the controller closes the fingers. The fingers stop closing when the measured torque is more than a pre-defined threshold value. In addition, by detecting the bounding box for the target observations, the values are used to decide which axis is more convenient for grasping. Figure 4-10 illustrates a Barrett Hand including three fingers grasping a polyhedral object. Figure 4-11 shows a Barrett WAM robot applying the explained grasp strategy in a real-world experiment.

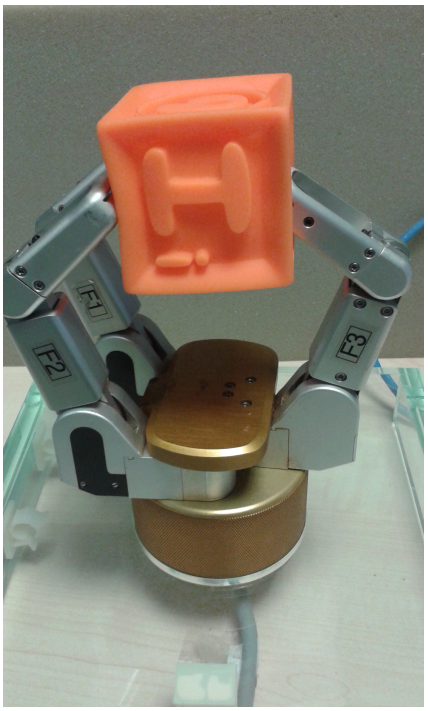


Figure 4-10: The Barrett Hand grasping an object on the test-stand. For more information, see Section 4.4.5.

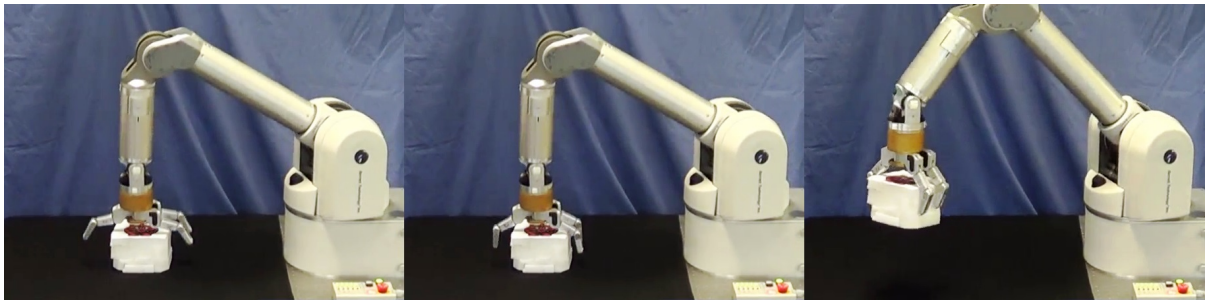


Figure 4-11: Grasping an object during a real-world experiment. For more information, see Section 4.4.5.

4.5 Experimental Results of VSL

In this section a set of simulated and real-world experiments are carried out. The simulated experiments are designed to gain an understanding of how VSL operates and to show the main idea of VSL without dealing with practical limitations and implementation difficulties. The real-world experiments, on the other hand, are conducted to validate the main capabilities and limitations of VSL in practice while dealing with uncertainties.

4.5.1 Simulated Experiments

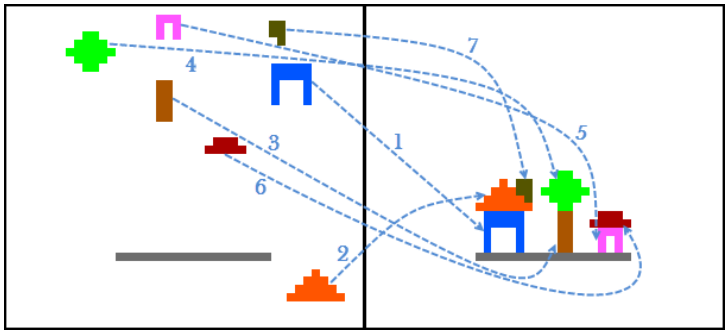
In this section, four simulated experiments are performed to illustrate the main idea behind the VSL approach. For each simulated experiment a set of 2D objects is made which the tutor can manipulate and assemble them on an empty workspace using keyboard or mouse. Each operation consists of a *pick* and a *place* action, which are executed by holding and releasing a mouse button.

Before describing the conducted experiments, it is worth noting that, in all the illustrations, the straight and curved arrows are used just to show the sequence of operations, not the actual trajectories for performing the movements.

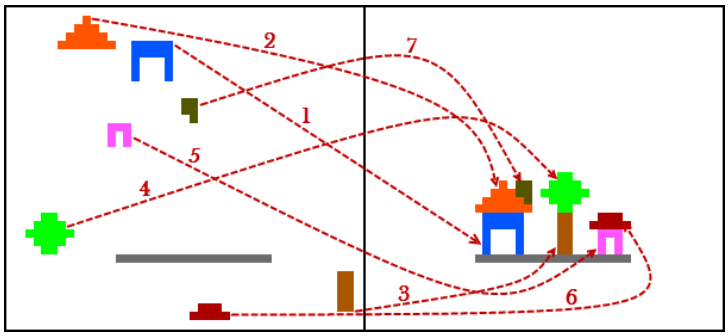
A House Scene

In the first VSL task, the *world* includes seven 2D objects with different colors. However, the objects are detected based on their shapes not their color features. The *world*

also includes a fixed baseline (i.e. a landmark $\mathcal{L}$) which cannot be manipulated. The goal is to assemble the set of objects on the baseline according to the demonstration. This task emphasizes VSL's capability of relative positioning of an object with respect to other surrounding objects in the *world*. This inherent capability of VSL is achieved through the use of visual *observations* which capture both the object of interest and its surrounding objects (i.e. its context). In addition, the baseline is provided to show the capability of absolute positioning of the VSL approach. It shows the fact that we can teach the robot to attain absolute positioning of objects without defining any explicit *a priori* knowledge. Figure 4-12a shows the sequence of operations performed in the demonstration phase. Recording *pre-action* and *pre-action observations*,$\mathcal{O}^{pre}$,$\mathcal{O}^{post}$, the robot learns the sequence of operations. Figure 4-12b shows the sequence of operations produced by VSL on a novel *world* which is achieved by reshuffling the objects randomly in the *world*.



(a) Demonstration by the tutor.



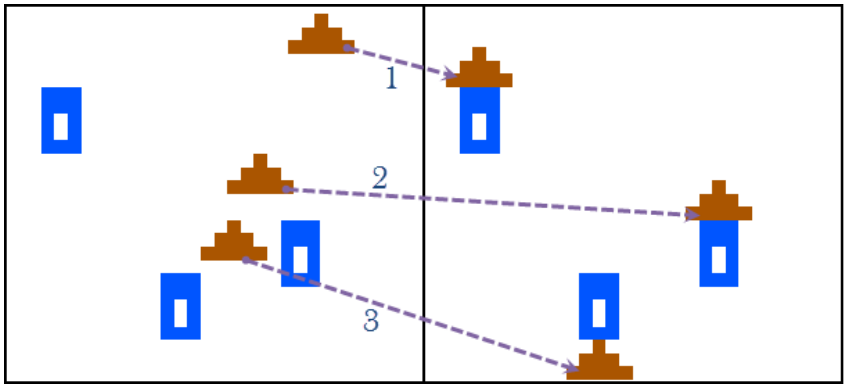
(b) Reproduction by the robot.

Figure 4-12: House scene simulated experiment to illustrate the relative and absolute positioning capabilities of VSL. For more information, see Section 4.5.1.

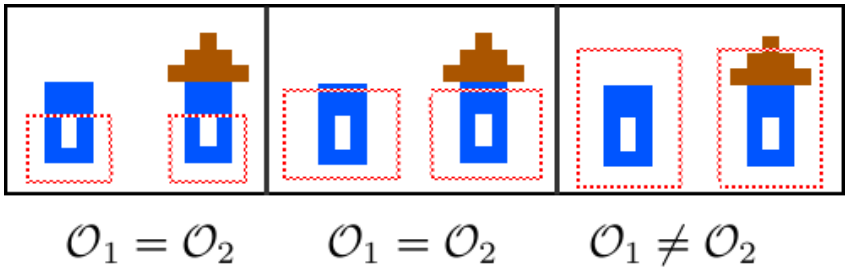
Roof Placement

In this task, the *world* includes two sets, each containing three visually identical objects (i.e., three blue 'house bodies' and three brown 'roofs'). As can be seen in Figure 4-13a, the tutor selects the 'roof' objects arbitrarily and places them on top of the 'bodies'. However, in the 3rd operation, the tutor intentionally puts the 'roof' at the bottom of the 'house body'.

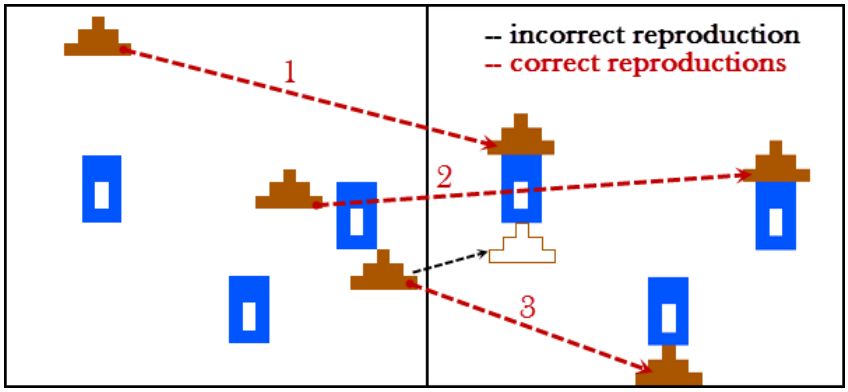
The goal of this experiment is to show the VSL's capability of disambiguation of multiple alternative matches. If the algorithm uses a fixed search *frame* ($\mathcal{F}_R$) that is smaller than the size of the 'bodies' (i.e., blue objects in the *world*), then, as shown in the first and second sub-figures 4-13b, the two captured observations can become equivalent (i.e., $\mathcal{O}_1 = \mathcal{O}_2$) and the the 3rd operation might be performed incorrectly (see the incorrect reproduction in Figure 4-13c). The reason is that, due to the size of the *frame*, the system perceives a section of the *world* not bigger than the size of a 'house body'. The system is not aware that an object is already assembled on the top and it will select the first matching *pre-place* position to place the last object there. To resolve this problem we use adaptive size *frames* during the match finding process in which the size of the *frame* starts from the smallest feasible size and grows up to the size of the *world*. The function findBestMatch in Algorithm 2, is responsible for creating and changing the size of the *frame* adaptively in each step. This technique helps the robot to resolve the ambiguity issue by adding more context inside the observation, which in effect narrows down the possible matches, until it resolves the ambiguity by leaving only a single matching observation. Finally, the sequence of operations in the reproduction phase, including correct and incorrect operations, is shown in Figure 4-13b.



(a) Demonstration



(b) Three steps of adaptive frame-size



(c) Reproduction

Figure 4-13: Roof placement simulated experiment to illustrate VSL's capability of disambiguation of multiple alternative matches. For more information, see Section 4.5.1.

Tower of Hanoi

The last simulated experiment is the famous mathematical puzzle, Tower of Hanoi, which consists of a number of disks of different sizes and three bases or rods which actually are landmarks. The objective of the puzzle is to move the entire stack to another rod. This experiment demonstrates almost all capabilities of the VSL approach. Two of these

capabilities are not accompanied by the previous experiments. Firstly, VSL enables the user to intervene to modify the reproduction. Such capability can be used to move the disks to another base (e.g. to move the stack of disks to the third base, instead of the second). This can be achieved only if the tutor performs the very first operation in the reproduction phase and moves the smallest disk on the third base instead of the second. Secondly, the VSL approach enables the tutor to perform multiple operations on the same object during the demonstration phase. Figure 4-14 illustrates the reproduction sequence of the Tower of Hanoi puzzle, including three disks.

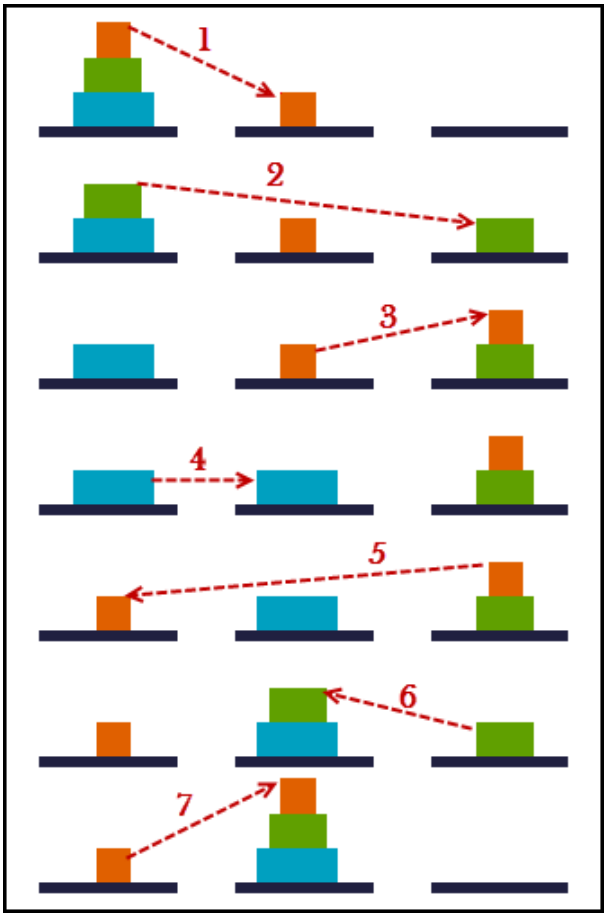


Figure 4-14: Tower of Hanoi simulated experiment to illustrate the capabilities of VSL to perform interactive reproduction, disambiguation, and absolute positioning. For more information, see Section 4.5.1.

4.5.2 Real-world Experiments

After performing three simulated experiments that illustrate the main idea of VSL, in order to validate the feasibility and capabilities of the proposed learning approach, in this section five real-world experiments are conducted[4].

Experimental Setup

As can be seen in Figure 4-15, the experimental setup for all the conducted real-world experiments in this section, consists of a torque-controlled 7 DoF Barrett WAM robotic arm equipped with a 3-finger Barrett Hand, a tabletop working area, a set of objects, and a CCD camera which is mounted above the workspace (not necessarily perpendicular to the workspace).

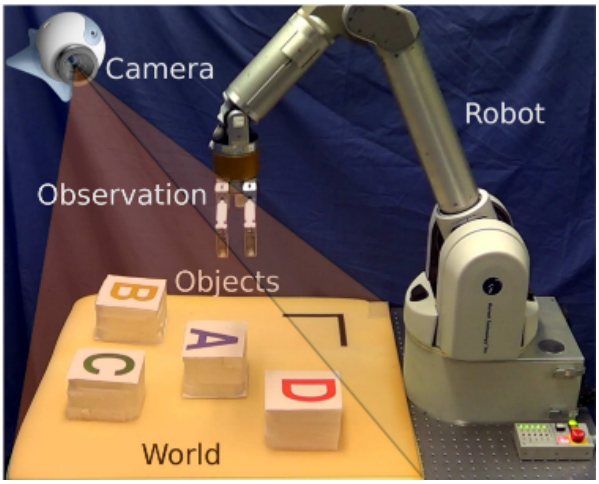


Figure 4-15: The experimental setup for a Visuospatial Skill Learning (VSL) task. For more information, see Section 4.5.2.

In all the conducted experiments, the robot learns simple object manipulation tasks including pick and place primitive actions. In order to perform a pick-and-place operation, the extracted pick and place poses are used to make a cyclic trajectory as explained in Section 4.4.4. The grasp strategy is implemented based on the method explained in Section 4.4.5. In the demonstration phase, the size of the *frame* for the *pre-*

[4] A video of the real-world experiments reported in this section is available online at my personal website http://www.ahmadzadeh.info/videos.

action observation is set equal to the size of the biggest object in the *world*, and the size of the *frame* for the *post-action observation* is set 2–3 times bigger than the size of the biggest objects in the *world*. In the reproduction phase, on the other hand, the size of the *frame* is set equal to the size of the *world*.

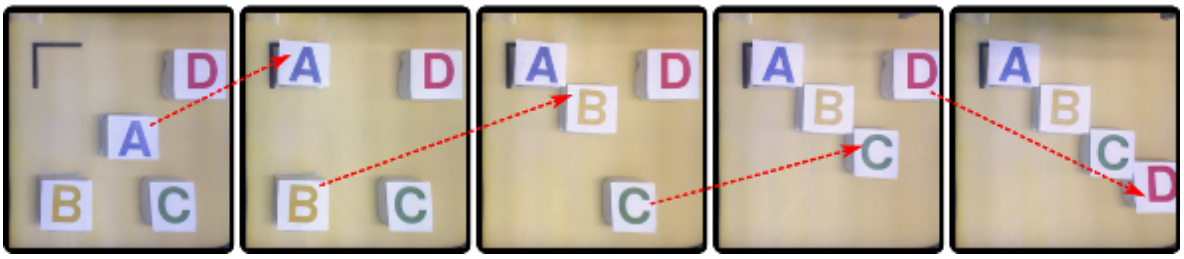
The vision sensor is mounted above the table facing the workspace. The resolution of the captured images are 1280×960 pixels. Although the trajectories are created in the end-effector space, the robot is controlled in the joint-space based on the inverse dynamics to avoid singularities. Also, during the reproduction phase, the controller keeps the orientation of the robot's hand (end-effector) perpendicular to the workspace plane, in order to facilitate the pick-and-place operation.

VSL relies on vision, which might be obstructed by other objects, by the tutor's body, or during the reproduction by the robot's arm. Therefore, for physical implementation of the VSL approach special care needs to be taken to avoid such obstructions.

Alphabet Ordering

In the first real-world VSL task, the *world* includes four cubic objects labeled with A, B, C, and D letters. Similar to the first simulated experiment the *world* also includes a fixed right angle baseline which is a landmark ($\mathcal{L}$). The goal is to reconfigure and sort the set of objects with respect to the baseline according to the demonstration. As reported in Table 4.1, this task emphasizes VSL's capability of relative positioning of an object with respect to other surrounding objects in the *world* (a visuospatial skill). This inherent capability of VSL is achieved through the use of visual *observations* which capture both the object of interest and its surrounding objects (i.e. its context). In addition, the baseline is provided to show the capability of absolute positioning of the VSL approach. It shows the fact that the robot can learn to attain absolute positioning of objects without any explicit *a priori* knowledge. Figure 4-16a shows the sequence of operations in the demonstration phase. Recording *pre-action* and *post-action observations*, the robot learns the sequence of operations. Figure 4-16b shows the sequence of operations produced by VSL starting from a novel *world* (i.e., new initial configuration) which is achieved by randomizing the objects in the *world*.

(a) The sequence of operations in the demonstration phase by the tutor

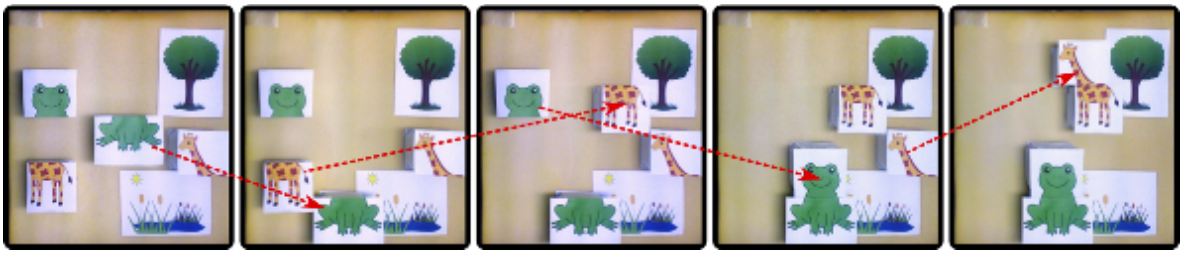


(b) The sequence of operations in the reproduction phase by the robot

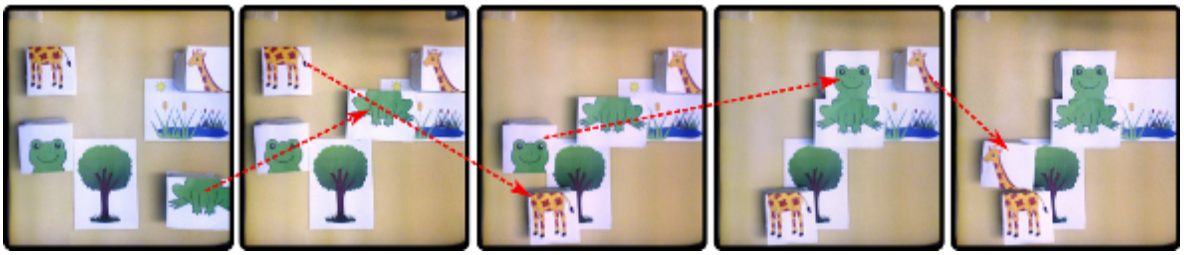
Figure 4-16: Alphabet ordering. The initial configurations of the objects in 4-16a and 4-16b are different. The red arrows show the pick-and-place operations. For more information, see Section 4.5.2.

Animal Puzzle

In the previous task, due to the absolute positioning capability of VSL, the final configuration of the objects in the reproduction and the demonstration phases are always the same. In this experiment, however, by removing the fixed baseline from the *world*, the final result can be a totally a new configuration of objects. The goal of this experiment is to show the VSL's capability of relative positioning which is reported in Table 4.1. In this VSL task, the *world* includes two sets of objects which complete a 'frog' and a 'giraffe' puzzle. There are two labels (i.e., landmarks) in the *world*, a 'pond' and a 'tree'. The goal is to assemble the 'frog' besides the 'pond' and the 'giraffe' besides the 'tree' label according to the demonstration. Figure 4-17a shows the sequence of the operations in the demonstration phase. To show the capability of generalization, the 'tree' and the 'pond' labels are randomly replaced by the tutor before the reproduction phase begins. Figure 4-17b shows the sequence of operations reproduced by VSL.



(a) The sequence of operations in the demonstration phase by the tutor



(b) The sequence of operations in the reproduction phase by the robot

Figure 4-17: Animal puzzle. The initial and the final configurations of the objects in the *world* are different in (a) and (b). The red arrows show the pick-and-place operations. For more information, see Section 4.5.2.

Tower of Hanoi

In this experiment, the Tower of Hanoi puzzle is performed again, this time in real-world. As mentioned before, this experiment demonstrates almost all capabilities of VSL comprising relative and absolute positioning, user intervention to modify the reproduction, and multiple operations performed on the same object. The sequence of operations in the reproduction phase is shown in Figure 4-18.

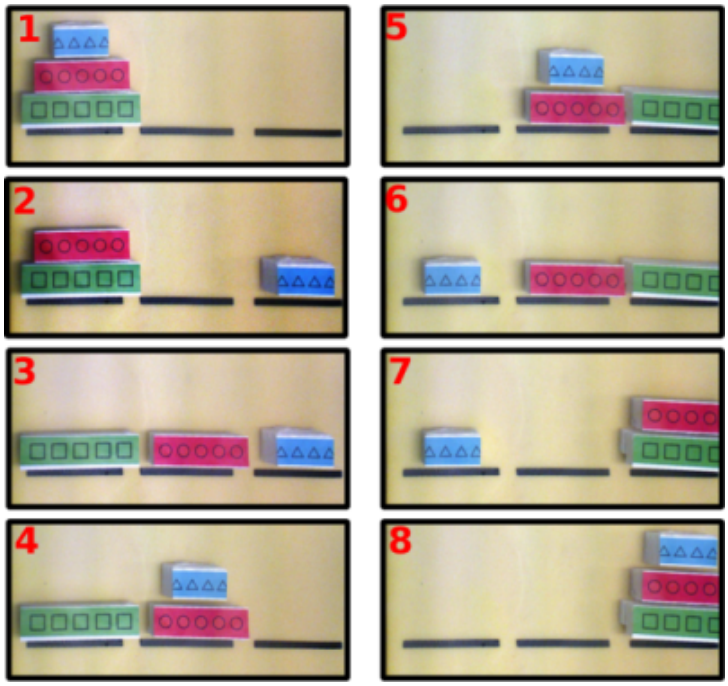


Figure 4-18: Tower of Hanoi. The sequence of operations in the reproduction phase by the robot. For more information, see Section 4.5.2.

Animals vs. Machines: A Classification Task

This interactive task demonstrates the VSL capability of classification of objects. The robot is provided with four objects, two 'animals' and two 'machines'. Also, two labeled bins are used in this experiment for classifying the objects. Similar to previous tasks, the objects, labels and bins are not initially known to the robot. In this task, firstly, all the objects are randomly placed in the *world*. The tutor randomly picks objects one by one and places them in the corresponding bins. In the reproduction phase, the tutor places one of the objects each time, not necessarily in a similar sequence with respect to the demonstration. The robot detects the object and classifies it. This is an interactive task between the human and the robot. The human tutor can modify the sequence of operations in the reproduction phase by presenting the objects to the robot in a different order with respect to the demonstration.

To achieve the mentioned capabilities, the algorithm is modified so that the robot doesn't follow the operations sequentially but searches in the *pre-action observation* dictionary to find the best matching *pre-action observation*. Then, it uses the selected *pre-*

action observation for the reproduction phase as before.

The sequence of operations in the demonstration phase are illustrated in Figure 4-19. Each column represents one pick-and-place operation. During each operation, the tutor picks an object and classifies it either as an 'animal' or as a 'machine'. In addition, the selected object in each operation is shown in the middle row.

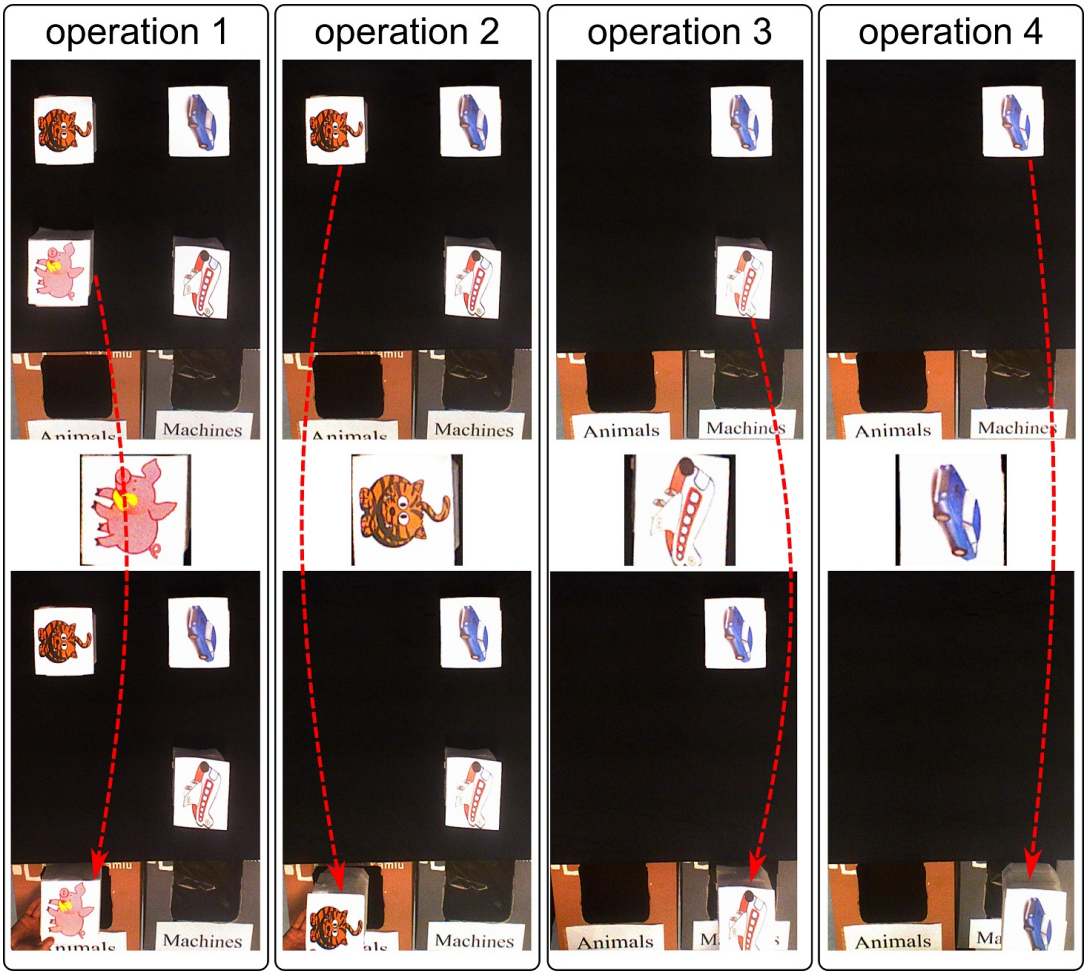


Figure 4-19: The sequence of operations in the demonstration phase. Each column represents one pick-and-place operation. In each operation, the tutor picks one object and classifies it either as an 'animal' or a 'machine'. The selected object in each operation is shown in the middle row. For more information, see Section 4.5.2.

Similar to the previous experiments, the match finding process is done using SIFT. Figure 4-20 shows two match finding processes in the reproduction phase.

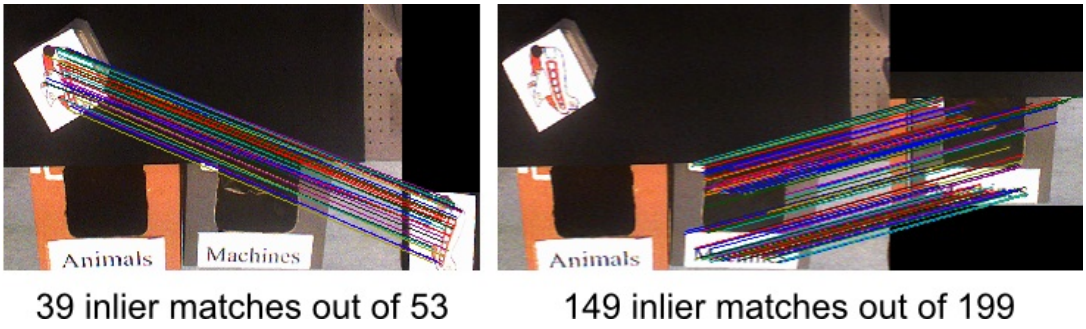


Figure 4-20: Two sets of match finding results in the reproduction phase are shown. For more information, see Section 4.5.2.

During the reproduction phase, initially there is no object on the workspace. The tutor places the objects one after another and the robot performs the classification task. Two operations during the reproduction phase are illustrated in Figure 4-21.

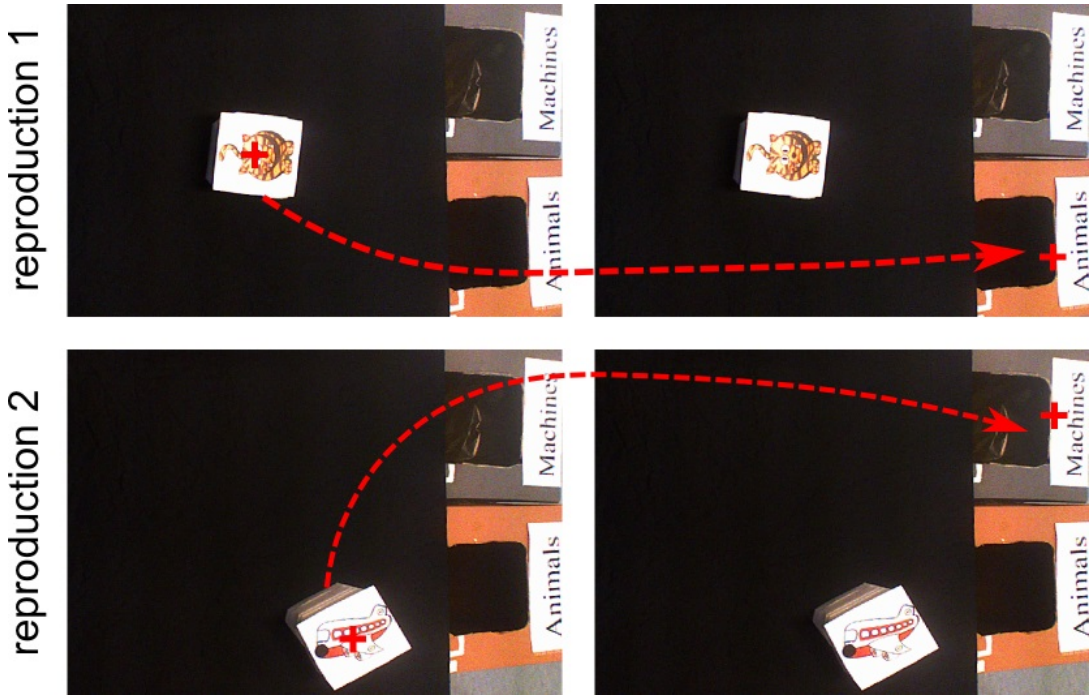


Figure 4-21: Two operations during the reproduction phase are shown. The red crosses on the objects and on the bins, show the detected positions for pick and place actions respectively. For more information, see Section 4.5.2.

Domino: A Turn-taking Task

The goal of this experiment is to show that VSL can deal with the tasks including the cognitive behaviour of turn-taking. In this VSL task, the *world* includes a set of objects

all of which are rectangular tiles. Each two pieces of the puzzle fit together to form an object (see Figure 4-22).

In this task, the tutor first demonstrates all the operations in the demonstration phase while the robot is observing the scene. In the reproduction phase, the tutor starts the game by placing the first object (or another) in a random place. The robot then takes the turn and finds and places the next matching domino piece. In this task, the modified algorithm from the previous task is utilized that searches through the *observation* dictionaries to find the corresponding *observation*. The tutor can also modify the sequence of operations in the reproduction phase by presenting the objects to the robot in a different order with respect to the demonstration. The sequence of reproduction are shown in Figure 4-22.

Figure 4-22: The sequence of reproduction performed by the robot and the tutor are shown for the turn-taking task of domino. For more information, see Section 4.5.2.

Results

Table 4.1 summarizes the main capabilities of VSL which are emphasized in each real-world experiment.

Table 4.1: Capabilities of VSL illustrated in each real-world experiment.

Capability / **Task**	Animal Puzzle	Alphabet Ordering	Tower of Hanoi	Animals vs. Machines	Domino
Relative positioning	✓	✓	✓	-	✓
Absolute positioning	-	✓	✓	-	-
Classification	-	-	-	✓	-
Turn-taking	-	-	✓	✓	✓
User intervention	-	-	✓	✓	✓

Table 4.2 lists the execution time for different steps of the implementation of the

VSL approach. The execution time for each item is averaged over different real-world experiments.

Table 4.2: Execution time for different steps of VSL

Steps	**Time (sec)**	**For each**
Homography	0.2-0.35	learning task
SIFT+RANSAC	18-26	operation (reproduction)
Image subtraction	0.09-0.11	comparison
Image thresholding	0.13-0.16	comparison
Trajectory generation	1.3-1.9	operation (reproduction)
Trajectory tracking	6.6-15	operation (reproduction)
Image rectification	0.3-0.6	image
Demonstration (calculation time)	3.8-5.6	operation
Reproduction	27.5-44.1	operation

In order to test the repeatability of VSL and to identify the possible factors of failure, the captured observations from the real world experiments were used while excluding the robot from the loop. All other parts of the loop were kept intact and each experiment was repeated three times. The result shows that less than 5% of pick-and-place operations failed. The main failure factor is the match finding error which can be resolved by adjusting the parameters of SIFT-RANSAC or using alternative match finding algorithms. The noise in the images and the occlusion of the objects can be listed as two other potential factors of failure. Despite the fact that VSL is scale-invariant, color-invariant, and view-invariant, it has some limitations. For instance, if the tutor accidentally moves one object while operating another, the algorithm may fail to find a pick/place position. One possible solution is to combine classification techniques together with the image subtraction and thresholding techniques to detect multi-object movements.

4.6 3D Visuospatial Skill Learning

With the advent of low-cost 3D sensors, real-time point cloud data has become easily available. Such advancements demand robot learning approaches to improve their capabilities by possessing and utilizing this new source of information. In this section, to improve the capabilities of VSL, the 2D VSL approach presented in Section 4.3 is extended to 3D space. Although the main structure of the algorithm remains unaltered, such extension requires more sophisticated image processing methods including 3D feature estimation, calculation of surface properties, dealing with noise, 3D match finding, and pose estimation.

The rest of the section is organized as follows. Point cloud processing methods used in the 3D VSL are mentioned in Section 4.6.1. Implementation of the 3D approach is compared with the implementation of the VSL in 2D in Section 4.6.2. Experimental results are reported in Section 4.6.3.

4.6.1 Point Cloud Processing

The 3D VSL approach utilizes point cloud processing methods in both demonstration and reproduction phases. In the demonstration phase, for each operation the algorithm requests data streams from the sensor and captures two point clouds, one *pre-action* and one *post-action* clouds.

Firstly, the captured raw point clouds are filtered using a pass-through filter to cut the values which are outside the workspace. This action reduces the number of points by removing unnecessary information (i.e. distant points) from the raw point clouds. The limits of filtering for each axis are derived from the workspace dimensions. Secondly, for each operation, from each pair of captured *pre-action* and *post-action* 3D point clouds, *pre-action* and *post-action observations* are generated. In this part of the process, a voxelizing approach is used for downsampling the point clouds. All the present points in a voxel will be approximated with their centroid. Voxelizing a point cloud increases the speed of further processing, nonetheless it decreases the resolution of the data. Therefore, the leaf size of the voxel grid should be selected appropriately. The points generated in

voxelization phase are stored into a *Kd*-tree structure. K Nearest-Neighbor search (KNN) is applied to extract a subtracted point cloud by finding correspondence between groups of points. The search precision (i.e. error bound) and the distance threshold are two parameters that should be set properly for the search algorithm. The result of applying the described process on a pair of captured *pre-action* and *post-action observations* are depicted in Figure 4-23. As mentioned before, during the demonstration phase two dictionary of *observations*, $\mathcal{O}^{pre}$ and $\mathcal{O}^{post}$, are captured and recorded.

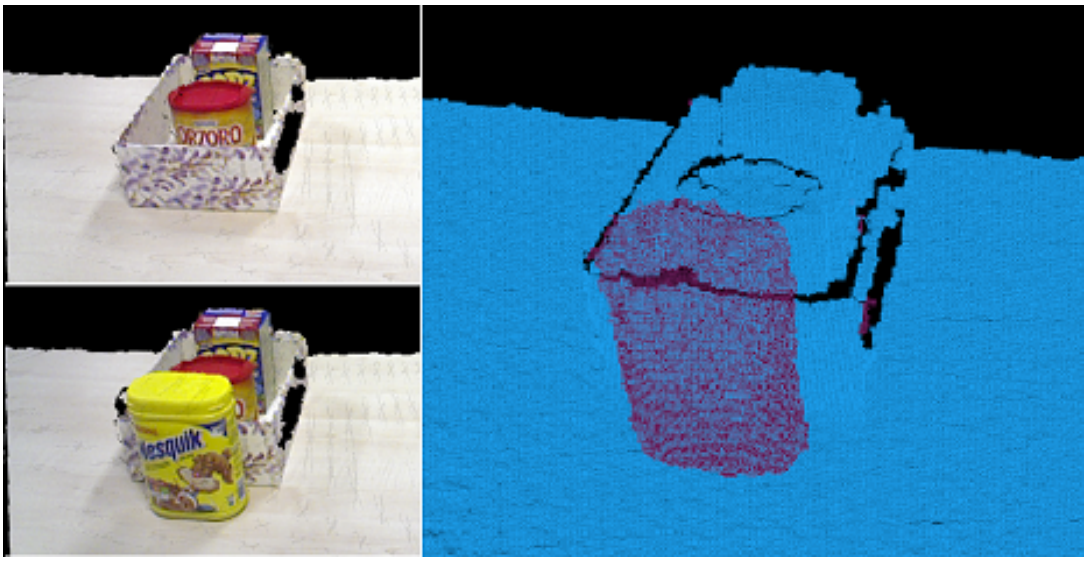
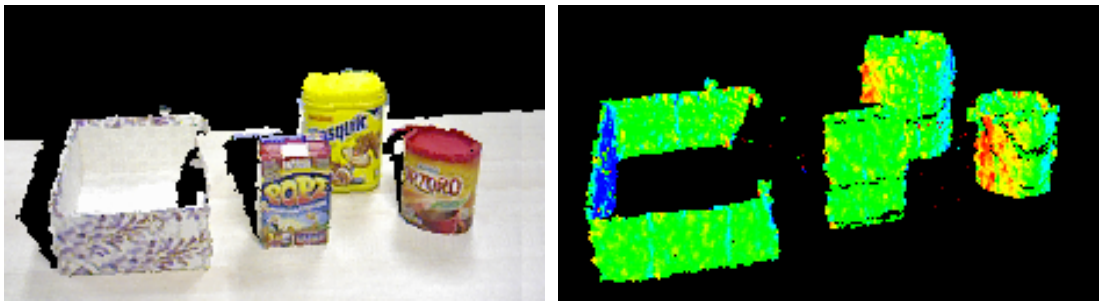


Figure 4-23: Result of a subtraction process between a pre-pick (top-left) and a pre-place (bottom-left) point clouds. The original clouds are voxelized at a resolution of 2 mm and the distance threshold for the KNN-search is set to 1 cm. For more information, see Section 4.6.1.

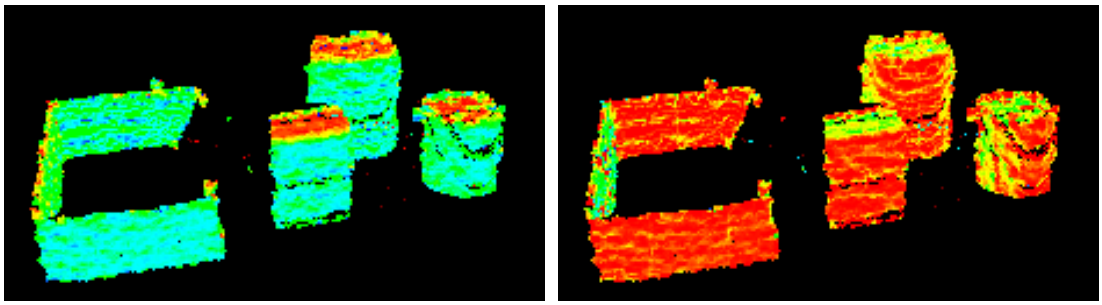
In the reproduction phase, for each operation the algorithm first captures a *world observation* ($\mathcal{W}_R$). Then, the corresponding recorded *observations* are loaded from the dictionaries and a metric is applied to find the best match between the new point cloud *observation* and the previously stored ones. In this phase, firstly the distant points are removed from the *world* and a voxelization filter is applied on the cloud. Surface normals are estimated for each cloud using a *Kd*-tree and search for neighboring points in a pre-defined radius. The surface normals are estimated by analysing eigenvectors and eigenvalues of a covariance matrix created from nearest neighbors search. The estimated surface normals are then used to find local features. There are many methods to represent

point features [168, 169]. An acceptable feature descriptor should be able to show the same characteristics in the presence of rigid transformation, different sampling density, and noise. In this section, Fast Point Feature Histogram (FPFH) proposed in [168] is utilized. FPFH encodes a point's k-neighborhood geometrical properties by generalizing the mean curvature around the point using a multi-dimensional histogram of values. FPFH is chosen because it is 6D pose-invariant and copes with varying sampling densities or noise levels, it depends on the quality of the surface normal estimations at each point. FPFH is a computationally efficient version of Point Feature Histogram (PFH).



(a) RGB-D

(b) Normal (X-axis)



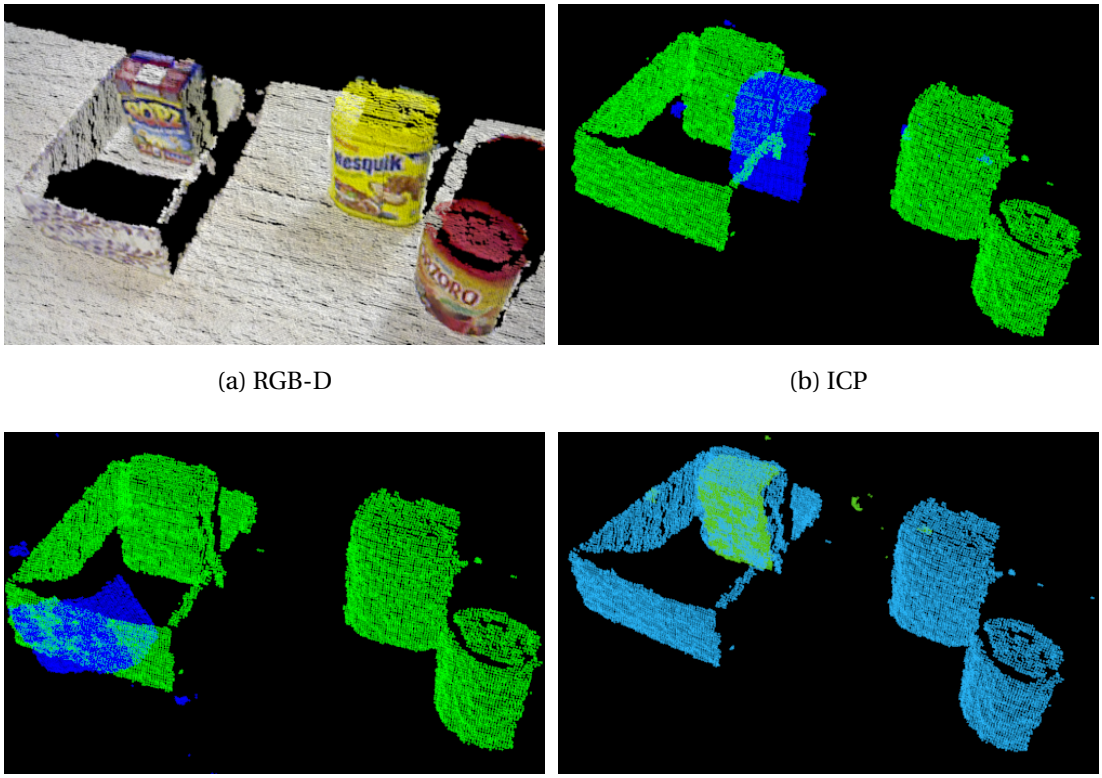
(c) Normal (Y-axis)

(d) Normal (Z-axis)

Figure 4-24: 4-24a illustrates the captured RGB-D cloud of a scene. Estimated normals in 3 axes are depicted in 4-24b,4-24c and 4-24d. For more information, see Section 4.6.1.

For the match finding and pose estimation process, three different algorithms consisting of Iterative Closest Point (ICP), Pre-rejective RANSAC [170], and SAmple Consensus Initial Alignment (SAC-IA) [168] are applied. ICP is an algorithm employed to minimize the difference between two clouds of points iteratively. Pre-rejective RANSAC is a modified RANSAC algorithm which outperforms the original algorithm by early elimination of bad pose hypothesis. SAC-IA is a new and fast sample consensus based

method which works by ranking correspondence candidates. Comparing the three algorithms in this case, the result suggests that pre-rejective RANSAC and ICP can find the best match correctly when the given object and the presented object in the scene are not partially occluded. Otherwise, these two methods usually fail. SAC-IA, on the other hand, finds the best match and estimates the pose efficiently. The result of a sample match finding process using the above-mentioned algorithms is shown in Figure 4-25. The estimated pose including the orientation and translation is used as the input for the trajectory module which is explained in the next section.



(a) RGB-D (b) ICP

(c) Pre-rejective RANSAC (d) SAC-IA

Figure 4-25: 4-25a illustrates the captured RGB-D cloud of a scene. The result of the match finding process using different methods are shown in 4-25b,4-25c and 4-25d. For more information, see Section 4.6.1.

4.6.2 2D vs. 3D VSL

The main differences for implementing VSL in 2D versus 3D are reported in Table 4.3. The table includes the techniques used in this thesis. However, the implementation of

VSL is not limited to these techniques and depending on the application, various image processing and point cloud processing techniques can be employed together with VSL.

4.6.3 Experimental Results of 3D VSL

Experimental Setup

In this section, the feasibility and capability of the proposed 3D VSL approach are experimentally validated. The presented algorithm is implemented in C++ and all the experiments were performed on a Linux PC with 8GB RAM and an Intel 3.20GHz CPU with 8 cores. As shown in Figure 4-26, the experimental setup for all the conducted experiments in this section consists of a torque-controlled 7 DoF Barrett WAM robotic arm equipped with 3-finger Barrett Hand, a tabletop working area, a set of objects, and an RGB-D sensor, namely, Asus Xtion Pro Live.

Figure 4-26: The experimental setup utilized for the conducted experiments in Section 4.6.3 using 3D VSL. For more information, see Section 4.6.3.

Table Cleaning

As shown in Figure 4-27, in the first 3D experiment, the *world* includes an empty box and a set of objects which are randomly placed in the workspace (the tabletop area). The

Table 4.3: The main differences between the 2D and 3D VSL approach.

	2D VSL	**3D VSL**
Source of information	2D Image	3D Point Cloud
Applicable for	3DoF Reconfiguration (X, Y, θ)	6DoF Reconfiguration (X, Y, Z, θ, ϕ, γ)
Transformation matrix	2D to 2D (3x3)	3D to 3D (4x4)
Estimating the observations in demonstration phase	Background Subtraction Thresholding	Voxelizing *Kd*-tree K Nearest-Neighbor search
Match finding in reproduction phase	2D Features 2D Feature Estimation 2D Metrics (SIFT) RANSAC for finding the rotation	3D Features 3D Feature Estimation Voxelizing Normal/Curvature Estimation 3D Metrics (ICP/RANSAC/SAC-IA)
Main assumptions	No overlap among objects Objects have the same height Objects are placed at the same level	Applicable for partially visible objects Objects can have different heights Objects can be placed in different levels

goal is to clean up the table by picking each object and placing two of them inside the empty box and the final one in front of the box according to the demonstration. The tutor demonstrates the task and the *observations* are captured by the sensor. The robot learns the task through VSL and is capable of reproducing it. Before the reproduction phase starts, the objects are reshuffled randomly in the workspace. Figure 4-28 shows the sequence of operations reproduced by the robot. Although in some operations the objects are occluded by the front wall of the box, the robot can accomplish the task successfully.

Figure 4-27: The set of objects used in the table cleaning task. For more information, see Section 4.6.3.

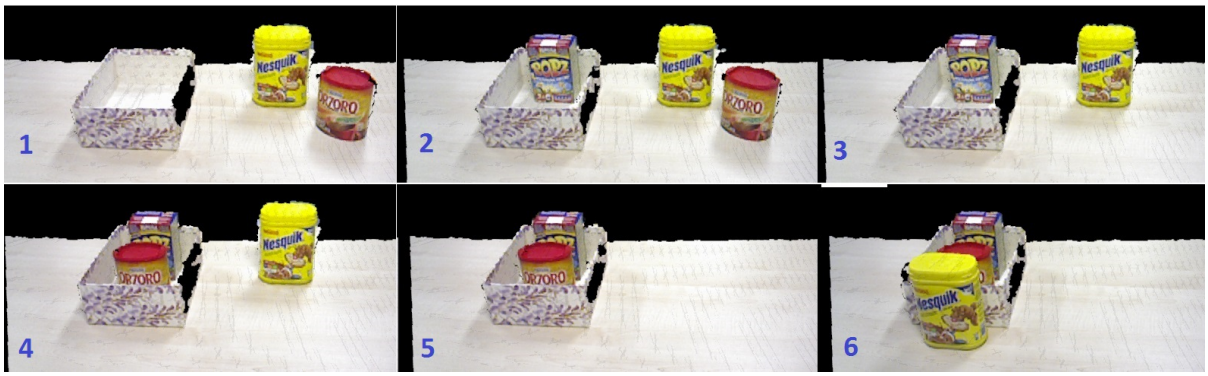


Figure 4-28: The sequence of operations reproduced by the robot for the table cleaning task. For more information, see Section 4.6.3.

Piling Objects

In the next 3D pick-and-place task, the set of objects are randomly placed on the table (the *world*). Additionally, in this experiment, there are two boxes (i.e. landmarks) in the workspace which are considered as static parts of the world and cannot be reconfigured

by the robot. However, these two boxes are repositioned before starting the reproduction phase by the tutor. The role of these boxes is similar to the role of the labels used in the 'animal puzzle' experiment in Section 4.5.2. In the first phase, the tutor starts piling the objects on top of the two boxes. The sequence of operations in the first phase are captured and together with the initial and final configurations of the objects are shown in Figure 4-29. After the learning phase is finished, the robot can reproduce the piling task in the workspace even if the objects and the boxes are randomly replaced. The sequence of operations in the reproduction phase together with the initial and final configurations of the objects are shown in Figure 4-30.

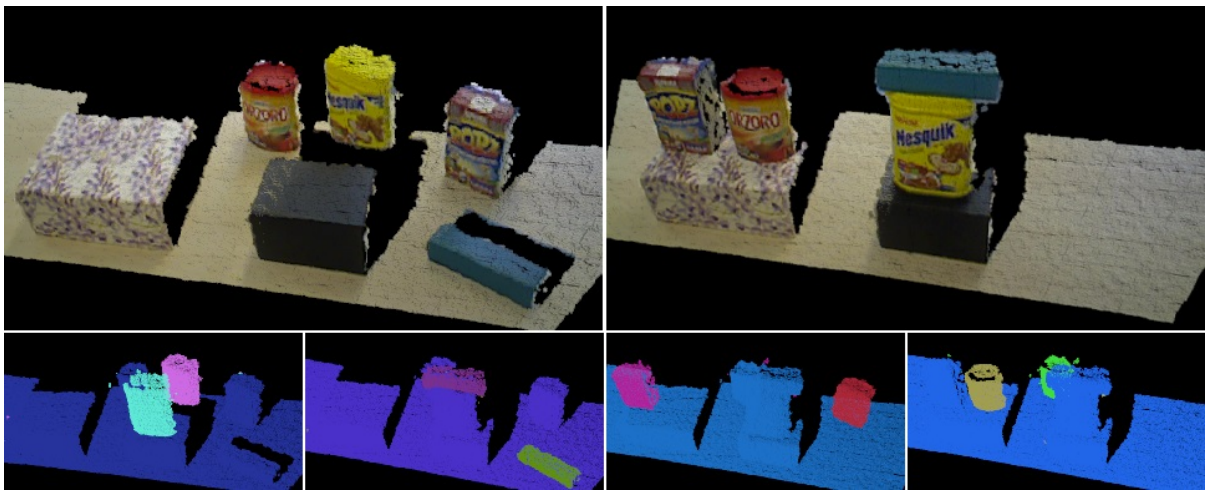


Figure 4-29: Demonstration of the piling task. The initial and final configurations of the objects in the world are shown in top (RGB-D). The bottom row, from left to right, shows the sequence of operations by the tutor. For more information, see Section 4.6.3.

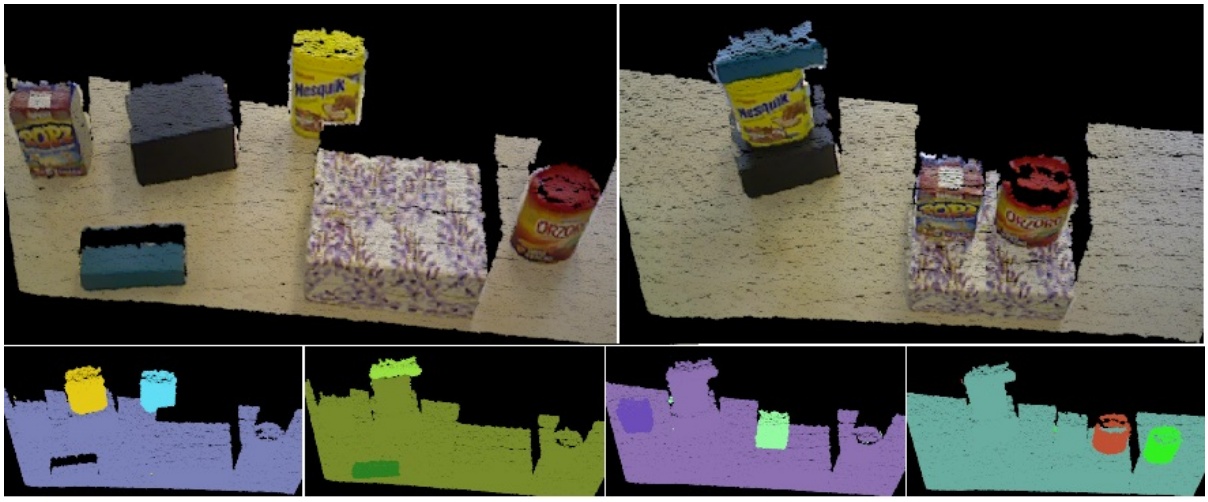


Figure 4-30: Reproduction of the piling task. The initial and final configurations of the objects in the world are shown in top (RGB-D). The bottom row, from left to right, shows the sequence of operations by the robot. For more information, see Section 4.6.3.

Results

3D VSL utilizes *pre-* and *post-action observations* extracted from a single demonstration. These *observations* are point cloud data which are actually 2.5D and are acquired from a single viewpoint. Since there is not *a priori* knowledge about the objects, if they are reconfigured in the *world*, they might not be detected and recognized successfully during the reproduction phase. In other words, if an object is observed in the demonstration phase, there is no guaranty that it can be detected in the reproduction phase while observed from a new point of view. This is due to the objects transformation before the reproduction phase. For instance, if the object was observed from a point of view and then it was flipped before the reproduction phase begins. The object might not be detected by the approach. The reason is that there is no enough data for the match finding phase. Thus, implementing a robust 3D VSL, requires more robust features applicable for both textured and textureless objects and modifications in the point cloud processing part of the algorithm. However, the main steps of the algorithm remain unaltered. Since image processing and point cloud processing are not the focus of this thesis, in this section a number of possible solutions are mentioned. One solution is to extract a set of features for each object from different viewpoints and produce a database for the existing objects in

advance. The database is used during the match finding process. This method has been employed in some recent researches [145, 153]. Another solution is to train a classifier for object recognition in the workspace [154].

4.7 Learning Symbolic Representations of Actions using VSL

Visuospatial Skill Learning (VSL) enables a robot to identify the spatial relationship between objects and learn a sequence of actions for achieving the goal of the task. VSL is capable of learning and generalizing different skills such as object reconfiguration, classification, and turn-taking interactions from a single demonstration [129, 130]. One of the shortcomings of VSL is that the primitive actions such as pick, place and reach, have to be manually programmed into the robot. For instance, in Section 4.4, a simple trajectory generation module was devised that is able to produce trajectories for pick and place primitive actions [129, 130]. This issue can be addressed by combining VSL with a trajectory-based learning approach such as imitation learning. The obtained approach can learn the primitive actions of the task using conventional learning from demonstration strategy while it learns the policy (e.g., sequence of actions), preconditions and effects of the actions using VSL.

Although the original VSL has its own internal planner, in order to utilize the advantages of standard symbolic planners, the original planner can be replaced with an already available action-level symbolic planner, namely SGPlan [171]. In order to ground the actions, the symbolic actions are defined in the planner and VSL maps identified preconditions and effects in a formalism suitable to be used at the symbolic level. The combination of high-level planning and low-level control first employed in the work by Nilsson [172]. However, such combination is immensely difficult because of the effort required to construct and interface matching high-level reasoning and low-level control components. There are few approaches in which a high-level symbolic representation is formed using low-level description of actions. The method proposed in [173] finds

relational, probabilistic STRIPS operators by searching for symbol groundings that maximize a metric balancing transition predictability with model size. The work by Konidaris *et al.* avoids this search by directly deriving the necessary symbol grounding from the actions [174].

The integration of VSL and a symbolic planner is brings significant advantages over using VSL separately. Learning preconditions and effects of the actions in VSL approach and making symbolic plans based on the learned knowledge deals with the long-standing important problem of Artificial Intelligence, namely Symbol Grounding. In the experiments it has been shown that the symbols and rules are grounded through learning in the problem domain.

In this section, a new robot learning framework is proposed by integrating a sensorimotor learning framework (imitation learning), a visual learning approach (VSL), and a symbolic-level action representation and planning layer. The capabilities and performance of the proposed framework is validated using real-world experiments with an iCub robot.

The most similar work to this integrated approach is that by Ekvall *et al.*, [145]. In their work, a task planning approach is used in combination with robot learning from demonstration. The robot generates states and identifies constraints of a task incrementally according to the order of the action execution. Imitation learning is employed to teach the task to the robot. The constraints are identified in two ways: either the tutor instructs the system or the constraints are considered by merging of multiple observations. Differently from the proposed approach, the objects are first modeled geometrically and a set of SIFT features for each object is extracted in off-line mode and used during the learning phase. They build a planning scenario that is bounded to the objects considered in the learning phase, whereas in the proposed approach actions whose validity is not limited to the actual learning scenario are defined.

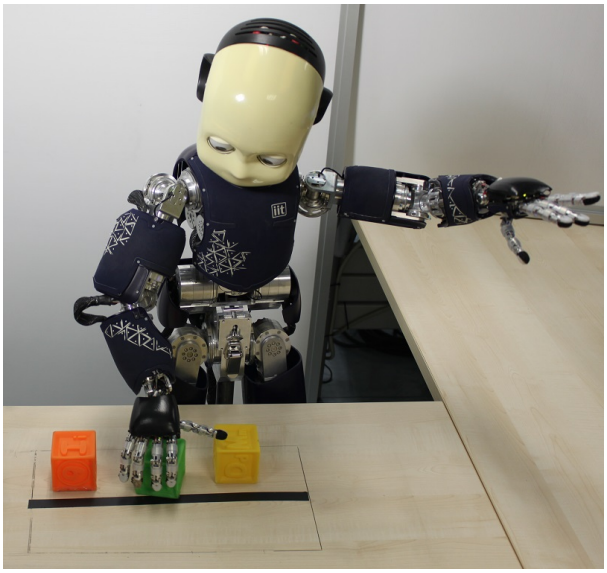


Figure 4-31: The iCub robot interacting with objects.

4.7.1 Overview of Symbolic VSL

The proposed learning approach consists of three layers:

- **Imitation Learning (IL)**, which is suitable for teaching trajectory-based skills (i.e. actions) to the robot [5]. For instance, pouring and reaching.
- **Visuospatial Skill Learning (VSL)**, introduced in Section 4.3. VSL is a goal-based approach based on visual perception [129, 130]. It captures the spatial relationship between an object and its surrounding objects and extracts a sequence of actions to reach the goal of the task.
- **Symbolic Planning (SP)**, which uses a symbolic representation of actions, to plan or replan the execution of the task.

In the proposed framework in this section, IL is employed to teach sensorimotor skills to the robot (e.g., pull and push). The learned skills are stored in a primitive action library, which is accessible by the planner. VSL captures the object's context for each demonstrated action. This context is the basis of the visuospatial representation and encodes implicitly the relative positioning of the object with respect to multiple other

objects simultaneously. By capturing the spatial relationship among objects, VSL extracts a sequence of actions to reach the goal of the task. In addition, it is capable of identifying the preconditions and effects for each action. As previously stated, VSL is internally equipped with a simple planner which has adequate capabilities as shown in Sections 4.5 and 4.6.3 and also in [129, 130].

In order to employ the advantages of symbolic-level action planners, PDDL [5] 3.0 compliant planner is integrated with IL and VSL to represent a general purpose representation of a planning domain. In the proposed framework, VSL identifies action preconditions and effects to modify their formal PDDL definition (i.e., the planning domain). Once formal action definitions are obtained, it is possible to exploit the planning domain to reason upon scenarios that are not strictly related to what has been learned, e.g., including different object configuration, a varied number of objects or different objects at all. One of the main advantages of the proposed integrated framework is that it extracts and utilizes multi-modal information from demonstrations including both the sensorimotor information and visual perception, which is used to build symbolic representation structures for actions. A high-level flow diagram of the proposed framework including the three layers together with their input and output can be seen in Figure 4-32.

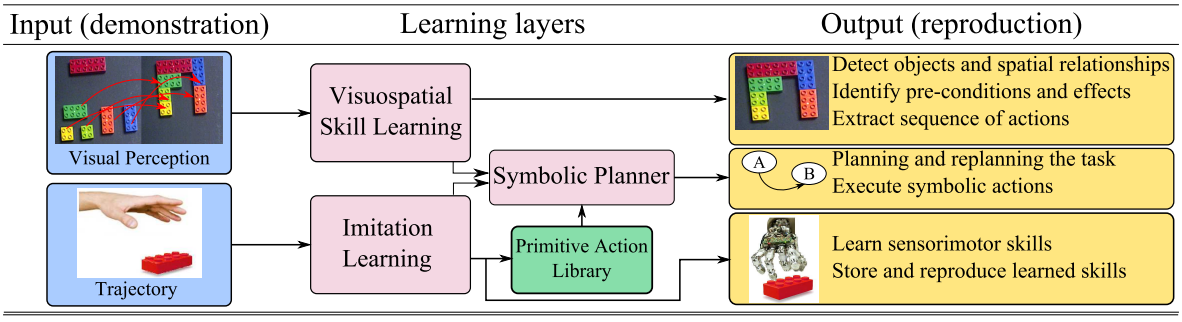


Figure 4-32: A flow diagram illustrating the proposed integrated approach comprising three main layers. The robot learns to reproduce primitive actions through imitation learning. It also learns the sequence of actions and identifies the constraints of the task using VSL. The symbolic planner is employed to solve new symbolic tasks and execute the plans. For more information, see Section 4.7.1.

[5] PDDL stands for *Planning and Domain Definition Language* which serves as the input format for most general-purpose planners [175].

4.7.2 Methodology

The robot learns the primitive trajectory-based actions through IL. The layer requires a set of demonstrations for each action, from which a set of attractors are learned and used to reproduce the action. The learned skills are stored in a primitive action library. In addition, the robot learns the sequence of actions through VSL using visual perception. As described in Section 4.3, the robot records a set of visual observations during the demonstration phase. It can then identify and reproduce the sequence of actions to reach the desired goal of the demonstrated task using spatial relationship between the objects. As shown in Figure 4-32, both IL and VSL observe the same demonstrations (not necessarily at the same time), but utilize different types of data. IL uses sensorimotor information (i.e., trajectories), whereas VSL uses visual data. VSL also extracts the preconditions and effects of each action, thereby updating the PDDL description of each learned action, as well as (if required) the demonstrated problem, including the strict sequence of shown actions. Obviously enough, if the knowledge about such a sequence is not considered in subsequent planning steps, the planner is free to choose any suitable course of actions to solve a given planning problem. Once a symbolic representation of actions is obtained, SP can be used to solve new problems in the learned domain, as well as reproduce plans learned using VSL.

4.7.3 Learning Sensorimotor Skills by Imitation Learning

IL enables robots to learn and reproduce trajectory-based skills from a set of demonstrations through kinesthetic teaching [32]. In the proposed framework, IL is utilized to teach primitive actions (i.e., pull and push) to the robot. In particular, Dynamic Movement Primitives (DMP) which are designed for modeling attractor behaviors of autonomous nonlinear dynamical systems are utilized [116].

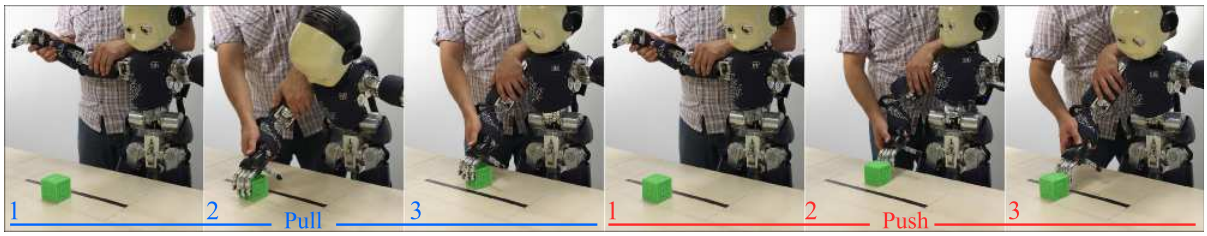


Figure 4-33: The tutor is teaching the primitive actions including pull and push to the robot using kinesthetic teaching. For more information, see Section 4.7.3.

In order to create a pattern for each action, multiple desired trajectories in terms of position, velocity and acceleration have to be demonstrated and recorded by a human operator, in the form of a vector $[y_{demo}(t), \dot{y}_{demo}(t), \ddot{y}_{demo}(t)]$, where $t \in [1,...,P]$ and P is the number of datapoints. A controller converts desired position, velocity, and acceleration, y, $\dot{y}$,and $\ddot{y}$, into motor commands. DMP employs a damped spring model that can be modulated with nonlinear terms such that it achieves a desired attractor behavior. The transformation system which represents dynamics of a damped spring system is as follows:

$$\begin{aligned} \tau \dot{z} &= \alpha_z(\beta_z(g-y)-z)+f+C_f \\ \tau \dot{y} &= z \end{aligned} \tag{4.10}$$

where y and z are the position and the velocity respectively, τ is a time constant and α_z and β_z are positive constants. With $\beta_z = \alpha_z/4$, y monotonically converges toward the goal g. f is the forcing term and C_f is an application dependent coupling term. Trajectories are recorded independently of time. Instead, the canonical system, is defined as:

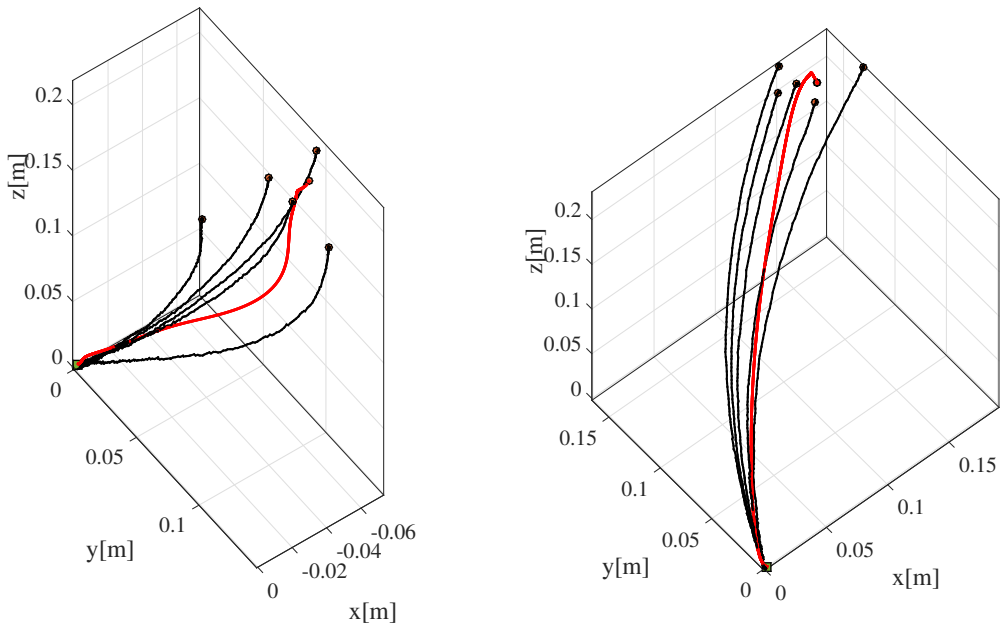
$$\tau \dot{x} = -\alpha_x x + C_c \tag{4.11}$$

where α_x is a constant and C_c is an application dependent coupling term. x is a phase variable, where $x = 1$ indicates the start of the time and the x close to zero means that the goal g has been achieved. Starting from some arbitrarily chosen initial state x_0 such as $x_0 = 1$ the state x converges monotonically to zero. f, in (4.10) is chosen as follows:

$$f(x) = \frac{\sum_{i=1}^{N} \psi_i(x)\omega_i}{\sum_{i=1}^{N} \psi_i(x)} x(g - y_0) \tag{4.12}$$

where ψ_i $i = 1, \ldots, N$ are fixed exponential basis functions, $\psi_i(x) = \exp(-h_i(x - c_i)^2)$, where σ_i and c_i are the width and centers of the basis functions respectively, and w_i are weights. y_0 is the initial state $y_0 = y(t = 0)$. The parameter g is the target coinciding the end of the movement $g = y_{demo}(t = P)$, $y_0 = y_{demo}(t = 0)$. The parameter τ must be adjusted to the duration of the demonstration. The learning process of the parameters w_i is accomplished with a locally weighted regression method, because it is very fast and each kernel learns independent of others [116].

In the considered scenario, two primitive actions, namely pull and push are used. For executing the push action on an object, firstly, the robot needs to reach a location just in front of the object. Also for executing the pulling action the robot needs to reach a location just beyond the object. As the first step, the tutor demonstrates each primitive action while holding and moving the robot's hand. Using the recorded sets of trajectories and applying DMP, the robot learns a set of attractors by which it can reproduce the action starting from a different pose towards a target. To achieve this goal, the *pull* action is decomposed into *reach after* and *move backward* sub-actions and also the *push* action is decomposed into *reach before* and *move forward* sub-actions. The recorded sets of demonstrations for both *reach before* and *reach after* sub-actions are depicted as black curves in Figure 4-34. Each red trajectory in Figure 4-34 illustrates a reproduction. In both Figures the goal, (i.e., the object), is shown in green. Finally, the learned primitive actions are stored in the primitive action library. Later, the planner connects the symbolic actions to the library of actions. The *move forward* and *move backward* sub-actions are straight line trajectories that can be learned or implemented directly into the trajectory generation module.



(a) reach-before sub-action.

(b) reach-after sub-action.

Figure 4-34: The recorded set of demonstrations for two sub-actions are shown in black. The reproduced trajectories from an arbitrary initial position towards the target (the green square) are shown in red. For more information, see Section 4.7.3.

4.7.4 Learning Action Sequences by VSL

At this stage, a library of primitive actions has been built. The robot learns the sequence of actions required to achieve the goal of the demonstrated task through VSL. It also identifies the spatial relationship between objects by observing demonstrations.

At the beginning of the demonstration, the objects are randomly placed in the *world* ($\mathcal{W}_D$). In case that, any absolute landmark is being used in the demonstration (e.g., borderline in the experiments in Section 4.7.7), the robot should be able to detect it in the *world* (line 2). During the demonstration, VSL captures one *pre-action observation* ($\mathcal{O}^{pre}$) and one *post-action observation* ($\mathcal{O}^{post}$) for each operation executed by the tutor using the specified *frame* ($\mathcal{F}_D$) (lines 5,6). For each detected object in the *world*, VSL creates a symbolic representation ($\mathcal{B}$) and extracts a feature vector (ϕ). The symbolic object is used by the symbolic planner and the extracted features are used by VSL for detecting the object during the reproduction phase. VSL also extracts the pose of the

object before and after action execution ($\mathcal{P}^{pre}$,$\mathcal{P}^{post}$). The pose vectors together with the detected landmark ($\mathcal{L}$) are used to identify preconditions and effects of the executed action through spatial reasoning (line 8). These predicates are then utilized to identify the executed action from the action set $\mathcal{A}$ (line 9). The sequence of identified actions are then stored in a policy vector Π. In this framework, the symbolic planner, utilizes the output of VSL to reproduce the task properly. Furthermore, as shown in Section 4.5, the second part of the algorithm is able to execute the learned sequence of actions independently. In such case, VSL observes the new *world* ($\mathcal{W}_R$) in which the objects are replaced randomly. Comparing the recorded *pre-* and *post-action observations*, VSL detects the best matches for each object and executes the actions from the learned policy.

4.7.5 Generalization of Learned Actions as Symbolic Action Models

In order to use the learned primitive actions for general-purpose task planning, the actions need to be represented as a symbolic, discrete, scenario-independent form. To this aim, a symbolic representation for both the pull and push actions based on the PDDL 3.0 formalism is defined. However, it is necessary to learn preconditions and effects for each action in the primitive action library. The sensorimotor information in the skill learning process is exploited to derive symbolic predicates, instead of defining them manually. So the robot first perceives the initial workspace in which one object is placed in `far`. Then the robot is asked to execute the pull action, and after the action is finished, the robot perceives the workspace again. The *pre-action* and *post-action observations* are shown in Figure 4-35. The robot uses VSL to extract the preconditions and effects of the pull action. The same steps are repeated for the push action. The domain file in the PDDL is updated automatically by applying this procedure. The preconditions and effects for the pull and push actions are reported in table 4.4.

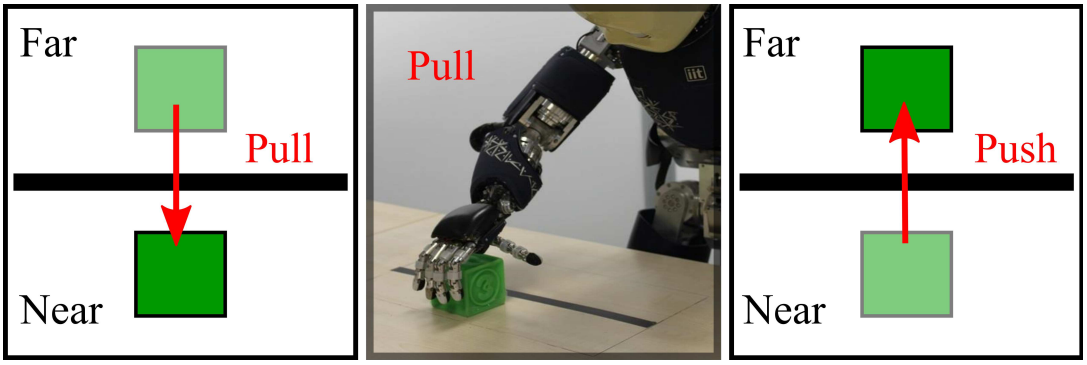


Figure 4-35: By extracting the preconditions and effects of the primitive actions while executing them, the robot generalizes the learned sensorimotor skills as symbolic actions. For more information, see Section 4.7.4.

Table 4.4: The preconditions and effects extracted by VSL and written to the planner's domain file. For more information, see Section 4.7.4.

Action	**push**	**pull**
precondition	`¬ (far ?b)`	`(far ?b)`
effect	`(far ?b)`	`¬ (far ?b)`

4.7.6 Implementation for Learning Symbolic Representation

After the VSL model has been identified, it is possible to generalize each action as well as the demonstrated action sequences, using a PDDL 3.0 compliant formalism. It is noteworthy that in the present work symbolic knowledge has not been bootstrapped from scratch. On the contrary, starting from the knowledge of the performed action types during the VSL demonstration, symbolic-level knowledge is updated with two kinds of constraints. The former class includes constraints (in the form of PDDL predicates) related to preconditions and effects. This is done by mapping the elements of observation dictionaries O^{pre} and O^{post} to relevant predicates. In the considered scenario, one predicate is enough to characterize push and pull actions, namely `(far ?b)`, where `far` is the predicate name and `?b` is a variable that can be grounded with a specific object that can be pushed and pulled. Specifically, `push(?b)` is an action that makes the predicate truth value switching from `¬(far ?b)` to `(far ?b)`, whereas the opposite holds for `pull(?b)`. Visual information about the location of objects in the *World* is mapped to the proper

truth value of the `(far ?b)` predicate. The execution of a `push` demonstration on a green cube (as processed by VSL) allows the system to understand that before the action $\neg$`(far` B_{green}`)` holds, after the action `(far` B_{green}`)` holds, and the two predicates contribute to the `:precondition` and `:effect` lists of a `push` action defined in PDDL.

The second class includes constraints (in the form of PDDL predicates) related to the trajectory of the plan that is executed in the demonstration. This can be done analyzing the sequence of actions as represented in the VSL model (i.e., the implicit chain defined by predicates in O^{post} and O^{pre} observation dictionaries). In PDDL 3.0, it is possible to define additional constraints, which implicitly limit the search space when the planner attempts to find a solution to the planning problem. In particular, given a specific planning domain (in the form of push and pull actions, as obtained by reasoning on the VSL model), the `problem` can be constrained to force the planner to follow a specific trajectory in the solution space. One possible constraint is to impose, for instance, the sequence of satisfiability of predicate truth values. As an example, let us assume the VSL model represents the fact that B_{green} must be always pushed after B_{blue}. According to our formal definition, this implies that the predicate `(far` B_{green}`)` must hold sometime after the predicate `(far` B_{blue}`)` holds. To encode this constraint in PDDL formalism, it is sufficient to use the constraint `(somewhere-after (far` B_{green}`) (far` B_{blue}`))`. If the constraint is used in the planning problem, the planner attempts to find a plan where this sequence is preserved, thereby executing `push(`B_{green}`)` strictly after `push(`B_{green}`)`. Finally, it is noteworthy that such a constraint is removed, the planner is free to choose any suitable sequence of actions to be executed (independently of the demonstration), provided that the goal state is reached.

4.7.7 Experimental Results of Symbolic VSL

Experimental Setup

In order to evaluate the capabilities and performance of the proposed framework, a set of real-world experiments are performed with an iCub humanoid robot[6]. As shown in

[6]A video of the real-world experiments reported in this section is available online at my personal website http://www.ahmadzadeh.info/videos.

Figure 4-31, the robot is standing in front of a tabletop including some polyhedral objects. All the objects have equivalent size but different colors and textures. Firstly, the tutor teaches the primitive actions to the robot using imitation learning. The primitive actions for these experiments include push and pull movements. The tutor performs a set of demonstrations for each action by holding and moving the robot's hand while the robot is perceiving the workspace (see Figure 4-33). Using the recorded set of demonstrations, the robot learns to reproduce each action from different initial position of the hand towards a desired object. The learned primitive actions are stored in the primitive action library.

To introduce the concept of `far` and `near` in the experiments, a black line, which is a borderline, is placed in the middle of the robot's workspace. The robot can detect the line using conventional edge detection and thresholding techniques. In addition, the size of the *frames* $(\mathcal{F}_D, \mathcal{F}_R)$ are defined equal to the size of the *world*.

During the demonstration phase, for each operation, the tutor removes the object from the *world*, and brings it back after an *observation* has been captured. Figure 4-36 shows some instances of image processing results from the conducted experiments in this section. Since image processing is not the focus of this thesis, a number of simple image processing techniques have been used for this purpose. In Section 4.4, the object detection is done using background subtraction technique and it has been shown the object identification process can be done using 2D and 3D match finding techniques (e.g., SIFT). In this section, however, blob detection and filtering by color on plane techniques are utilized by considering this fact that the iCub robot is capable of detecting the height of the table at the beginning of the experiment by touching it. To detect the object which is moved and the place that the object has been moved to, background subtraction is used. In both cases, the center of the object and the area are calculated and are shown with a red marker in Figure 4-36.

In addition, a simple (boolean) spatial reasoning has been applied to the VSL algorithm that utilizes the detected pose of the object and the detected borderline, and then decides that the object is `far` or `near`. However, to extract more sophisticated constraints, qualitative spatial reasoning languages such as region connection calculus [176] can be employed.

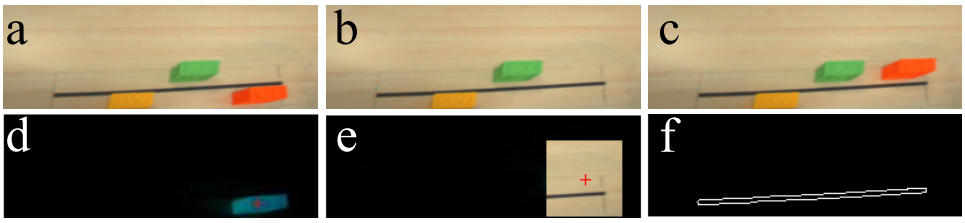


Figure 4-36: (a) and (b) are rectified pre-action and post-action observations; (c) is a rectified pre-action observation for the next operation; (d) is obtained by background subtraction between (a) and (b); (e) is obtained by background subtraction between (b) and (c); (f) is the result of detecting the border line for spatial reasoning. For more information, see Section 4.7.7.

A Simple VSL Task

In order to evaluate the obtained connection between the symbolic actions and their equivalent sensorimotor skills the following experiment has been conducted. Initially, there are three objects on the table. The tutor provides the robot with a sequence of actions depicted in Figure 4-37. The robot perceives the demonstration, records the observations and extracts the sequence of actions using VSL. In the reproduction phase, the objects are reshuffled in the workspace. For the reproduction of the task, instead of SP (Symbolic Planner), the internal planner of VSL is exploited. Starting from a different initial configuration of the objects in the workspace, VSL is capable of extracting the ordered sequence of actions and reproduce the same task effortlessly. As shown in Figure 4-38, the robot achieves the goal of the task using VSL. This experiment shows that the robot can relate the learned primitive actions including push and pull, with their preconditions.

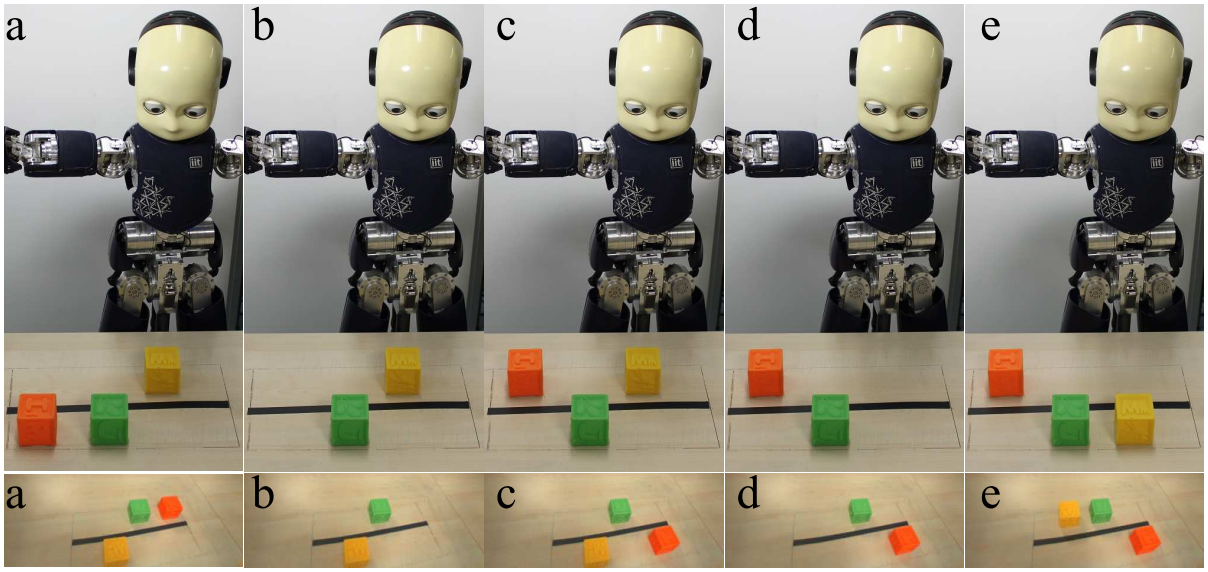


Figure 4-37: The set of demonstrations by the tutor for a simple VSL task. The bottom row shows the workspace from the robot's point of view. For more information, see Section 4.7.7.

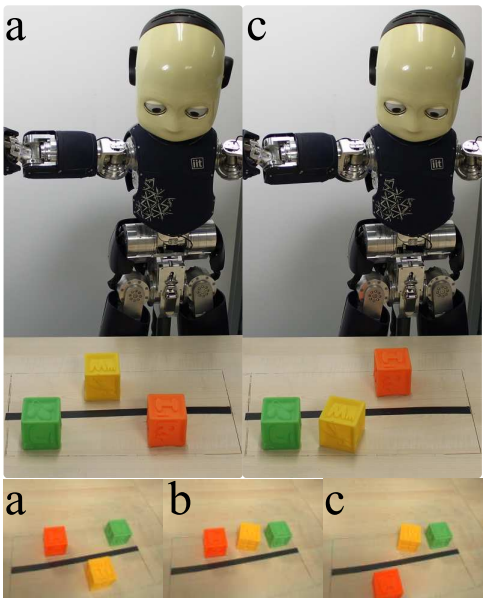


Figure 4-38: Starting from a new configuration, the robot reproduces the learned task using VSL. (a) and (c) show initial and final configurations. The bottom row shows the workspace from the robot's point of view. For more information, see Section 4.7.7.

Keeping All Objects Near

In the previous experiment, it has been shown that the internal planner of VSL is capable of reproducing the task. However, SP is more capable in generalizing the learned skill. In

this experiment, the robot learns a simple game including an implicit rule: *keep all objects near*. For each object, two properties are defined: color and position. SP provides the proposed approach with the capability of generalizing over the number and even shape of the objects. In order to utilize this capability, the tutor performs a set of demonstrations to eliminate the effect of color implying that any object independent of its color should not be `far` from the robot. Three sets of demonstrations are performed by the tutor which are shown in Figure 4-39. Each demonstration starts with a different initial configuration of objects. The performed demonstrations imply the following rules:

$$\begin{aligned} r_1^1 &: \langle B_{green} \rightarrow \texttt{near} \rangle \text{ IF } \langle B_{orange}\ is\ \texttt{far} \rangle \\ r_1^2 &: \langle B_{orange} \rightarrow \texttt{near} \rangle \text{ IF } \langle B_{green}\ is\ \texttt{near} \rangle \\ r_2^1 &: \langle B_{green} \rightarrow \texttt{near} \rangle \text{ IF } \langle B_{orange}\ is\ \texttt{near} \rangle \\ r_3^1 &: \langle B_{orange} \rightarrow \texttt{near} \rangle \text{ IF } \langle B_{green}\ is\ \texttt{far} \rangle \end{aligned} \tag{4.13}$$

where r_i^j indicates the j^{th} rule extracted from the i^{th} demonstration. The first demonstration includes two rules r_1^1, r_1^2 because two actions are executed. It can be seen that the right parts of r_1^1 and r_2^1 and the right parts of r_1^2 and r_3^1 eliminate each other. Also the color attribute on the left side of the rules eliminates the effect of the color on action execution. During the demonstration, for each object, the robot extracts the spatial relationships between the objects and the borderline, and decides if the object is `far` or `near` (i.e., `not far`). It then applies the extracted precondition of $r_1^1 : \langle B_? \rightarrow \texttt{near} \rangle$ to that object. Afterwards, the robot is capable of performing the learned skill on a various number of objects and even unknown objects with different colors.

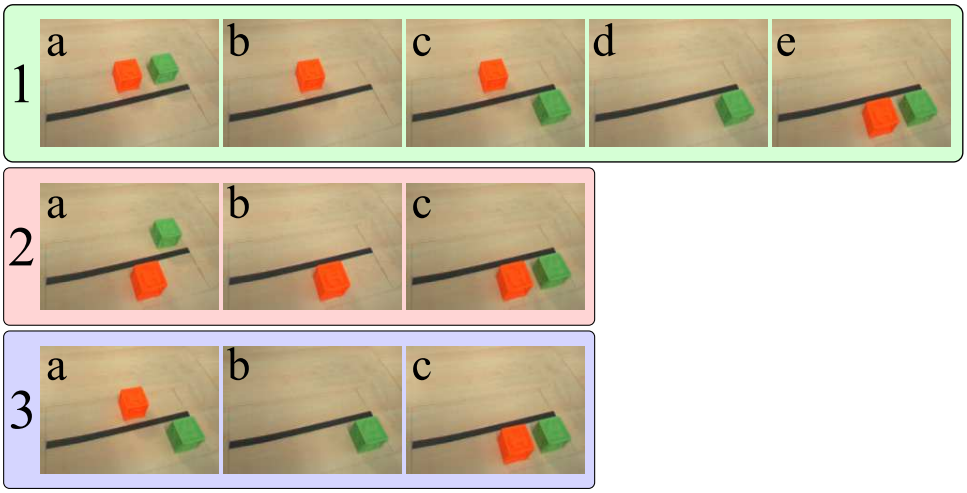


Figure 4-39: Three different sets of demonstrations devised by the tutor for implying the rule of the game in the second experiment. For more information, see Section 4.7.7.

Pull All Objects in a Given Order

In the previous experiment, the robot learned that all objects must be kept close, without giving importance to the order of actions. In this experiment, to give priority to the sequence of actions to be performed, the color attribute for each object is included as well. For instance, consider the ordered set $\langle orange, green, yellow \rangle$ in which the orange cube and the yellow cube are the most and least important cubes, respectively. This means that we want the orange cube to be moved before the green one, and the green cube before the yellow one. The robot has to learn that the most important cube has to be operated first. Besides, learning a sequence of actions in which the order is important, is one of the main capabilities of VSL. Therefore, if the task was demonstrated once and the reproduction part was done by VSL (and not by SP) the robot would learn the task using one single demonstration. In addition, VSL provides the robot with the capability of generalizing the task over the initial configuration of the objects.

Furthermore, it is possible to encode such an ordering also in a planning domain, at the cost of adding extra constraints (in the form of predicate) to the definition of the planning problem. In this case the planner should include two extra predicates, namely `(somewhere-after (far` B_{green}`) (far` B_{orange}`))` and `(somewhere-after (far` B_{yellow}`) (far` B_{green}`))`. The inclusion of these predicates must be managed at

an extra-logic level, e.g., by extracting the sequence of actions using VSL and inserting the corresponding constrains in the planner problem definition. In this case, there is no need to have more demonstrations, but expressing the order of sequence as preconditions may be done automatically.

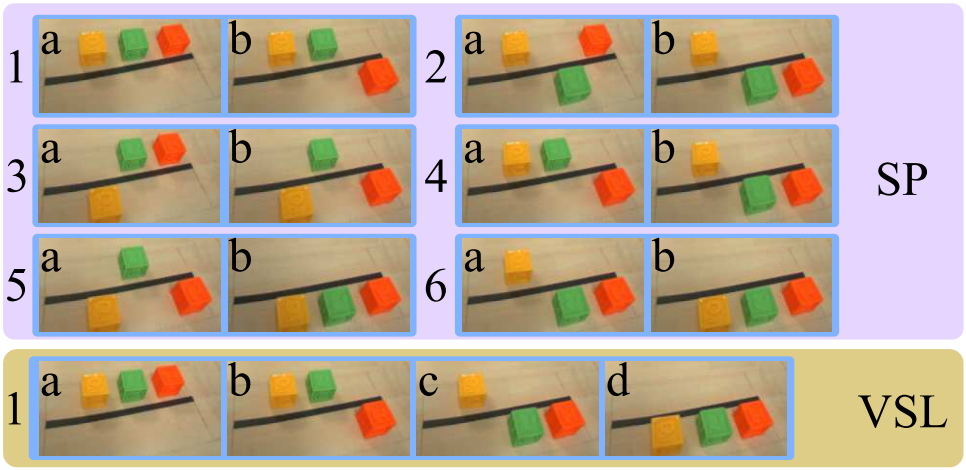


Figure 4-40: In the third experiment, using SP, the robot needs six sets of demonstrations for each of which the initial (a) and final (b) configurations of the objects are shown. While using VSL the robot requires a single demonstration for which the sequence of operations are shown (a-d). For more information, see Section 4.7.7.

4.8 Chapter Summary

In this chapter a novel skill learning approach has been proposed which allows a robot to acquire new skills for object manipulation by observing a demonstration. The proposed approach is called Visuospatial Skill Learning (VSL), because it allows a robot to learn new visuospatial skills. VSL which is a significant contribution of this thesis, utilizes visual perception as the main source of information. As opposed to conventional trajectory-based learning approaches, VSL is a goal-based robot learning approach that focuses on achieving a desired goal configuration of objects relative to one another while maintaining the sequence of operations. VSL is capable of learning and generalizing multi-operation skills from a single demonstration, while requiring minimum *a priori* knowledge about environment. In contrast to many existing approaches, VSL leverages simplicity, efficiency and user-friendly human-robot interaction. The feasibility and efficiency of VSL has been

validated through simulated and real-world experiments.

In VSL, the primitive actions are generated using a trajectory generation module. These trajectories are programmed manually into the system and need to be tuned according to the desired task. To overcome this drawback, VSL has been integrated with a conventional trajectory-based approach, imitation learning. The robot learns the required primitive actions through kinesthetic teaching using imitation learning. And then it learns the sequence of actions to reach the desired goal of the task through VSL. The obtained framework easily extends and adopts robot's capabilities to novel situations, even by users without programming ability.

VSL has its own simple internal planner in the reproduction phase. To utilize the advantages of standard symbolic planners, the original planner can be replaced with an already available action-level symbolic planner. In order to ground the actions, the symbolic actions are defined in the planner and VSL maps identified preconditions and effects in a formalism suitable to be used at the symbolic level. The experimental results have shown the improvement in generalization to find a solution that can be acquired from both multiple and single demonstrations. The integration of VSL and SP is a substantial contribution of this chapter, which has a potential for the future. The coupling between VSL and SP is natural and intuitive and brings significant advantages over using VSL separately. Learning preconditions and effects of the actions in VSL approach and making symbolic plans based on the learned knowledge deals with the long-standing important problem of Artificial Intelligence, namely Symbol Grounding. The feasibility and capability of the proposed framework have been experimentally validated. In the experiments it has been shown that the symbols and rules are grounded through learning in the problem domain.

Finally by integrating Imitation Learning (IL), Visuospatial Skill Learning (VSL), and conventional planning methods a novel learning framework has been developed. In this framework, the sensorimotor skills are learned through IL. The sequence of performed actions is learned through demonstrations using VSL. A standard action-level planner is used to represent a symbolic description of the skill, which allows the system to represent the skill in a discrete, symbolic form. VSL identifies the underlying constraints of the task

and extracts symbolic predicates, thereby updating the planner representation while the skills are being learned. Therefore the planner maintains a generalized representation of each skill as a reusable action, which can be planned and performed independently during the learning phase. The feasibility and efficiency of the integrated framework has been validated through real-world experiments with the iCub humanoid robot.

"What we know is a drop, what we don't know is an ocean."

—Isaac Newton

5 Conclusions

THIS chapter summarizes the thesis as a whole by reviewing the contents described in the core chapters. It then points out the research contributions extracted from each chapter. In addition, a few remarks on future direction of the research are made.

5.1 Thesis Summary

Autonomous robots are supposed to operate independently in unstructured environments without any form of external control for extended periods of time. Following such definition, most existing robotic platforms can be categorized as either non-autonomous or semi-autonomous systems. Nowadays, however, developing autonomous systems is strongly under investigation. Researchers are striving to evolve robots with full cognitive capacities that can solve complex tasks independently. These systems are capable of learning in

their current situation to deal with unexpected environmental conditions, updating their learned models, and making decisions self-reliably.

In order to elevate autonomy in robots, a collection of robot learning approaches has been proposed in this thesis. These approaches allow a robot to acquire novel skills or adopt to its environment through learning algorithms. They also address several characteristics of autonomous systems by placing the focus on learning reactive and visuospatial skills. These skills are extremely important for an autonomous system and affect many aspects of autonomy in robots remarkably. In addition, these skills seem not to be thoroughly studied in the literature and need further attention.

Reactive behavior is one of the most crucial behaviors that an autonomous system should possess. Biological systems, such as animals and humans are able to learn and develop reactive behaviors efficiently. Robots, on the other hand, are not good at being reactive and this is one of the factors that makes them incompatible with unpredictable and changing environments. By definition, reactive means tending to be responsive or to react to a stimulus, which can be either internal or external. Internal stimuli occur in the internal states of the system for instance noise on an internal sensor, or failure in an actuator. External stimuli, on the other hand, occur in the environment in which the robot is operating. An external disturbance such as wind or water current, obstacle collision, or variation of illumination in a vision system are some examples of external stimuli.

A novel learning approach has been proposed in Chapter 2 that allows a robot to react to external stimuli while operating in an unstructured environment. The proposed hierarchical learning approach that has been devised for performing the challenging task of autonomous valve turning, includes three main layers. Each layer realizes specific subtasks to improve the autonomy of the system. In the first layer, the robot acquires novel motor skills of approaching and grasping the valve by kinesthetic teaching. A hybrid force/motion control strategy has been utilized in this layer to dissipate undesired forces and torques during the turning phase. A Reactive Fuzzy Decision Maker (RFDM) has been devised in the second layer which reacts to the relative movement between the valve and the gripper, sensor delay and forces/torques applied to the valve, according to the distance between the valve and the gripper. The reactive layer evaluates the dynamic

behavior of the system and modulates the robot's movement accordingly by changing the direction and rate of the movement. In the third layer, a supervised learning method has been utilized to tune the behavior of the system based on expert knowledge. Conscious knowledge of a human expert is utilized to devise the main rules of the system. Whereas, the subconscious knowledge of the human expert is used to tune the rules. The proposed hierarchical learning approach has been validated through real-world experiments first in the laboratory environment and finally in underwater. The results have shown the feasibility and efficiency of the proposed approach.

Autonomous robots are required to react efficiently not only to external uncertainties but also to failures and uncertainties which occur in the internal states of the system. To tackle this problem, a learning framework has been proposed in Chapter 3 that enables a robot to learn a reactive behavior for dealing with internal failures. This chapter has placed the focus on improving fault-tolerance in underwater robotic domain in order to increase the reliability and autonomy of AUVs. One of the most crucial failures which can endanger both the underwater robot and the mission is thruster failure. The proposed learning framework discovers new control policies to overcome thruster failures as they happen by employing a model-based direct policy search. The policies are learned on an on-board simulated model of the AUV. When a fault is detected and isolated, the model is reconfigured according to the new condition. Since the learning framework 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. One of the advantages of the proposed framework is that it is applicable in both cases when the AUV becomes under-actuated in the presence of a fault or remains over-actuated. For dealing with multiple conflicting objectives, the proposed learning framework has been extended by employing multi-objective reinforcement learning. The extended framework discovers a set of optimal solutions, each of which can be used to generate a trajectory that is able to navigate the AUV towards a specified target while satisfying multiple objectives. Unlike most existing methods which disregard the faulty thruster, the proposed framework can also deal with partially broken thrusters. The feasibility and efficiency of the proposed

learning framework have been validated through simulated and real-world experiments.

A novel skill learning approach for object manipulation has been proposed in Chapter 4 which allows a robot to acquire new skills. The robot attains the new skills by observing a demonstration while interacting with a human tutor. The proposed approach focuses on achieving a desired goal configuration of objects relative to one another. Based on visual perception, the robot captures a visuospatial representation including the relative positioning of the object with respect to multiple other objects simultaneously. Since it learns by observing the visuospatial representation, the approach is called Visuospatial Skill Learning (VSL). VSL is capable of learning and generalizing different skills such as object reconfiguration, classification, and turn-taking interaction. The robot learns to achieve the goal from a single demonstration while requiring minimum *a priori* knowledge about the environment. The capabilities of VSL have been illustrated using simulated and real-world experiments. The proposed VSL approach has been extended by tackling its limitations through integration. In VSL, the primitive actions have to be programmed manually into the system. To handle this drawback, VSL has been integrated with a conventional trajectory-based approach, imitation learning. The sensorimotor skills are learned through imitation learning. A standard action-level planner is used to represent a symbolic description of the skill, which allows the system to represent the skill in a discrete, symbolic form. VSL identifies the underlying constraints of the task and extracts symbolic predicates, thereby updating the planner representation while the skills are being learned. Therefore the planner maintains a generalized representation of each skill as a reusable action, which can be planned and performed independently during the learning phase. Several real-world experiments have been conducted to show the validity of the proposed approach.

5.2 Contributions

The proposed goal of investigating and evolving robot learning approaches for acquiring reactive and visuospatial skills has been accomplished in this thesis. All the approaches have been validated through various simulated and real-world experiments. For the real-

world experiments, four robotic platforms have been employed: a 7-DoF Kuka robotic manipulator, a 7-DoF Barrett WAM equipped with a Barrett Hand, a Girona500 AUV equipped with a 4-DoF robotic arm, and an iCub humanoid robot. Several research contributions were achieved during the development of this goal that are listed below.

Contributions of Chapter 2

- A hierarchical learning approach has been proposed that enables a robot to acquire a reactive behavior for coping with uncertainties. The approach has been devised for performing the challenging task of autonomous robotic valve turning. The proposed hierarchical architecture consists of three main layers each of which realizes specific subtasks to improve the autonomy of the system.

- The robot acquires novel motor skills for approaching and grasping the valve through kinesthetic teaching based on imitation learning. In addition, a hybrid force/motion control strategy has been utilized to dissipate undesired forces and torques during the turning phase. Integration of the imitation learning technique with a hybrid control strategy is one of the contributions of this thesis.

- A reactive fuzzy decision maker system has been devised that modulates the movements of the robot during the execution of the task by observing the uncertainties and disturbances. The reactive system affects the behavior of the robot by modulating the rate and the direction of the movement. The proposed reactive system enables the robot to deal with uncertainties in the environment by learning from human knowledge. Conscious knowledge of a human expert is utilized to devise the fuzzy rule set. Whereas, the subconscious knowledge of the human expert is used to tune the rules.

- Autonomous valve turning is a challenging task that many robotic groups are trying to accomplish. One of the contributions of this thesis is achieving autonomous valve turning in underwater environment by employing the proposed hierarchical learning approach. The reactive behavior of the robot during the execution of the

task evidently plays a crucial role that results in elevating the autonomy of the system.

Contributions of Chapter 3

- To address the problem of dealing with changes in the internal states of the system, a novel learning framework has been proposed that enables a robot to learn a reactive behavior. The proposed framework focuses on elevating fault-tolerance and reliability in AUVs. Using a model-based direct policy search, new control policies are discovered to overcome thruster failures as they happen. The policies are learned on a simulated model of the AUV on-board. The model is adapted according to the new condition when a fault is detected and isolated. The learned policy is then executed on the real robot in a closed-loop using the state feedback of the AUV. One of the advantages of this approach over existing approaches is that it is applicable in both cases which the robot becomes under-actuated or remains over-actuated in the presence of the failure. Most existing approaches have been developed for over-actuated systems and there are few cases dealing with under-actuated scenarios which are not applicable when the system remains over-actuated. In addition, the proposed 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. Unlike most existing methods which disregard the faulty thruster, the proposed framework can also deal with partially broken thrusters to increase the autonomy of the AUV.

- In underwater domain, there are few cases involving online learning of a behavior. Demonstrating the feasibility of the proposed learning framework in real-world experiments therefore, is an important contribution. In other words, using a state-dependent policy and applying a closed-loop fault-tolerant control with state feedbacks, an optimal policy can be computed on-board and executed online. So, contrary to many existing methods, the proposed framework generates an optimal policy that can be directly utilized by the AUV in real-world scenarios. This

capability also reduces the involvement of human supervisor in the level of trajectory generation, and consequently improves the autonomy of the system.

- To deal with multiple objectives which can be complementary, conflicting, or independent, two variants of the proposed frameworks have been developed. The first variant utilizes linear scalarization technique which is suitable for independent and complementary objectives. The second variant, on the other hand, utilizes a multi-objective algorithm that can deal with multiple conflicting objectives by discovering multiple optimal solutions.

- The presented learning framework can be generalized and implemented in various underwater vehicles and it is not developed exclusively for the Girona500 AUV. The reason is that the theoretical aspect of the approach 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.

Contributions of Chapter 4

- One of the contributions of this thesis is the Visuospatial Skill Learning approach which allows a robot to acquire new visuospatial skills for object manipulation by observing a demonstration. VSL utilizes visual perception as the main source of information. As opposed to conventional trajectory-based learning approaches, VSL is a goal-based robot learning approach that focuses on achieving a desired goal configuration of objects relative to one another while maintaining the sequence of operations. VSL is capable of learning and generalizing multi-operation skills from a single demonstration, while requiring minimum *a priori* knowledge about the objects and the environment. In contrast to many existing approaches, VSL leverages simplicity, efficiency and user-friendly human-robot interaction.

- In VSL, the primitive actions have to be programmed manually into the system. To overcome this drawback, VSL has been integrated with a conventional trajectory-based approach, imitation learning. Thus, the robot firstly learns the required

primitive actions through kinesthetic teaching. And then it learns the sequence of actions to reach the desired goal of the task through VSL. The obtained approach easily extends and adopts robot's capabilities to novel situations, even by users without programming ability.

- VSL has its own simple internal planner in the reproduction phase. To utilize the advantages of standard symbolic planners, the original planner has been replaced with an existing action-level symbolic planner. In order to ground the actions, the symbolic actions are defined in the planner and VSL maps identified preconditions and effects in a formalism suitable to be used at the symbolic level. The integration of VSL and SP is a substantial contribution of this thesis. The coupling between VSL and SP is natural and intuitive and brings significant advantages over using VSL separately. Learning preconditions and effects of the actions in VSL approach and making symbolic plans based on the learned knowledge deals with the long-standing important problem of Artificial Intelligence, namely Symbol Grounding. The feasibility and capability of the proposed framework have been experimentally validated. In the experiments, it has been shown that the symbols and rules are grounded through learning in the problem domain.

- By integrating Imitation Learning (IL), Visuospatial Skill Learning (VSL), and conventional planning methods a novel learning framework has been developed. In this framework, the sensorimotor skills are learned through IL. The sequence of performed actions is learned through demonstrations using VSL. A standard action-level planner has been used to represent a symbolic description of the skill, which allows the system to represent the skill in a discrete, symbolic form. VSL identifies the underlying constraints of the task and extracts symbolic predicates, thereby updating the planner representation while the skills are being learned. Therefore the planner maintains a generalized representation of each skill as a reusable action, which can be planned and performed independently during the learning phase. The feasibility and efficiency of the integrated framework have been validated through real-world experiments

5.3 Future Work

In the proposed VSL approach only visual information is used. Also the developed learning framework including, imitation learning, VSL and a symbolic planner, the trajectories in the form of position were also considered. A possible extension for this framework is in the information-space that defines the context. For instance, force data extracted from demonstrations can be used as the main or additional source of context. This may be done using wrist-place worn accelerometers or a motion-capture system.

In the proposed VSL an external vision sensor has been used that captures observations from workspace. To develop a novel interface for robot programming, one interesting extension is to include virtual reality devices (e.g. Oculus Rift). In such case, the human performs the demonstration using his hands, while perceiving the scene in 3D using the VR device. This idea can lead to better results with less number of failures.

Appendices

A

Differential Evolution Algorithm

THIS appendix provides a short description of the Differential Evolution algorithm which has been employed in Chapter 3. Differential Evolution (DE) is a population-based stochastic method for global optimization and its recombination and mutation operators are the variation operators used to generate new solutions. Unlike, Genetic Algorithm (GA) and several Evolution Strategy (ES) approaches, solutions in DE are encoded with real values. Moreover, DE does not use a fixed distribution as the Gaussian distribution adopted in ES approaches; instead, the current distribution of the solutions in the search space determines the search direction and even the step-size for each individual. DE searches for a global optimum point in a D-dimensional real parameter space $\mathbb{R}^D$. The algorithm starts with a randomly initialized population of N_p D-dimensional real-values parameter vector. The i^{th} vector of the population at the current generation G is denoted as:

$$\mathbf{x}_{i,G} = [x_{1,i,G}, x_{2,i,G}, \ldots, x_{D,i,G}]^{\mathrm{T}} \tag{A.1}$$

The first step is to select the control parameters of the algorithm including the scale factor F, the crossover rate C_r, and the population size N_p. G is the generation number which is set to 0 at the beginning. The first population is uniformly distributed to a specified range defined by user constrained by minimum and maximum bounds:

$$\mathbf{x}_{\min} = [x_{1,\min}, x_{2,\min}, \ldots, x_{D,\min}]^{\mathrm{T}} \tag{A.2}$$

$$\mathbf{x}_{\max} = [x_{1,\max}, x_{2,\max}, \ldots, x_{D,\max}]^{\mathrm{T}} \tag{A.3}$$

with $i = [1, 2, \ldots, N_p]$. The next step is mutation. The mutation strategy uses three random samples, say ρ_1, ρ_2 and ρ_3, and the scale factor F. The difference of any two of these three vectors is scaled by F and the scaled difference is added to the third one as follows:

$$\mathbf{v}_{i,G} = \mathbf{x}_{\rho_1,G} + F(\mathbf{x}_{\rho_2,G} - \mathbf{x}_{\rho_3,G}). \tag{A.4}$$

In the next phase, the crossover operation is introduced to increase the diversity of the perturbed parameter vector. The trial vector

$$\mathbf{u}_{i,G+1} = (u_{1,i,G+1}, u_{2,i,G+1}, ..., u_{D,i,G+1}), \tag{A.5}$$

is formed according to

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & \text{if } \texttt{rand}(j) \leq C_r \text{ or } j = \texttt{rnbr}(i) \\ x_{j,i,G} & \text{otherwise} \end{cases} \tag{A.6}$$

where, $\texttt{rand}(j)$ is the j^{th} evaluation of a uniform random number. $C_r \in [0,1]$ is the crossover constant. $\texttt{rnbr}(i)$ is a randomly chosen index which ensures that $\mathbf{u}_{i,G+1}$ gets at least one parameter from $\mathbf{v}_{i,G+1}$. The selection operator is used to decide whether or not the trial vector should become a member of the next generation. By comparing the trial vector to the target vector, the selection operator selects the vector with smaller cost. A

pseudocode listing of the main algorithm is provided in Algorithm 3.

In order to classify the different variants of DE the notation `DE/a/b/c` is proposed by Storn and Price [96]. `a` specifies the vector to be mutated. Two examples are `rand` and `best`. `b` is the number of difference vectors used in the strategy. `c` denotes the crossover scheme (e.g. `bin`). The explained DE strategy in this section can be classified as: `DE/rand/1/bin`. This appendix does not describe the DE algorithm comprehensively. More information about DE can be found in [96]. Furthermore, the effect of the constant parameters of the algorithm and related tuning procedures have been studied in recent years [100].

Input : $\{C_r, F, N_p, D, G_{\max}\}$

Initialization:

$G \leftarrow 0$ // generation number

$i \leftarrow 1$ // population number

$\mathbf{x}_{i,G} = [x_{1,i,G}, x_{2,i,G}, \ldots, x_{D,i,G}]^{\mathrm{T}}$ with $i = [1, 2, \ldots, N_p]$ // each individual

$\mathbf{P}_G = \{\mathbf{x}_{1,G}, \ldots, \mathbf{x}_{N_p,G}\}$ // initialize a population in the constrained bounds

while *Termination criterion* **do**

  for $i = 1, \ldots, N_p$ **do**

    // do for each individual sequentially

    // Mutation

    $\mathbf{v}_{i,G} = \mathbf{x}_{r_1^i,G} + F(\mathbf{x}_{r_2^i,G} - \mathbf{x}_{r_3^i,G})$

    // Crossover

    $$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{if} \quad \mathtt{rand}(j) \le C_R \quad \text{or} \quad j = \mathtt{rnbr}(i) \\ x_{ji,G} & \text{otherwise} \end{cases}$$ // Selection

    if $f(\mathbf{u}_{i,G}) \le f(\mathbf{u}_{i,G})$ **then**

      $\mathbf{x}_{i,G+1} = \mathbf{u}_{i,G}$

    else

      $\mathbf{x}_{i,G+1} = \mathbf{x}_{i,G}$

    end

  end

  $G \leftarrow G + 1$

end

Algorithm 3: The Differential Evolution algorithm.

B

CMA-ES Algorithm

THIS appendix provides a short description of the CMA-ES algorithm which has been employed in Chapter 2. CMA stands for covariance matrix adaptation. The CMA-ES algorithm is a stochastic optimization evolution strategy for real-parameter, non-linear, non-convex functions in continuous domain [46, 177]. CMA-ES is designed for black-box minimization of an objective function:

$$f : \mathbb{R}^n \rightarrow \mathbb{R} \tag{B.1}$$

$$\mathbf{x} \mapsto f(\mathbf{x}) \tag{B.2}$$

The objective is to find one or more search points (candidate solutions), $\mathbf{x} \in \mathbb{R}^n$, with a function value $f(\mathbf{x})$, as small as possible. In CMA-ES a population of new offsprings

is generated by sampling a multivariate normal distribution. The basic equation for sampling the generation $g = 0, 1, 2, \ldots$, is as follows:

$$\mathbf{x}_k^{(g+1)} \sim \mathbf{m}^{(g)} + \sigma^{(g)} \mathcal{N}(\mathbf{0}, \mathbf{C}^{(g)}) \;\; k = 1, \ldots, \lambda, \tag{B.3}$$

where $\sim$ indicates the identical distribution on the left and right side, $\mathcal{N}(\mathbf{0}, \mathbf{C}^{(g)})$ is a multivariate normal distribution with zero mean and covariance matrix $\mathbf{C}^{(g)}$, $\mathbf{x}_k^{(g+1)} \in \mathbb{R}^n$ is the k offspring from generation $g+1$, $\mathbf{m}^{(g)} \in \mathbb{R}^n$ is mean value of the search distribution at generation g, $\sigma^{(g)} \in \mathbb{R}_+$ is overall standard variation or step-size at generation g, $\mathbf{C}^{(g)} \in \mathbb{R}^{n \times n}$ is covariance matrix at generation g, and $\lambda \geq 2$ is population size or number of offsprings.

The mean of the new generation $\mathbf{m}^{(g+1)}$ for the search distribution is a weighted average of μ selected points from the sample $x_1^{(g+1)}, \ldots, x_\lambda^{(g+1)}$ as follows:

$$\mathbf{m}^{(g+1)} = \sum_{i=0}^{\mu} w_i \, x_{i:\lambda}^{(g+1)} \tag{B.4}$$

$$\sum_{i=0}^{\mu} w_i = 1, \quad w_1 \geq w_2 \geq \ldots \geq w_\mu > 0, \tag{B.5}$$

where $\mu \leq \lambda$ is the parent population size, $w_{i=1\ldots\mu} \in \mathbb{R}_+$ is positive weight coefficients for recombination. $x_{i:\lambda}^{(g+1)}$ indicates i best individual out of $x_1^{(g+1)}, \ldots, x_\lambda^{(g+1)}$. The index $i : \lambda$, denotes the index of the i ranked individual and $f(x_{1:\lambda}^{(g+1)}) \leq f(x_{2:\lambda}^{(g+1)}) \leq, \ldots, \leq f(x_{\lambda:\lambda}^{(g+1)})$.

Equation (B.4) implements truncation selection by choosing $\mu < \lambda$ out of λ offspring points. Assigning different weights w_i must also be interpreted as a selection mechanism. Equation (B.5) implements weighted intermediate recombination by taking $\mu > 1$ individuals into account for a weighted average. Variance effective selection mass, μ_{eff} is defined to select the effective solution points. Choosing $\mu_{\text{eff}} \approx \lambda/4$, with $\mu = \lambda/2$ is a reasonable choice. In CMA-ES the evolution path $p_c \in \mathbb{R}^n$ with exponential smoothing and starting from $p_c = 0$ is defined as:

$$p_c^{(g+1)} = (1-c_c)p_c^{(g)} + \sqrt{c_c(2-c_c)\mu_{\text{eff}}}\frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}}, \tag{B.6}$$

where $p_c^{(g)} \in \mathbb{R}^n$ is evolution path at generation g, $c_c \leq 1$ and $1/c_c$ is the backward time horizon of the evolution path, and the factor $\sqrt{c_c(2-c_c)\mu_{\text{eff}}}$ is a normalization constant. CMA-ES updates the covariance matrix according to the following equation:

$$\mathbf{C}^{(g+1)} = (1-c_1-c_\mu)\mathbf{C}^{(g)} + c_1 p_c^{(g+1)} p_c^{(g+1)^T} + c_\mu \sum_{i=0}^{\mu} w_i y_{i:\lambda}^{(g+1)} (y_{i:\lambda}^{(g+1)^T}), \tag{B.7}$$

where $c_1 \approx 2/n^2$, $c_\mu \approx \min(\mu_{\text{eff}}/n^2, 1-c_1)$, and $y_{i:\lambda}^{(g+1)} = (x_{i:\lambda}^{(g+1)} - m^{(g)})/\sigma^{(g)}$. In addition, the conjugate evolution path p_σ is defined as follows:

$$p_\sigma^{(g+1)} = (1-c_\sigma)p_\sigma^{(g)} + \sqrt{c_\sigma(2-c_\sigma)\mu_{\text{eff}}}\mathbf{C}^{(g)^{-1/2}}\frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}}, \tag{B.8}$$

where $p_\sigma^{(g)} \in \mathbb{R}^n$ is conjugate evolution path at generation g, $c_\sigma \leq 1$ and $1/c_\sigma$ is the backward time horizon of the evolution path, and the factor $\sqrt{c_\sigma(2-c_\sigma)\mu_{\text{eff}}}$ is a normalization constant. $\mathbf{C}^{(g)^{-1/2}}$ is an eigen-decomposition of $\mathbf{C}^{(g)}$. And finally, the step-size is controlled using the following equation:

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\| p_\sigma^{(g+1)} \|}{\mathbf{E}\| \mathcal{N}(\mathbf{0}, \mathbf{I}) \|} - 1\right)\right), \tag{B.9}$$

where $d_\sigma \approx 1$ is a damping parameter that scales the change magnitude of $\sigma^{(g)}$.

Input : $\{\lambda, \mu, w_{i=1\ldots\mu}, c_\sigma, d_\sigma, c_c, c_1, c_\mu\}$

Initialization:

$p_c = 0$, $p_\sigma = 0$, $\mathbf{C} = \mathbf{I}$, $g = 0$

choose $m \in \mathbb{R}^n$ and $\sigma \in \mathbb{R}_+$

repeat

  $g \leftarrow g + 1$ `// sample new population`

  $z_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

  $y_k = \mathbf{B}\mathbf{D}z_k \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$

  $x_k = m + \sigma y_k \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$

  `// selection and recombination`

  $\langle y \rangle_w = \sum_{i=1}^{\mu} w_i y_{i:\lambda}$, where $\sum_{i=1}^{\mu} w_i = 1, w_i > 0$

  $m \leftarrow m + \sigma \langle y \rangle_w = \sum_{i=1}^{\mu} w_i x_{i:\lambda}$

  `// step-size control`

  $p_\sigma \leftarrow (1 - c_\sigma) p_\sigma + \sqrt{c_\sigma (2 - c_\sigma) \mu_{\text{eff}}} \mathbf{C}^{-1/2} \langle y \rangle_w$

  $\sigma^{(g+1)} \leftarrow \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|p_\sigma^{(g+1)}\|}{\mathbf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|} - 1 \right) \right)$

  `// covariance matrix adaptation`

  $p_c \leftarrow (1 - c_c) p_c + h_\sigma \sqrt{c_c (2 - c_c) \mu_{\text{eff}}} \langle y \rangle_w$

  $\mathbf{C} \leftarrow (1 - c_1 - c_\mu)\mathbf{C} + c_1 (p_c p_c^T + \delta(h_\sigma)\mathbf{C}) + c_\mu \sum_{i=0}^{\mu} w_i y_{i:\lambda} (y_{i:\lambda}^T)$

until *Termination criterion*

Algorithm 4: The (μ/μ_W)-CMA evolution strategy.

Default parameters are selected according to the following equations:

$$\lambda = 4 + 3\ln\lfloor n \rfloor \tag{B.10}$$

$$\mu = \lfloor \mu' \rfloor \tag{B.11}$$

$$\mu' = \lambda/2 \tag{B.12}$$

$$w_i = \frac{w_i'}{\sum_{i=1}^{\mu} w_i'} \tag{B.13}$$

$$w_i' = \ln(\mu' + 0.5) - \ln(i) \;\; i = 1, \ldots, \mu \tag{B.14}$$

$$c_\sigma = \frac{\mu_{\text{eff}} + 2}{n + \mu_{\text{eff}} + 5} \tag{B.15}$$

$$d_\sigma = 1 + 2\max\left(0, \sqrt{\frac{\mu_{\text{eff}} - 1}{n+1}} - 1\right) + c_\sigma \tag{B.16}$$

$$c_c = \frac{4 + \mu_{\text{eff}}/n}{n + 4 + 2\mu_{\text{eff}}/n} \tag{B.17}$$

$$c_1 = \frac{2}{(n+1.3)^2 + \mu_{\text{eff}}} \tag{B.18}$$

$$c_\mu = \min\left(1 - c_1, \alpha_\mu \frac{\mu_{\text{eff}} - 2 + 1/\mu_{\text{eff}}}{(n+2)^2 + \alpha_\mu \mu_{\text{eff}}/2}\right) \tag{B.19}$$

More information on CMA-ES can be found in [46, 177].

C

Cross Entropy Method

THIS appendix provides a short description of the Cross Entropy Method which has been employed in Chapter 2. The Cross Entropy Method (CEM) is proposed by Rubinstein *et al.,* [44]. CEM is a Monte Carlo approach used for continuous and combinatorial optimization and importance sampling. CEM can be applied to static and noisy global optimization problems. In CEM firstly a random data sample is generated according to a specified mechanism. Then, the parameters of the random mechanism are updated to produce a better sample in the next iteration. For estimating the quality $\ell = \mathbb{E}_u[H(\mathbf{X})] = \int H(\mathbf{x}) f(\mathbf{x};\mathbf{u}) d\mathbf{x}$, where H is some performance function and $f(\mathbf{x};\mathbf{u})$ is a member of some parametric family of distributions. This quantity can be estimated using importance sampling as: $\ell = \frac{1}{N}\sum_{i=1}^{N} H(\mathbf{X}_i)\frac{f(\mathbf{X}_i;u)}{g(\mathbf{X}_i)}$, where $\mathbf{X}_1,\ldots,\mathbf{X}_N$ is a random sample from g. The optimal importance sampling density (PDF) is given by, $g^* = H(\mathbf{x}) f(\mathbf{x};\mathbf{u})/\ell$. The goal of CEM is to approximate the optimal PDF by adaptively selecting members of the

parametric family that are closest to the optimal PDF g^*.

1 **Initialization:**

2 $t = 1$ // iteration counter

3 choose $\mathbf{v}^0 \in \mathbb{R}^n$

4 **repeat**

    // sample new population

5 $\mathbf{X}_1, \ldots, \mathbf{X}_N \sim f(.; \mathbf{v}^{t-1})$

6 $\mathbf{v}^t = \arg\max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^{N} \frac{f(\mathbf{X}_i; \mathbf{u})}{f(\mathbf{X}_i; \mathbf{v}^{t-1})} \log f(\mathbf{X}_i; \mathbf{v})$

7 $t \leftarrow t + 1$

8 **until** *Termination criterion*

Algorithm 5: The Cross Entropy Method.

In Algorithm 5, $\mathbf{v}^t$ is the parameter vector in the t^{th} iteration. $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is a sample vector drawn from $f(.; \mathbf{v}^{t-1})$ PDF. For more information about the Cross Entropy Method, refer to [44, 45].

D

Simulated Annealing Algorithm

THIS appendix provides a short description of the Simulated Annealing algorithm which has been employed in Chapter 3. Simulated Annealing (SA) is a probabilistic meta-heuristic which has been proposed in the area of combinatorial optimization [95] that is, when the objective function is defined in a discrete domain. SA is proposed for finding the global minimum of a cost function that may possess several local minima [178]. The method is reported to perform well in the presence of a very high number of variables (even tens of thousands). The SA heuristic was modified in order to apply to the optimization of multi-modal functions defined in continuous domain [179]. SA comes from the Metropolis algorithm, a simulation of the recrystallization of atoms in metal during its annealing (gradual and controlled cooling). During annealing, atoms migrate naturally to configurations that minimize the total energy of the system, even if during this migration the system undergoes high-energy configurations. The observation of this behavior suggested

the application of the simulation of such a process to combinatorial optimization problems [180]. The choice of the temperature or cooling scheduling and the next candidate distribution are the most important decisions in the definition of an SA algorithm [97]. The temperature schedule defines how the temperature in SA is decreased. There exists a wide range of methods to determine this temperature schedule [181]. Corana *et al.,* [179] proposed a self-tuning SA algorithm the step size of which is configured in order to maintain a number of accepted solutions. The probability distribution used by Corana et al. is flat. Ingber [182], on the other hand, proposed to use a Cauchy distribution and a faster temperature decrease. The very fast simulated re-annealing proposed by Ingber and Rosen [183] searches for different sensitivities in parameters by processing the first derivatives of the objective function. In our work, we use Boltzman temperature scheduling. The SA algorithm proposed in [180] uses a Gaussian probability distribution with a self-controlled standard deviation in order to maintain the number of accepted solutions. The starting temperature must be hot enough to allow a move to almost any neighborhood state. If this is not done, the ending solution will be very close to the starting solution. However, if the temperature starts at a too high value then the search can move to any neighbor and thus transform the search into a random search. Effectively, the search will be random until the temperature is cool enough to start acting as an SA algorithm. Several areas of application of SA can be enumerated, some of which are: chemistry and chemical engineering [184], image processing [185], economics and finance [186], electrical engineering and circuit design [187], geometry and physics [188], machine learning [189], networking and communication [190], etc. The pseudocode listing of the main Simulated Annealing algorithm is provided in Algorithm 6.

Input : $\{n, \mathrm{Iter}_{\max}, T_{\max}\}$

Output: $\{S_{\text{best}}\}$

Initialization:

$S_{\text{curr}} \leftarrow$ create n initial random solution $S_{\text{best}} \leftarrow S_{\text{curr}}$

for $i = 1, \ldots, Iter_{\max}$ **do**

  $S(i) \leftarrow$ create neighbor solution from S_{curr}

  $T_{curr} \leftarrow$ calculate current temperature $T_{\max}$

  if $\texttt{cost}(S(i)) \leq \texttt{cost}(S_{\text{curr}})$ **then**

    $S_{\text{curr}} = S(i)$

    if $\texttt{cost}(S(i)) \leq \texttt{cost}(S_{\text{best}})$ **then**

      $S_{\text{best}} = S(i)$

    end

  end

  else if $\left(\exp(\frac{\texttt{cost}(S_{\text{curr}}) - \texttt{cost}(S(i))}{T(S(i))} > \texttt{rand}()\right)$ **then**

    $S_{\text{curr}} = S(i)$

  end

end

Algorithm 6: The Simulated Annealing algorithm.

In Algorithm 6, n is the size of the problem, $\mathrm{Iter}_{\max}$ is the maximum number of iterations, $T_{\max}$ is the maximum temperature, S_{curr} and S_{best} are the current and best solutions respectively. This appendix does not provide a comprehensive description of SA. More information about SA can be found in [95, 178, 179].

E

Modified Price Algorithm

THIS appendix provides a short description of the Modified Price's algorithm which has been employed in Chapters 2 and 3. The Price's algorithm is a combination of a global and a local derivative-free method, designed for the optimization of non-linear, multi-modal, multivariate functions. This algorithm is global, derivative-free, and iterative. The algorithm is divided into two main phases: a controlled global random-search phase, and a deterministic local line-search phase. The algorithm utilized in the global phase has been introduced by Brachetti *et al.,* [47], and is reported in Algorithm 7. The global phase is population-based, and the initial population is drawn at random over D (line 2). It is also possible to add to the initial population any good point known in advance by the designer. The population at any time is composed by the best m points ever sampled, where m is a parameter of the algorithm. The bigger m, the more likely the algorithm is to avoid non-global minima. The algorithm terminates when

the difference between the best and the worst point of the population is less than the parameter ϵ. The population evolves by sampling a random family of $n+1$ points from the population itself, where n is the dimensionality of the domain, and computing the weighted centroid (lines 6–7). The next trial point is computed as a weighted reflection of the worst point of the population with respect to the weighted centroid (line 12). This algorithm has been proved to converge to the global minimum if uniform random sampling is performed together with the weighted centroid reflection [47]. Since this step guarantees the convergence in the limit by assigning a non-zero probability to the neighborhood of any point on the domain, it often compromises the performance on most functions in practice. Therefore, Leonetti *et al.,* [94] chose not to perform the uniform sampling, and to rely only on the heuristic provided by the centroid reflection. This approach has been extensively numerically evaluated in the literature [47, 94, 191].

Leonetti et al. [94] followed the approach presented by [47] and combined this global search with a deterministic local search. Therefore, instead of employing Algorithm 7 with a very small ϵ (typically in the order of 10^{-6}) they let $\epsilon = 10^{-2}$, and perform a deterministic local search in the neighborhood represented by the population at the time the global search is terminated. Although, global stochastic search is a powerful method to avoid being trapped in local minima, the random nature of its sampling makes it less effective when the region of the global minimizer has been identified. Thus, the best-point from the first global phase is used as the starting point of the following local search, which employs a coordinate-search algorithm with line-search expansions [192] reported in Algorithm 8.

The algorithm uses positive step sizes $\alpha_j, j = 1,\dots,2n$ along the cardinal directions $\{e_1,\dots,e_n,-e_1,\dots,-e_n\}$ to search for a point that improves the current best-point starting from θ^0. The parameters are initially set to:

$$\alpha_j = \frac{\text{median}\{\|x^i - x^S_{\min}\|,\, x^i \in S\}}{2}, j = 1,\dots,2n \tag{E.1}$$

where S is the final population from the previous algorithm. Thus, the parameters α_j, that are the initial steps in each direction, are set to half the median of the distance between

the points in the population and the current best-point ($x^S_{\min}$). This seamless integration between the global and local phases provides an improvement over the original formulation of this method [94], so that the parameter of the latter can be computed from the population of the former. We considered other measures for the radius of the population, for instance, the mean or the maximum distance, but the median proved to be the most effective in practice. Finally, if the largest step $\max_{i=1,\dots,2n}\{\tilde{\alpha}^k_i\}$ is smaller than the parameter ϵ the algorithm terminates (line 2). Trial points are subsequently generated, along the direction e_i and with steps α_i. If a point along a direction e_i starting at y^k_i improves y^k_i of at least $\gamma\cdot(\alpha^k_i)^2$, and also improves the current best-point x^k_i, then α^k_i is increased by a factor δ, and a farther point along e_i is tried (lines 7–14). This is called the expansion phase. If the trial point is not sufficiently improving, α^k_i is reduced by a factor σ in a contraction phase (lines 16–17), and other directions are polled. Typically $\delta = 2$ and $\sigma = 1/2$. When all the directions have been contracted up to ϵ the algorithm terminates. This line search algorithm draws its inspiration from gradient-based methods, and performs optimization along given lines, in this case the coordinate axes. While none of the directions is as fast descending as the gradient, about half of them will be improving directions. The expansion phase, in which the step is increased by a factor δ, allows to proceed along a successful line at an increasing pace, proving very effective in practice. Coordinate-search algorithms poll the function on a finite number of points, and are guaranteed to provide a locally optimum value at the required precision ϵ. No point closer to the current best-point than ϵ is polled. For more information about the Modified Price's algorithm, refer to [47, 94].

Input : $\{m \geq n+1, \epsilon > 0\}$

$k \leftarrow 0$

compute the initial set: $S^k = \{\theta_1^k, \ldots, \theta_m^k\}$ where the points θ_i^k, $i = 1, \ldots, m$ are chosen at random over a box D

evaluate J at each point θ_i^k, $i = 1, \ldots, m$.

determine the points $\theta_{\max}^k$, $\theta_{\min}^k$ and the values $J_{\max}^k$, $J_{\max}^k$ such that:
$J_{\max}^k = J(\theta_{\max}^k) = \max_{\theta \in S^k} J(\theta)$ and $J_{\min}^k = J(\theta_{\min}^k) = \min_{\theta \in S^k} J(\theta)$

if $J_{\max}^k - J_{\min}^k \leq \epsilon$ **then** STOP

choose at random $n+1$ points $\theta_{i_0}^k, \theta_{i_1}^k, \ldots, \theta_{i_n}^k$ over S^k, where
$J(\theta_{i_0}^k) \geq J(\theta_{i_j}^k), \quad j = 1, \ldots, n$

determine the centroid $c^k = \sum_{j=0}^n w_j^k \theta_{i_j}^k$

$w_j^k = \frac{\eta_j^k}{\sum_{r=0}^n \eta_r^k}$

$\eta_j^k = \frac{1}{J(\theta_{i_j}^k) - J_{\min}^k + \phi^k}$

$\alpha^k = 1 - \frac{J(\theta_{i_0}^k) - \sum_{j=0}^n w_j^k J(\theta_{i_j}^k)}{J_{\max}^k - J_{\min}^k + \phi^k}$

$\phi^k = n \frac{(J_{\max}^k - J_{\min}^k)^2}{J_{\max}^0 - J_{\min}^0}$

determine the trial point $\bar{\theta}^k$ given by $\bar{\theta}^k = c^k - \alpha^k(\theta_{i_0}^k - c^k)$

if $\bar{\theta}^k \notin D$ **then** go to 6

else compute $J(\bar{\theta}^k)$

if $J(\bar{\theta}^k) \geq J_{\max}^k$ **then**

  $S^{k+1} \leftarrow S^k$

  $k \leftarrow k+1$

  go to 6

else

  $S^{k+1} = S^k \cup \{\bar{\theta}^k\} - \{\theta_{\max}^k\}$

  $k \leftarrow k+1$

  go to 4

end

Algorithm 7: The controlled random search phase of Modified Price's algorithm.

Input : $\left\{\theta^0 \in \mathbb{R}^n, \tilde{\alpha}_1^0, \ldots, \tilde{\alpha}_{2n}^0 > 0, \sigma \in (0,1), \gamma \in (0,1), \delta > 1, \epsilon > 0\right\}$

$k \leftarrow 0$

if $\max_{i=1,\ldots,2n} \{\tilde{\alpha}_i^k\} \leq \epsilon$ **then** STOP

$i \leftarrow 1$

$y_1^k \leftarrow \theta^k$

$x^k \leftarrow \theta^k$

if $[J(y_i^k + \tilde{\alpha}_i^k e_i) \leq J(y_i^k) - \gamma(\tilde{\alpha}_i^k)^2$ *and* $J(y_i^k + \tilde{\alpha}_i^k e_i) < J(x^k)]$ **then**

  $\alpha_i^k = \tilde{\alpha}_i^k$

  $x^k = y_i^k + \alpha_i^k e_i$

  while $[J(y_i^k + \delta\alpha_i^k e_i) \leq J(y_i^k) - \gamma(\delta\alpha_i^k)^2$ *and* $J(y_i^k + \delta\alpha_i^k e_i) < J(x^k)]$ **do**

    $\alpha_i^k \leftarrow \delta\alpha_i^k$

    $x^k \leftarrow y_i^k + \alpha_i^k e_i$

  end

  $\tilde{\alpha}_i^{k+1} \leftarrow \alpha_i^k$

else

  $\alpha_i^k \leftarrow 0$

  $\tilde{\alpha}_i^{k+1} \leftarrow \sigma\tilde{\alpha}_i^k$

end

$y_{i+1}^k \leftarrow y_i^k + \alpha_i^k e_i$

if $i < 2n$ **then**

  $i \leftarrow i + 1$

  go to 7

end

$\theta^{k+1} \leftarrow x^k$

$k \leftarrow k + 1$

go to 2

Algorithm 8: The line search phase of Modified Price's algorithm.

List of Publications

The research presented in this thesis has led to a series of publications which are listed in this section according to their connections to different chapters of the thesis.

Related to Chapter 2 - Learning Reactive Behavior: A hierarchical Approach

- A. Carrera, **S. R. Ahmadzadeh**, A. Ajoudani, P. Kormushev, M. Carreras and D. G. Caldwell, *"Towards Autonomous Robotic Valve Turning,"* Journal of Cybernetics and Information Technologies (**CIT**), Vol. 12, no. 3, pp. 17–26, 2012.
- **S. R. Ahmadzadeh**, P. Kormushev, D. G. Caldwell, *"Autonomous Robotic Valve Turning: A Hierarchical Learning Approach,"* in Proceedings IEEE International Conference on Robotics and Automation, (**ICRA** 2013), Karlsruhe, Germany, 6-11 May, pp. 4614-4619, 2013.
- **S. R. Ahmadzadeh**, P. Kormushev, R. S. Jamisola Jr., D. G. Caldwell, *"Learning Reactive Robot Behavior for Autonomous Valve Turning,"* IEEE-RAS International Conference on Humanoid Robots (**Humanoids** 2014), Madrid, Spain, 18-20 Nov., 2014.

Related to Chapter 3 - Learning Reactive Behavior: A Fault-Tolerant Approach

- M. Leonetti, **S. R. Ahmadzadeh**, P. Kormushev, *"On-line Learning to Recover from Thruster Failures on Autonomous Underwater Vehicles,"* In Proceedings MTS/IEEE International Conference **OCEANS** 2013, San Diego, USA, 23-26 Sept. 2013.
- **S. R. Ahmadzadeh**, M. Leonetti, P. Kormushev, *"Online Direct Policy Search for Thruster Failure Recovery in Autonomous Underwater Vehicles,"* Proceeding of 6th International workshop on Evolutionary and Reinforcement Learning for Autonomous Robot System (**ERLARS** 2013), Taormina, Italy, 2-9 Sept. 2013.
- N. Jamali, P. Kormushev, **S. R. Ahmadzadeh**, D. G. Caldwell, *"Covariance Analysis as a Measure of Policy Robustness in Reinforcement Learning,"* In Proceedings MTS/IEEE International Conference **OCEANS** 2014, Taipei, Taiwan, 7-10 Apr. 2014.
- **S. R. Ahmadzadeh**, M. Leonetti, A. Carrera, M. Carreras, P. Kormushev, D. G. Caldwell, *"Online Discovery of AUV Control Policies to Overcome Thruster Failures,"* IEEE International Conference on Robotics and Automation, (**ICRA** 2014), Hong Kong, China, 31 May-7 June 2014.
- **S. R. Ahmadzadeh**, P. Kormushev, D. G. Caldwell, *"Multi-Objective Reinforcement Learning for AUV Thruster Failure Recovery,"* In IEEE Symposium on Adaptive

Dynamic Programming and Reinforcement Learning (**ADPRL** 2014), Proc. IEEE Symposium Series on Computational Intelligence (**SSCI** 2014), Florida, USA, 8-12 Dec, 2014.

Related to Chapter 4 - Visuospatial Skill Learning

- **S. R. Ahmadzadeh**, P. Kormushev, D. G. Caldwell, *"Visuospatial Skill Learning for Object Reconfiguration Tasks, "* in Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (**IROS** 2013), Tokyo, Japan, 3-8 Nov., pp. 685-691, 2013.
- **S. R. Ahmadzadeh**, P. Kormushev, D. G. Caldwell, *"Interactive Robot Learning of Visuospatial Skills, "* in Proceedings 16th IEEE International Conference on Advanced Robotics (**ICAR** 2013), Montevideo, Uruguay, 25-29 Nov., 2013.
- **S. R. Ahmadzadeh**, A. Paikan, F. Mastergiovanni, L. Natale, P. Kormushev, D. G. Caldwell, *"Learning Symbolic Representation of Actions from Human Demonstrations, "* in Proceedings IEEE International Conference on Robotics and Automation (**ICRA** 2015), Seattle, Washington, USA, 26-30 May, 2015.

Bibliography

[1] Edson Prestes, Joel Luis Carbonera, Sandro Rama Fiorini, Vitor AM Jorge, Mara Abel, Raj Madhavan, Angela Locoro, Paulo Goncalves, Marcos E. Barreto, Maki Habib *et al.*, "Towards a core ontology for robotics and automation," *Robotics and Autonomous Systems*, vol. 61, no. 11, pp. 1193–1204, 2013.

[2] George A. Bekey, *Robotics: state of the art and future challenges*. Imperial College Press, 2008.

[3] Marco Pavone, B. Acikmese, Issa A. Nesnas, Joseph Starek, M. Thoma, F. Allgöwer, and M. Morari, "Spacecraft autonomy challenges for next generation space missions," *Lecture Notes in Control and Information Systems*, 2014.

[4] David M. Lane, Francesco Maurelli, Petar Kormushev, Marc Carreras, Maria Fox, and Konstantinos Kyriakopoulos, "Persistent autonomy: the challenges of the PANDORA project," *Proceedings of IFAC MCMC*, 2012.

[5] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[6] Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue, "Learning by watching: Extracting reusable task knowledge from visual observation of human performance," *Robotics and Automation, IEEE Transactions on*, vol. 10, no. 6, pp. 799–822, 1994.

[7] Minoru Asada, Yuichiro Yoshikawa, and Koh Hosoda, "Learning by observation without three-dimensional reconstruction," *Intelligent Autonomous Systems (IAS-6)*, pp. 555–560, 2000.

[8] Manuel Lopes and José Santos-Victor, "Visual learning by imitation with motor representations," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 35, no. 3, pp. 438–449, 2005.

[9] Rodney A. Brooks, "A robust layered control system for a mobile robot," *Robotics and Automation, IEEE Journal of*, vol. 2, no. 1, pp. 14–23, 1986.

[10] Alessandro Saffiotti, "The uses of fuzzy logic in autonomous robot navigation," *Soft Computing*, vol. 1, no. 4, pp. 180–197, 1997.

[11] Jonathan Evans, Pedro Patrón, Ben Smith, and David M. Lane, "Design and evaluation of a reactive and deliberative collision avoidance and escape architecture for autonomous robots," *Autonomous Robots*, vol. 24, no. 3, pp. 247–266, 2008.

[12] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver, "Visual navigation for mobile robots: A survey," *Journal of intelligent and robotic systems*, vol. 53, no. 3, pp. 263–296, 2008.

[13] Eiichi Yoshida, Kazuhito Yokoi, and Pierre Gergondet, "Online replanning for reactive robot motion: Practical aspects," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 5927–5933.

[14] R.C. Luo, Meng-Chu Ko, Yi-Ting Chung, and R. Chatila, "Repulsive reaction vector generator for whole-arm collision avoidance of 7-DoF redundant robot manipulator," in *Advanced Intelligent Mechatronics (AIM), 2014 IEEE/ASME International Conference on*, July 2014, pp. 1036–1041.

[15] Mongi A. Abidi, Richard O. Eason, and Rafael C. Gonzalez, "Autonomous robotic inspection and manipulation using multisensor feedback," *Computer*, vol. 24, no. 4, pp. 17–31, 1991.

[16] David A. Anisi, Erik Persson, and Clint Heyer, "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, 2011, pp. 235–240.

[17] Matko Orsag, Christopher Korpela, Stjepan Bogdan, and Paul Oh, "Valve turning using a dual-arm aerial manipulator," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*. IEEE, 2014, pp. 836–841.

[18] Nicholas Alunni, C. Phillips-Grafftin, Halit Bener Suay, Daniel Lofaro, Dmitry Berenson, Sonia Chernova, Robert W. Lindeman, and Paul Oh, "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, 2013, pp. 1–6.

[19] Arash Ajoudani, Jinoh Lee, Alessio Rocchi, Mirko Ferrati, Enrico Mingo, Alessandro Settimi, Darwin G. Caldwell, Antonio Bicchi, and Nikolaos Tsagarakis, "A manipulation framework for compliant humanoid coman: Application to a valve turning task," in *Humanoid Robots (Humanoids), 2014 IEEE-RAS International Conference on*. IEEE, 2014, pp. 664–670.

[20] Arnau Carrera, Seyed Reza Ahmadzadeh, Arash Ajoudani, Petar Kormushev, Marc Carreras, and Darwin G. Caldwell, "Towards autonomous robotic valve turning," *Cybernetics and Information Technologies*, vol. 12, no. 3, 2012.

[21] Seyed Reza Ahmadzadeh, Petar Kormushev, and Darwin G. Caldwell, "Autonomous robotic valve turning: A hierarchical learning approach," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4614–4619.

[22] Seyed Reza Ahmadzadeh, Petar Kormushev, Rodrigo S. Jamisola, and Darwin G. Caldwell, "Learning reactive robot behavior for autonomous valve turning," in *Humanoid Robots (Humanoids), 2014 IEEE-RAS International Conference on*. Madrid,Spain: IEEE, November 2014, pp. 685–691.

[23] Thomas S. Wikman and Wyatt S. Newman, "A fast, on-line collision avoidance method for a kinematically redundant manipulator based on reflex control," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*. IEEE, 1992, pp. 261–266.

[24] Oussama Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[25] Kristin Glass, Richard Colbaugh, David Lim, and Homayoun Seraji, "Real-time collision avoidance for redundant manipulators," *Robotics and Automation, IEEE Transactions on*, vol. 11, no. 3, pp. 448–457, 1995.

[26] Yunong Zhang and Jun Wang, "Obstacle avoidance for kinematically redundant manipulators using a dual neural network," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 1, pp. 752–759, 2004.

[27] Alessandro De Luca, Alin Albu-Schaffer, Sami Haddadin, and Gerd Hirzinger, "Collision detection and safe reaction with the dlr-iii lightweight manipulator arm," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE, 2006, pp. 1623–1630.

[28] Eiichi Yoshida and Fumio Kanehiro, "Reactive robot motion using path replanning and deformation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5456–5462.

[29] David A. Anisi, Charlotte Skourup, and ABB Petrochemicals, "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, 2012.

[30] Jee-Hwan Ryu, Dong-Soo Kwon, and Pan-Mook Lee, "Control of underwater manipulators mounted on an ROV using base force information," in *Robotics and Automation (ICRA), 2001 IEEE International Conference on*, vol. 4. IEEE, 2001, pp. 3238–3243.

[31] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal, "Robot programming by demonstration," *Handbook of robotics*, vol. 1, 2008.

[32] Auke J. Ijspeert, Jun Nakanishi, and Stefan Schaal, "Trajectory formation for imitation with nonlinear dynamical systems," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 2. IEEE, 2001, pp. 752–757.

[33] Richard S. Sutton and Andrew G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.

[34] Petar Kormushev, Sylvain Calinon, and Darwin G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Advanced Robotics*, vol. 25, no. 5, pp. 581–603, 2011.

[35] Stefan Schaal, Auke Ijspeert, and Aude Billard, "Computational approaches to motor learning by imitation," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 358, no. 1431, pp. 537–547, 2003.

[36] Sylvain Calinon, Florent Guenter, and Aude Billard, "On learning, representing, and generalizing a task in a humanoid robot," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, no. 2, pp. 286–298, 2007.

[37] Sylvain Calinon and Aude Billard, "A probabilistic programming by demonstration framework handling constraints in joint space and task space," in *Intelligent Robots and Systems (IROS), 2008 IEEE/RSJ International Conference on*. IEEE, 2008, pp. 367–372.

[38] Benjamin Reiner, Wolfgang Ertel, Heiko Posenauer, and Markus Schneider, "Lat: A simple learning from demonstration method," in *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 4436–4441.

[39] Auke J. Ijspeert, Jun Nakanishi, and Stefan Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Robotics and Automation (ICRA), 2002 IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 1398–1403.

[40] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Robotics and Automation, 2009. (ICRA). IEEE International Conference on*. IEEE, 2009, pp. 763–768.

[41] Li-Xin Wang, *A Course on Fuzzy Systems*. Prentice-Hall press, USA, 1999.

[42] Tomohiro Takagi and Michio Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *Systems, Man and Cybernetics, IEEE Transactions on*, no. 1, pp. 116–132, 1985.

[43] Michio Sugeno and G.T. Kang, "Structure identification of fuzzy model," *Fuzzy sets and systems*, vol. 28, no. 1, pp. 15–33, 1988.

[44] Reuven Y. Rubinstein and Dirk P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer, 2004.

[45] Pieter-Tjerk De Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of operations research*, vol. 134, no. 1, pp. 19–67, 2005.

[46] Nikolaus Hansen, "The cma evolution strategy: a comparing review," in *Towards a new evolutionary computation*. Springer, 2006, pp. 75–102.

[47] P. Brachetti, M. De Felice Ciccoli, Gianni Di Pillo, and Stefano Lucidi, "A new version of the price's algorithm for global optimization," *Journal of Global Optimization*, vol. 10, no. 2, pp. 165–184, 1997.

[48] Alin Albu-Schäffer, Christian Ott, and Gerd Hirzinger, "A unified passivity-based control framework for position, torque and impedance control of flexible joint robots," *The International Journal of Robotics Research*, vol. 26, no. 1, pp. 23–39, 2007.

[49] Günter Schreiber, Andreas Stemmer, and Rainer Bischoff, "The fast research interface for the kuka lightweight robot," in *IEEE Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications How to Modify and Enhance Commercial Controllers (ICRA 2010)*, 2010.

[50] Marc H. Raibert and John J. Craig, "Hybrid position/force control of manipulators," *Journal of Dynamic Systems, Measurement, and Control*, vol. 103, no. 2, pp. 126–133, 1981.

[51] Oussama Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 1, pp. 43–53, 1987.

[52] Tsuneo Yoshikawa and Xin-Zhi Zheng, "Coordinated dynamic hybrid position/force control for multiple robot manipulators handling one constrained object," *The International Journal of Robotics Research*, vol. 12, no. 3, pp. 219–230, 1993.

[53] Samir N. Das and Samir K. Das, "Determination of coupled sway, roll, and yaw motions of a floating body in regular waves," *International Journal of Mathematics and Mathematical Sciences*, vol. 2004, no. 41, pp. 2181–2197, 2004.

[54] Rodrigo S. Jamisola, Denny N. Oetomo, Marcelo H. Ang, Oussama Khatib, Tao Ming Lim, and Ser Yong Lim, "Compliant motion using a mobile manipulator: an operational space formulation approach to aircraft canopy polishing," *Advanced Robotics*, vol. 19, no. 5, pp. 613–634, 2005.

[55] David Ribas, Narcıs Palomeras, Pere Ridao, Marc Carreras, and Angelos Mallios, "Girona 500 AUV: From survey to intervention," *Mechatronics, IEEE/ASME Transactions on*, vol. 17, no. 1, pp. 46–53, 2012.

[56] David G. Lowe, "Object recognition from local scale-invariant features," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2. IEEE, 1999, pp. 1150–1157.

[57] Seyed Reza Ahmadzadeh, Matteo Leonetti, and Petar Kormushev, "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, September 2013.

[58] Matteo Leonetti, Seyed Reza Ahmadzadeh, and Petar Kormushev, "On-line learning to recover from thruster failures on autonomous underwater vehicles," in *OCEANS 2013*. IEEE, 2013.

[59] Seyed Reza Ahmadzadeh, Matteo Leonetti, Arnau Carrera, Marc Carreras, Petar Kormushev, and Darwin G. Caldwell, "Online discovery of AUV control policies to overcome thruster failure," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. Hong Kong, China: IEEE, May 2014, pp. 6522–6528.

[60] Seyed Reza Ahmadzadeh, Petar Kormushev, and Darwin G. Caldwell, "Multi-objective reinforcement learning for AUV thruster failure recovery," in *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), 2014, Proc. IEEE Symposium Series on Computational Intelligence (SSCI)*. Florida, USA: IEEE, December 2014, pp. 1–8.

[61] Massimo Caccia, R. Bono, Ga Bruzzone, Gi Bruzzone, E. Spirandelli, and Gianmarco Veruggio, "Experiences on actuator fault detection, diagnosis and accomodation for ROVs," *International Symposiyum of Unmanned Untethered Sub-mersible Technol*, 2001.

[62] Kelvin Hamilton, Dave Lane, Nick Taylor, and Keith Brown, "Fault diagnosis on autonomous robotic vehicles with recovery: an integrated heterogeneous-knowledge approach," in *Robotics and Automation (ICRA), 2001 IEEE International Conference on*, vol. 4. IEEE, 2001, pp. 3232–3237.

[63] Gianluca 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.

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

[65] Tarun Kanti Podder, Gianluca Antonelli, and Nilanjan Sarkar, "Fault tolerant control of an autonomous underwater vehicle under thruster redundancy: Simulations and experiments," in *Robotics and Automation (ICRA), 2000 IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 1251–1256.

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

[67] Tor Arne Johansen and Thor I. Fossen, "Control allocation - a survey," *Automatica*, vol. 49, no. 5, pp. 1087–1103, 2013.

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

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

[70] Albert S. Cheng and Naomi Ehrich Leonard, "Fin failure compensation for an unmanned underwater vehicle," in *Proceedings of the 11th International Symposium on Unmanned Untethered Submersible Technology*. Citeseer, 1999.

[71] Mae 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.

[72] Michael Andonian, Dario Cazzaro, Luca Invernizzi, Monique Chyba, and Sergio 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.

[73] Jin-Kyu Choi and Hayato 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.

[74] Michael W. Oppenheimer, David B. Doman, and M. Bolender, "Control allocation," *The control handbook, control system applications*, 2010.

[75] Wayne C. Durham, "Constrained control allocation," *Journal of Guidance, Control, and Dynamics*, vol. 16, no. 4, pp. 717–725, 1993.

[76] John C. Virnig and David S. Bodden, "Multivariable control allocation and control law conditioning when control effectors limit(stovl aircraft)," in *AIAA Guidance, Navigation and Control Conference, Scottsdale, AZ*, 1994, pp. 572–582.

[77] Marc Bodson and Susan A. Frost, "Load balancing in control allocation," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 2, pp. 380–387, 2011.

[78] John AM Petersen and Marc Bodson, "Constrained quadratic programming techniques for control allocation," *Control Systems Technology, IEEE Transactions on*, vol. 14, no. 1, pp. 91–98, 2006.

[79] Dale Enns, "Control allocation approaches," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 1998, pp. 98–108.

[80] Marc Bodson, "Evaluation of optimization methods for control allocation," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 4, pp. 703–711, 2002.

[81] Tor Arne Johansen, Thor I. Fossen, and Stig P. Berge, "Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming," *Control Systems Technology, IEEE Transactions on*, vol. 12, no. 1, pp. 211–216, 2004.

[82] William C. Webster and Joao Sousa, "Optimum allocation for multiple thrusters," in *The Proceedings of the... International Offshore and Polar Engineering Conference*. International Society of Offshore and Polar Engineers, 1999.

[83] Thor I. Fossen and Tor A. Johansen, "A survey of control allocation methods for ships and underwater vehicles," in *Control and Automation, 2006. MED'06. 14th Mediterranean Conference on*. IEEE, 2006, pp. 1–6.

[84] Kristin Y. Pettersen and Olva Egeland, "Robust control of an underactuated surface vessel with thruster dynamics," in *American Control Conference, 1997. Proceedings of the 1997*, vol. 5. IEEE, 1997, pp. 3411–3415.

[85] Naomi Ehrich Leonard, "Compensating for actuator failures: Dynamics and control of underactuated underwater vehicles," in *International Symposium on Unmanned Untethered Submersible Technology*. University of New Hampshire-Marine Systems, 1995, pp. 168–177.

[86] Naomi Ehrich Leonard, "Periodic forcing, dynamics and control of underactuated spacecraft and underwater vehicles," in *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, vol. 4. IEEE, 1995, pp. 3980–3985.

[87] Monique Chyba, Thomas Haberkorn, Ryan N. Smith, and George R. Wilkens, "A geometrical analysis of trajectory design for underwater vehicles," *Discrete and Continuous Dynamical Systems-B*, vol. 11, no. 2, pp. 233–262, 2009.

[88] FY Bi, YJ Wei, Jingqing Zhang, and W. Cao, "Position-tracking control of underactuated autonomous underwater vehicles in the presence of unknown ocean currents," *Control Theory & Applications, IET*, vol. 4, no. 11, pp. 2369–2380, 2010.

[89] Petar V. Kokotovie, "The joy of feedback: nonlinear and adaptive," *IEEE Control Systems Magazine*, vol. 12, no. 3, pp. 7–17, 1992.

[90] Sylvain Koos, Antoine Cully, and Jean-Baptiste Mouret, "Fast damage recovery in robotics with the t-resilience algorithm," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1700–1723, 2013.

[91] Thor I. Fossen, "Guidance and control of ocean vehicles," *Wiley, New York*, 1994.

[92] George C. Karras, Charalampos P. Bechlioulis, Matteo Leonetti, Narcís Palomeras, Petar Kormushev, Kostas J. Kyriakopoulos, and Darwin G. Caldwell, "On-line identification of autonomous underwater vehicles through global derivative-free optimization," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 3859–3864.

[93] George Konidaris, Sarah Osentoski, and Philip Thomas, "Value function approximation in reinforcement learning using the fourier basis," in *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

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

[95] Scott Kirkpatrick, D. Gelatt Jr., and Mario P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[96] Rainer Storn and Kenneth 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.

[97] Mitsunori Miki, Tomoyuki Hiroyasu, and Keiko Ono, "Simulated annealing with advanced adaptive neighborhood," in *Second international workshop on Intelligent systems design and application*. Citeseer, 2002, pp. 113–118.

[98] Balram Suman and Prabhat 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.

[99] Price Kenneth, Storn Rainer, and A. Lampinen Jouni, "Differential evolution: a practical approach to global optimization," *Natural Computing Series*, 2005.

[100] Roger Gämperle, Sibylle D. Müller, and Petros Koumoutsakos, "A parameter study for differential evolution," *Advances in intelligent systems, fuzzy systems, evolutionary computation*, vol. 10, pp. 293–298, 2002.

[101] Swagatam Das and Ponnuthurai Nagaratnam Suganthan, "Differential evolution: A survey of the state-of-the-art," *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 1, pp. 4–31, 2011.

[102] Peter Vamplew, John Yearwood, Richard Dazeley, and Adam Berry, "On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts," in *AI 2008: Advances in Artificial Intelligence*. Springer, 2008, pp. 372–378.

[103] Zoltán Gábor, Zsolt Kalmár, and Csaba Szepesvári, "Multi-criteria reinforcement learning." in *ICML*, vol. 98, 1998, pp. 197–205.

[104] Shie Mannor and Nahum Shimkin, "A geometric approach to multi-criterion reinforcement learning," *The Journal of Machine Learning Research*, vol. 5, pp. 325–360, 2004.

[105] Sriraam Natarajan and Prasad Tadepalli, "Dynamic preferences in multi-criteria reinforcement learning," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 601–608.

[106] Daniel J. Lizotte, Michael H. Bowling, and Susan A. Murphy, "Efficient reinforcement learning with multiple reward functions for randomized controlled trial analysis," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 695–702.

[107] Christian Robert Shelton, "Importance sampling for reinforcement learning with multiple objectives," Ph.D. dissertation, Massachusetts Institute of technology MIT, Department of electerical Engineering and Computer Science, August 2001.

[108] Leon Barrett and Srini Narayanan, "Learning all optimal policies with multiple criteria," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 41–47.

[109] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32–49, 2011.

[110] B.V. Babu and M. Mathew Leenus Jehan, "Differential evolution for multi-objective optimization," in *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 4. IEEE, 2003, pp. 2696–2703.

[111] Hui Li and Qingfu Zhang, "A multiobjective differential evolution based on decomposition for multiobjective optimization with variable linkages," in *Parallel problem solving from nature-PPSN IX*. Springer, 2006, pp. 583–592.

[112] Hussein A. Abbass, Ruhul Sarker, and Charles Newton, "PDE: a pareto-frontier differential evolution approach for multi-objective optimization problems," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, vol. 2. IEEE, 2001, pp. 971–978.

[113] B.V. Babu and B. Anbarasu, "Multi-objective differential evolution (MODE): an evolutionary algorithm for multi-objective optimization problems (MOOPs)," in *Proceedings of International Symposium and 58th Annual Session of IIChE*. Citeseer, 2005.

[114] Valerio De Carolis, David Lane, and Keith Brown, "Low-cost energy measurement and estimation for autonomous underwater vehicle," in *Proceedings of IEEE-MTS OCEANS'14, Taipei, Taiwan*, 2014.

[115] Michael Kaiser and Rüdiger Dillmann, "Building elementary robot skills from human demonstration," in *Robotics and Automation (ICRA), 1996 IEEE International Conference on*. IEEE, 1996, pp. 2700–2705.

[116] Auke J. Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[117] Auke J. Ijspeert, Jun Nakanishi, and Stefan Schaal, "Learning attractor landscapes for learning motor primitives," *Advances in neural information processing systems*, vol. 15, pp. 1523–1530, 2002.

[118] Petar Kormushev, Sylvain Calinon, and Darwin G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Advanced Robotics*, vol. 25, no. 5, pp. 581–603, 2011.

[119] Sethu Vijayakumar and Stefan Schaal, "Locally weighted projection regression: An o(n) algorithm for incremental real time learning in high dimensional space," in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, vol. 1, 2000, pp. 288–293.

[120] Pieter Abbeel and Andrew Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 1.

[121] Klas Kronander and Aude Billard, "Online learning of varying stiffness through physical human-robot interaction," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1842–1849.

[122] Darrin C. Bentivegna, Christopher G. Atkeson, Aleš Ude, and Gordon Cheng, "Learning to act from observation and practice," *International Journal of Humanoid Robotics*, vol. 1, no. 04, pp. 585–611, 2004.

[123] Petar Kormushev, Sylvain Calinon, and Darwin G. Caldwell, "Robot motor skill coordination with EM-based reinforcement learning," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 3232–3237.

[124] B.G. Burdea and H.J. Wolfson, "Solving jigsaw puzzles by a robot," *Robotics and Automation, IEEE Transactions on*, vol. 5, no. 6, pp. 752–764, 1989.

[125] Scott Niekum, Sarah Osentoski, George Konidaris, and Andrew G. Barto, "Learning and generalization of complex tasks from unstructured demonstrations," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 5239–5246.

[126] Deepak Verma and Rajesh Rao, "Goal-based imitation as probabilistic inference over graphical models," *Advances in neural information processing systems*, vol. 18, pp. 1393–1400, 2005.

[127] Neil Dantam, Irfan Essa, and Mike Stilman, "Linguistic transfer of human assembly tasks to robots," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 237–242.

[128] Crystal Chao, Maya Cakmak, and Andrea L. Thomaz, "Towards grounding concepts for transfer in goal learning from demonstration," in *Development and Learning (ICDL), 2011 IEEE International Conference on*, vol. 2. IEEE, 2011, pp. 1–6.

[129] Seyed Reza Ahmadzadeh, Petar Kormushev, and Darwin G. Caldwell, "Visuospatial skill learning for object reconfiguration tasks," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 685–691.

[130] Seyed Reza Ahmadzadeh, Petar Kormushev, and Darwin. G. Caldwell, "Interactive robot learning of visuospatial skills," in *Advanced Robotics (ICAR) 2013, 16th International Conference on*. IEEE, 2013, pp. 1–8.

[131] Seyed Reza Ahmadzadeh, Ali Paikan, Fulvio Mastrogiovanni, Lorenzo Natale, Petar Kormushev, and Darwin G. Caldwell, "Learning symbolic representations of actions from human demonstrations," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. Seattle, Washington, USA: IEEE, May 2015.

[132] Andrew N. Meltzoff and M. Keith Moore, "Imitation of facial and manual gestures by human neonates," *Science*, vol. 198, no. 4312, pp. 75–78, 1977.

[133] Andrew N. Meltzoff and M. Keith Moore, "Newborn infants imitate adult facial gestures," *Child development*, pp. 702–709, 1983.

[134] Giacomo Rizzolatti, Luciano Fadiga, Vittorio Gallese, and Leonardo Fogassi, "Premotor cortex and the recognition of motor actions," *Cognitive brain research*, vol. 3, no. 2, pp. 131–141, 1996.

[135] Stefan Schaal, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[136] GaLam Park, Syungkwon Ra, ChangHwan Kim, and JaeBok Song, "Imitation learning of robot movement using evolutionary algorithm," in *Proceedings of the 17th World Congress, International Federation of Automatic Control (IFAC)*, 2008, pp. 730–735.

[137] Takashi Suehiro and Katsushi Ikeuchi, "Towards an assembly plan from observation: Part ii: Correction of motion parameters based on fact contact constraints," in *Intelligent Robots and Systems (IROS), 1992 IEEE/RSJ International Conference on*, vol. 3. IEEE, 1992, pp. 2095–2102.

[138] Katsushi Ikeuchi and Takashi Suehiro, "Toward an assembly plan from observation. i. task recognition with polyhedral objects," *Robotics and Automation, IEEE Transactions on*, vol. 10, no. 3, pp. 368–385, 1994.

[139] Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue, "Seeing, understanding and doing human task," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*. IEEE, 1992, pp. 2–9.

[140] Markus Ehrenmann, Oliver Rogalla, Raoul Zöllner, and Rüdiger Dillmann, "Teaching service robots complex tasks: Programming by demonstration for workshop and household environments," in *Proceedings of the 2001 International Conference on Field and Service Robots (FSR)*, vol. 1, 2001, pp. 397–402.

[141] Mohammed Yeasin and Subhasis Chaudhuri, "Toward automatic robot programming: learning human skill from visual data," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 30, no. 1, pp. 180–185, 2000.

[142] Michael Pardowitz, R. Zollner, and Rüdiger Dillmann, "Incremental acquisition of task knowledge applying heuristic relevance estimation," in *Robotics and Automation (ICRA), 2006 IEEE International Conference on*. IEEE, 2006, pp. 3011–3016.

[143] Norbert Krüger, Justus Piater, Florentin Wörgötter, Christopher Geib, Ron Petrick, Mark Steedman, Aleš Ude, Tamim Asfour, Dirk Kraft, Damir Omrcen *et al.*, "A formal definition of object-action complexes and examples at different levels of the processing hierarchy," *PACO-PLUS Technical Report*, 2009.

[144] Staffan Ekvall and Danica Kragic, "Learning task models from multiple human demonstrations," in *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*. IEEE, 2006, pp. 358–363.

[145] Staffan Ekvall and Danica Kragic, "Robot learning from demonstration: a task-level planning approach," *International Journal of Advanced Robotic Systems*, vol. 5, no. 3, pp. 223–234, 2008.

[146] Dave Golland, Percy Liang, and Dan Klein, "A game-theoretic approach to generating spatial descriptions," in *Proceedings of the 2010 conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2010, pp. 410–419.

[147] Martin Mason and Manuel Lopes, "Robot self-initiative and personalization by learning through repeated interactions," in *Human-Robot Interaction (HRI), 2011 6th ACM/IEEE International Conference on*. IEEE, 2011, pp. 433–440.

[148] Oliver Kroemer and Jan Peters, "A flexible hybrid framework for modeling complex manipulation tasks," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1856–1861.

[149] Eren Erdal Aksoy, Alexey Abramov, Johannes Dörr, Kejun Ning, Babette Dellen, and Florentin Wörgötter, "Learning the semantics of object–action relations by observation," *The International Journal of Robotics Research*, pp. 1–29, 2011.

[150] Mohamad Javad Aein, Eren Erdal Aksoy, Minija Tamosiunaite, Jeremie Papon, Ales Ude, and F Worgotter, "Toward a library of manipulation actions based on semantic object-action relations," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 4555–4562.

[151] Neil Dantam, P. Koine, and Mike Stilman, "The motion grammar for physical human-robot games," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5463–5469.

[152] Neil Dantam and Mike Stilman, "The motion grammar: Linguistic perception, planning, and control," *Robotics: Science and Systems VII*, p. 49, 2012.

[153] Chavdar Papazov, Sami Haddadin, Sven Parusel, Kai Krieger, and Darius Burschka, "Rigid 3D geometry matching for grasping of known objects in cluttered scenes," *The International Journal of Robotics Research*, vol. 31, no. 4, pp. 538–553, 2012.

[154] Sergio Guadarrama, Lorenzo Riano, Dave Golland, Daniel Gouhring, Yangqing Jia, Dan Klein, Pieter Abbeel, and Trevor Darrell, "Grounding spatial relations for human-robot interaction," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 1640–1647.

[155] Scott Niekum, Sachin Chitta, Andrew G. Barto, Bhaskara Marthi, and Sarah Osentoski, "Incremental semantically grounded learning from demonstration." in *Robotics: Science and Systems*, vol. 9, 2013.

[156] Ashley Feniello, Hao Dang, and Stan Birchfield, "Program synthesis by examples for object repositioning tasks," in *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 4428–4435.

[157] Wolfram Burgard, Armin B. Cremers, Dieter Fox, Dirk Hähnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun, "Experiences with an interactive museum tour-guide robot," *Artificial intelligence*, vol. 114, no. 1, pp. 3–55, 1999.

[158] John D. Kelleher, Geert-Jan M. Kruijff, and Fintan J. Costello, "Proximity in context: an empirically grounded computational model of proximity for processing topological spatial expressions," in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2006, pp. 745–752.

[159] Reinhard Moratz and Thora Tenbrink, "Spatial reference in linguistic human-robot interaction: Iterative, empirically supported development of a model of projective relations," *Spatial cognition and computation*, vol. 6, no. 1, pp. 63–107, 2006.

[160] Marjorie Skubic, Dennis Perzanowski, Sam Blisard, Alan Schultz, William Adams, Magda Bugajska, and Derek Brock, "Spatial language for human-robot dialogs," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 34, no. 2, pp. 154–167, 2004.

[161] Richard Hartley and Andrew Zisserman, *Multiple view geometry in computer vision*. Cambridge Univ Press, 2000, vol. 2.

[162] David G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[163] Tzu Yen Wong, Peter Kovesi, and Amitava Datta, "Projective transformations for image transition animations," in *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*. IEEE, 2007, pp. 493–500.

[164] Jake K Aggarwal and Michael S Ryoo, "Human activity analysis: A review," *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, p. 16, 2011.

[165] Karinne Ramirez-Amaro, Michael Beetz, and Gordon Cheng, "Automatic segmentation and recognition of human activities from observation based on semantic reasoning," in *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 5043–5048.

[166] Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic, "Data-driven grasp synthesis: A survey," *Robotics, IEEE Transactions on*, vol. 30, pp. 289–309, 2014.

[167] Yanyu Su, Yan Wu, Kyuhwa Lee, Zhijiang Du, and Yiannis Demiris, "Robust grasping for an under-actuated anthropomorphic hand under object position uncertainty," in *Humanoid Robots (Humanoids), 2012 IEEE-RAS International Conference on*, Nov 2012, pp. 719–725.

[168] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Robotics and Automation, 2009. (ICRA). IEEE International Conference on*. IEEE, 2009, pp. 3212–3217.

[169] Bastian Steder, Radu Bogdan Rusu, Kurt Konolige, and Wolfram Burgard, "NARF: 3D range image features for object recognition," in *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 44, 2010.

[170] Anders Glent Buch, Dirk Kraft, Joni-Kristian Kämäräinen, Henrik Gordon Petersen, and Norbert Krüger, "Pose estimation using local structure-specific shape and appearance context," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2080–2087.

[171] Yixin Chen, Chih-Wei Hsu, and Benjamin W Wah, "Sgplan: Subgoal partitioning and resolution in planning," *Edelkamp et al.(Edelkamp, Hoffmann, Littman, & Younes, 2004)*, 2004.

[172] Nils J. Nilsson, "Shakey the robot," DTIC Document, Tech. Rep., 1984.

[173] Nikolay Jetchev, Tobias Lang, and Marc Toussaint, "Learning grounded relational symbols from continuous data for abstract reasoning," 2013.

[174] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez, "Constructing symbolic representations for high-level planning," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[175] Malik Ghallab, Craig Knoblock, David Wilkins, Anthony Barrett, Dave Christianson, Marc Friedman, Chung Kwok, Keith Golden, Scott Penberthy, David E Smith *et al.*, "PDDL-the planning domain definition language," Yale Center for Computational Vision and Control, Tech Report CVC TR-98-003/DCS TR-1165, October 1998.

[176] David A. Randell, Zhan Cui, and Anthony G. Cohn, "A spatial logic based on regions and connection." *Knowledge Representation and Reasoning*, vol. 92, pp. 165–176, 1992.

[177] Anne Auger and Nikolaus Hansen, "A restart cma evolution strategy with increasing population size," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 2. IEEE, 2005, pp. 1769–1776.

[178] Vladimír Černỳ, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *Journal of optimization theory and applications*, vol. 45, no. 1, pp. 41–51, 1985.

[179] Angelo Corana, Michele Marchesi, Claudio Martini, and Sandro Ridella, "Minimizing multimodal functions of continuous variables with the "simulated annealing" algorithm corrigenda for this article is available here," *ACM Transactions on Mathematical Software (TOMS)*, vol. 13, no. 3, pp. 262–280, 1987.

[180] Renato S. Tavares, Thiago C. Martins, and Marcos S.G. Tsuzuki, "Simulated annealing with adaptive neighborhood: A case study in off-line robot path planning," *Expert Systems with Applications*, vol. 38, no. 4, pp. 2951–2965, 2011.

[181] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, *Numerical recipes in C+: the art of scientific computing*. Cambridge University Press Cambridge, 2009, vol. 994.

[182] Lester Ingber, "Very fast simulated re-annealing," *Mathematical and computer modelling*, vol. 12, no. 8, pp. 967–973, 1989.

[183] Lester Ingber and Bruce Rosen, "Genetic algorithms and very fast simulated reannealing: A comparison," *Mathematical and Computer Modelling*, vol. 16, no. 11, pp. 87–100, 1992.

[184] Daniel A. Liotard, "Algorithmic tools in the study of semiempirical potential surfaces," *International journal of quantum chemistry*, vol. 44, no. 5, pp. 723–741, 1992.

[185] Erik Sundermann and Ignace Lemahieu, "Pet image reconstruction using simulated annealing," in *Proceedings of the SPIE Medical Imaging'95 (Image Processing)*, 1995, pp. 378–386.

[186] Lester Ingber, "Trading markets with canonical momenta and adaptive simulated annealing," Lester Ingber, Tech. Rep., 1996.

[187] Rob A. Rutenbar, "Simulated annealing algorithms: An overview," *Circuits and Devices Magazine, IEEE*, vol. 5, no. 1, pp. 19–26, 1989.

[188] Horacio Martínez-Alfaro, Homero Valdez, and Jaime Ortega, "Linkage synthesis of a four-bar mechanism for n precision points using simulated annealing," in *Asme Design Engineering Technical Conferences/25th Biennial Mecanisms and Robotics Conference. Atlanta.*, 1998.

[189] Horacio Martinez-Alfaro and Donald R. Flugrad, "Collision-free path planning for mobile robots and/or AGVs using simulated annealing," in *Systems, Man, and Cybernetics, 1994. Humans, Information and Technology, 1994 IEEE International Conference on*, vol. 1. IEEE, 1994, pp. 270–275.

[190] Alejandro Quintero and Samuel Pierre, "Assigning cells to switches in cellular mobile networks: a comparative study," *Computer Communications*, vol. 26, no. 9, pp. 950–960, 2003.

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

[192] Stefano Lucidi and Marco Sciandrone, "On the global convergence of derivative-free methods for unconstrained optimization," *SIAM Journal on Optimization*, vol. 13, no. 1, pp. 97–116, 2002.

ISTITUTO ITALIANO DI TECNOLOGIA
DEPARTMENT OF ADVANCED ROBOTICS
Via Morego, 30
16163 Genova
www.iit.it