THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Robot Learning for Deformable Object Manipulation Tasks

Exploring machine learning techniques for deformable linear object tasks: insights from shape-servoing and cable-routing

Rita Laezza

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2024

Robot Learning for Deformable Object Manipulation Tasks

Exploring machine learning techniques for deformable linear object tasks: insights from shape-servoing and cable-routing

Rita Laezza
ISBN 978-91-8103-030-3

Acknowledgments, dedications, and similar personal statements in this thesis, reflect the author's own views.

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 5488
ISSN 0346-718X

Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31 772 1000

Cover:
Image of an ABB Dual-Arm YuMi® robot manipulating two deformable linear objects to write the letters RL (Robot Learning) on a cable-routing table. A special thanks goes to Finn, who provided the CAD models used to make it.

Printed by Chalmers Reproservice
Gothenburg, Sweden, March 2024

To my dearly missed grandmothers,
avó Zezita *and* ***nonna Tilde****.*

Robot Learning for Deformable Object Manipulation Tasks

Exploring machine learning techniques for deformable linear object tasks: insights from shape-servoing and cable-routing

Rita Laezza
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Deformable Object Manipulation (DOM) is a challenging problem in robotics. Until recently, there has been limited research on the subject, with most robotic manipulation methods being developed with rigid objects in mind. Part of the challenge in DOM is that non-rigid objects require algorithms capable of generalizing to changes in shape as well as different mechanical properties. Machine Learning (ML) has been shown successful in fields, such as computer vision and natural language processing, where generalization is important thus encouraging the application of ML to robotic manipulation.

This thesis tackles DOM problems using ML techniques for tasks with Deformable Linear Objects (DLOs), e.g. ropes and cables, found in a variety of industrial applications. DLOs encapsulate a lot of the general challenges in DOM, making them good case studies on the effectiveness of ML for other types of deformable objects. Typically, ML algorithms require large amounts of data that are better satisfied in simulation. Therefore, the **ReForm** simulation sandbox is introduced, which includes six DLO manipulation tasks. ReForm aims to facilitate comparison and reproducibility of robot learning research on tasks where the goal is to control the shape of a DLO. Such shape control tasks are categorized as: *explicit*, if a precise shape is to be achieved; or *implicit*, if its deformation is dictated by a more abstract goal.

Two representative DLO manipulation tasks are addressed: **(i) shape-servoing** (explicit) and **(ii) cable-routing** (implicit). For shape-servoing, special emphasis is given to Reinforcement Learning (RL) methods. Initial work tackles shape-servoing of an elastoplastic DLO towards a unique goal, using online RL with ReForm. Subsequent work moves towards a multi-goal task in a real-world experimental setup, using offline RL methods to learn directly from real data. In the cable-routing works, the aim is to lay the groundwork for solving this type of task through motion primitives, with limited use of ML. First, a vision-based approach is presented, which is able to route a cable through randomly placed fixtures. Then, a force-based approach is introduced for a similar problem, in which the state and stiffness of a DLO can be estimated through contact with fixtures.

Keywords: Robotics, Machine Learning, Reinforcement Learning, Robot Learning, Deformable Object Manipulation, Deformable Linear Objects.

List of Publications

This thesis is based on the work contained in the following papers:

[A] **Rita Laezza**, Robert Gieselmann, Florian T. Pokorny, Yiannis Karayiannidis, "ReForm: A Robot Learning Sandbox for Deformable Linear Object Manipulation". 2021 IEEE International Conference on Robotics and Automation (ICRA).

[B] **Rita Laezza**, Yiannis Karayiannidis, "Learning Shape Control of Elastoplastic Deformable Linear Objects". 2021 IEEE International Conference on Robotics and Automation (ICRA).

[C] **Rita Laezza**, Mohammadreza Shetab-Bushehri, Gabriel A. Waltersson, Erol Özgür, Youcef Mezouar, Yiannis Karayiannidis, "Offline Goal-Conditioned Reinforcement Learning for Shape Control of Deformable Linear Objects". *Submitted to IEEE Robotics and Automation Letters (RA-L).*

[D] Gabriel A. Waltersson, **Rita Laezza**, Yiannis Karayiannidis, "Planning and Control for Cable-Routing with Dual-Arm Robot". 2022 IEEE International Conference on Robotics and Automation (ICRA).

[E] Finn Süberkrüb, **Rita Laezza**, Yiannis Karayiannidis, "Feel the Tension: Manipulation of Deformable Linear Objects in Environments with Fixtures using Force Information". 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Other papers by the author, not included in this thesis, are:

[F] **R. Laezza**, Y. Karayiannidis, "Shape Control of Elastoplastic Deformable Linear Objects through Reinforcement Learning". *IROS Workshop on Robotic Manipulation of Deformable Objects (ROMADO)*, October 25-29, 2020, Las Vegas, NV, USA (Virtual).

[G] **R. Laezza**, R. Gieselmann, F. T. Pokorny, Y. Karayiannidis, "Presenting ReForm: A Robot Learning Sandbox for Deformable Linear Object Manipulation". *ICRA Workshop on Representing and Manipulating Deformable Objects*, May 31 - June 7, 2021, Xi'an, China (Hybrid).

[H] **R. Laezza**, F. Süberkrüb, Y. Karayiannidis, "Blind Manipulation of Deformable Linear Objects Based on Force Information from Environmental Contacts". *ICRA Workshop on Representing and Manipulating Deformable Objects*, May 23-27, 2022, Philadelphia, USA.

[I] **R. Laezza**, M. Shetab-Bushehri, E. Özgür, Y. Mezouar, and Y. Karayiannidis, "Offline Reinforcement Learning for Shape Control of Deformable Linear Objects from Limited Real Data". *ICRA Workshop on Representing and Manipulating Deformable Objects*, May 29 - June 2, 2023, London, UK.

[J] K. Freitag, **R. Laezza**, J. Zbinden, and M. Ortiz-Catalan, "Improving Bionic Limb Control through Reinforcement Learning in an Interactive Game Environment". *ICML Workshop on Interactive Learning with Implicit Human Feedback*, July 23-29, 2023, Honolulu, USA.

[K] K. Freitag, Y. Karayiannidis, J. Zbinden, **R. Laezza**, "Fine-tuning Myoelectric Control through Reinforcement Learning in a Game Environment". *Submitted to IEEE Transactions on Biomedical Engineering.*

[L] A. Hannius, **R. Laezza**, J. Zbinden, "Towards Pose Invariant Bionic Limb Control: A Comparative Study of Two Unsupervised Domain Adaptation Methods". *Submitted to 2024 IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob).*

This PhD thesis is built upon the previously published Licentiate thesis: [1].

Acknowledgments

I have often heard that **robotics research is a team sport**. This has proven to be true throughout my PhD journey. Given the interdisciplinary nature of the field, it really does require good collaborations to create meaningful contributions. Therefore, I would like to begin by expressing my deepest gratitude to our team leader, Yiannis Karayiannidis, for his warm guidance and support during my doctoral studies. Next, I would like to acknowledge my co-supervisor, Florian T. Porkorny, and my collaborator, Robert Gieselmann, from KTH. Together with me and Yiannis, we formed a collaboration project within the Wallenberg AI, Autonomous Systems and Software Program (WASP). Hence, I am thankful towards the Knut and Alice Wallenberg Foundation that provides funding for this program. I would also like to express my gratitude towards Gabriel A. Waltersson and Finn Süberkrüb who worked with us at Chalmers for their Master's thesis projects and helped extend our research much further. Finally, I want to thank my greatest collaborator, Reza Shetab-Bushehri, who worked with me on our final and most challenging project, which would have been impossible without him.

From WASP, I want to acknowledge the affiliated Professors involved in the graduate school courses and trips, who provided invaluable knowledge and enabled wonderful experiences that I will always cherish. I would also like to thank my fellow WASP students with whom I've shared these wonderful experiences and some of the most interesting and entertaining discussions. I would further like to thank Algoryx, Anders and Kenneth in particular, who supported my research with free access to AGX Dynamics. I'm happy that they have since become industrial partners in the WASP Research Arena (WARA) for Robotics. In addition, I would like to thank Ludvig and Kristofer, who first connected me with Algoryx.

From Chalmers, I am grateful to have the opportunity to show appreciation for all of my friends and colleagues that made going to work every day a joyful experience. Doing a PhD can be both stressful and isolating, so having such a wonderful group of people to share the ups and downs of research and add some more life into the work-life equation, was truly invaluable. There are too many people to list here, so to everyone that walked the fifth floor halls in the east wing of the labyrinthic EDIT building with me, know that you are included. There are however some that stand out so much, that I cannot help but to mention them by name. My two academic big brothers, Ramin

and Albin, who were there for me at the start. Gabriel, who after his MSc thesis continued to do a PhD, and with whom I've had the pleasure of sharing an office ever since. Lorenzo, who joined us in this final year. The lovely *seventh floor people*: Laura, Alejandra, Yara, Mattia and Moein, that met us halfway in the sixth floor kitchen, to share some laughs in the middle of the day. Constantin, Alvin and Ahmet, whose projects have also ventured into either reinforcement learning or robotics, leading to interesting exchanges of ideas. Finally, so many constellations of us have spent great times together outside of Chalmers: from playing padel with Samuel, José and Alejandra, to playing tennis with Sten Elling, Maxi and Oguz; from climbing with Gabriel, Yara, Alvin, Sabino, Endre, Lorenzo and Rémi, to swimming with Alejandra, Laura, Erik, Ben and Sherry. These will be the memories that I take with me.

During my time at Chalmers, research was the primary but not my only focus. Teaching was a big part of my experience and I would like to thank the Professors that led the courses I was involved in, who also played an important role in my development, namely Bo, Torsten, Yiannis and Dean. Also a special shout-out goes to my fellow teaching assistants, Albert and Godwin. Further, I had the honor to serve two years on the PhD student council of the Electrical Engineering department, representing the Mechatronics group. This was one of the most rewarding experiences during my journey, and it has been a pleasure working together with my fellow PhD students towards improving our shared graduate venture. In this last year, I have also represented the department on the Chalmers board of graduate students. For both of these roles, I would like to thank everyone that worked along side me.

By the end of my studies, I started collaborating with the Center for Bionics and Pain Research (CBPR), in an attempt to move once more towards healthcare related fields. As a Biomedical Engineer, my interest in robotics has always been motivated by its potential for medical applications. I am happy to have been able to do exactly that, together with Jan, Kilian and Alexander. More specifically, we worked towards improving bionic limb control, thus returning to my Master's thesis topic. For this, I must also thank Max and Petter who financially supported this collaboration.

To my family, I am eternally grateful for their loving support. To my sister, Nadia, and my wonderful niece, Sofia, that bring so much joy into my life. To my mom and dad, Alda and Salvatore, who made everything possible. Lastly, I want to thank my dear Karl for going along this journey with me.

Acronyms

AI:	Artificial Intelligence
AL:	Apprenticeship Learning
ANN:	Artificial Neural Network
ARAP:	As-Rigid-As-Possible
BC:	Behavior Cloning
CNN:	Convolutional Neural Network
CV:	Computer Vision
DDOD:	Dense Depth Object Descriptor
DDPG:	Deep Deterministic Policy Gradient
DL:	Deep Learning
DLO:	Deformable Linear Object
DNN:	Deep Neural Network
DoF:	Degree of Freedom
DOM:	Deformable Object Manipulation
DP:	Dynamic Programming
DRL:	Deep Reinforcement Learning
FT:	Force-Torque
GA:	Genetic Algorithm
GCRL:	Goal-Conditioned Reinforcement Learning
HER:	Hindsight Experience Replay
HQP:	Hierarchical Quadratic Programming

IK:	Inverse Kinematics
IL:	Imitation Learning
LfD:	Learning from Demonstrations
MC:	Monte Carlo
MDP:	Markov Decision Process
ML:	Machine Learning
MLP:	Multilayer Perceptron
NLP:	Natural Language Processing
PG:	Policy Gradient
RGB-D:	Red Green Blue - Depth
RL:	Reinforcement Learning
RoL:	Robot Learning
SL:	Supervised Learning
SPR:	Structure Preserved Registration
TD:	Temporal Difference
TD3:	Twin Delayed DDPG

Contents

Abstract ii

List of Publications iii

Acknowledgments v

Acronyms vii

I Overview 1

1 Introduction 3
- 1.1 Deformable Object Manipulation . . . 5
- 1.2 Robot Learning . . . 7
- 1.3 Research Objective . . . 10
- 1.4 Thesis Outline . . . 11

2 Deformable Linear Object Manipulation 13
- 2.1 Modeling . . . 14
  - Geometric Modeling . . . 14
  - Physical Modeling . . . 19
- 2.2 Simulation . . . 23

2.3 Sensing . . . 25
2.4 Planning . . . 26
2.5 Control . . . 27
2.6 Representative Tasks . . . 28
Shape-Servoing . . . 29
Cable-Routing . . . 31

3 Reinforcement Learning **33**
3.1 MDP Formulation . . . 36
3.2 Value Function Approximation . . . 39
3.3 Policy Approximation . . . 41
3.4 Deep Reinforcement Learning . . . 47
Deep Deterministic Policy Gradient . . . 49
Twin-Delayed DDPG . . . 51
3.5 Offline RL . . . 53
3.6 RL Simulation Environments . . . 55

4 Research Contributions **57**
4.1 Summary of Publications . . . 58
Paper A . . . 59
Paper B . . . 60
Paper C . . . 61
Paper D . . . 62
Paper E . . . 63

5 Concluding Remarks **65**
5.1 Reflection . . . 66
5.2 Future Work . . . 69

References **71**

II Publications **85**

A ReForm: A Robot Learning Sandbox for DLO Manipulation **A1**
1 Introduction . . . A3
2 Related Work . . . A5
3 ReForm . . . A7

4 Manipulation Tasks . . . A12
5 Benchmarking Experiments . . . A13
6 Conclusion . . . A17
References . . . A17

B Learning Shape Control of Elastoplastic DLOs B1
1 Introduction . . . B3
2 Related Work . . . B6
3 Background . . . B7
3.1 Reinforcement Learning . . . B7
3.2 DLO Simulation . . . B9
4 Problem Statement . . . B10
5 Shape Representation . . . B12
6 RL Formulation . . . B14
7 Experimental Results . . . B15
8 Concluding Remarks . . . B20
References . . . B20

C Offline Goal-Conditioned RL for Shape Control of DLOs C1
1 Introduction . . . C3
2 Related Work . . . C6
3 Problem Statement . . . C7
4 Methods . . . C10
4.1 DLO Tracking . . . C10
4.2 Controller . . . C11
4.3 Offline Goal-Conditioned Reinforcement Learning . . . C13
5 Experiments . . . C15
5.1 Experimental Setup . . . C16
5.2 Data Augmentation . . . C18
5.3 BC Regularization . . . C21
5.4 Material Properties . . . C21
5.5 Limitations . . . C22
6 Conclusions . . . C22
References . . . C23

D Planning and Control for Cable-Routing D1
1 Introduction . . . D3

2 Related Work D5
3 Overview D8
4 Planning D9
4.1 Roadmap and Tasks D10
4.2 Stochastic Replanning D10
5 Control D16
5.1 Individual Manipulation D16
5.2 Coordinated Manipulation D17
5.3 Feasibility Objectives and HQP D18
6 Experiments D18
7 Concluding Remarks D21
References D22

E Feel the Tension: Manipulation of DLOs with Fixtures **E1**
1 Introduction E3
2 Related Work E5
3 Problem Statement E6
4 Model Representation E8
4.1 Feature Points, V E8
4.2 Edges, E E9
5 Model Estimation E11
5.1 Feature Point Estimation E11
5.2 Edge Elasticity Estimation E12
5.3 Model Update by Estimates E12
6 Manipulation Primitives and Control E13
7 Experimental Results E16
7.1 Elasticity Parameter Estimation and DLO Tensioning . E16
7.2 Contact Point Estimation E17
7.3 Manipulation Primitive Evaluation E19
7.4 Harness Production and Model Verification E20
8 Conclusion E21
References E21

Part I

Overview

CHAPTER 1

Introduction

At present, most robots are confined to industrial settings where their environment is highly controlled and the objects they manipulate are mostly rigid. Humans are not even allowed near these robots, since they are programmed to operate in a predefined manner and will not perceive anyone in their path, leading to injuries. Over the past couple of decades[1], the field of collaborative robotics has started to gain momentum, striving for closer interaction between humans and robots. As this cooperation becomes more common, robots are beginning to spread into other sectors, such as healthcare, agriculture and the service industry. Each sector adds to the variability in tasks and the objects being manipulated, thus leading to new challenges. This thesis addresses a subset of these challenges, namely: Deformable Object Manipulation (DOM).

The ability to handle non-rigid objects is particularly important since so many human tasks require skilled manipulation of deformable materials. This includes tasks like suturing in healthcare; harvesting of crops in agriculture; and cooking in the service industry. Note that in each of these examples,

[1] The first collaborative robot (or cobot) can be traced back to 1996, and was developed by a couple of professors at Northwestern University. The ABB YuMi robot depicted in the cover was the first dual-arm cobot, and was released in 2015.

there is a considerable amount of variation both in the objects being handled and in the techniques for executing a certain type of task. Looking closer at the example of surgical suturing, there are differences in tissue properties between individuals, as well as within a single individual, e.g. suturing vessels is significantly different from suturing skin.

Often, when faced with the job of automating a DOM task, engineers design specialized tools or machines. In agriculture, automated harvesting systems are examples of such machines. Even though this leads to efficient results, it requires a unique apparatus for each task. In contrast, humans can learn to execute a large set of tasks with their own hands. One goal of robotics research is therefore to create general purpose collaborative robots which are capable of dexterous manipulation akin to that of humans. Contrary to industrial robots which are intended for repetitive production at superhuman speed, strength and precision, collaborative robots have to operate in a safe, gentle and adaptive manner.

Robotic manipulation tasks are often solved by first deriving a model of the object and environment, to then use these for the design of a controller. However, deformable objects have complex nonlinear dynamics, which make modeling more challenging. Furthermore, due to the wide range of both mechanical and geometrical properties, these objects constitute a large and heterogeneous class. Taking the example of cooking, one would need to derive a model of each food item being manipulated. Alternatively, Machine Learning (ML) methods can be used to obtain modeling parameters or even to learn a control policy without the need for an explicit model. Furthermore, ML can be employed either to imitate human manipulation or to go beyond that by learning directly through experience. These methods fall under the umbrella of Robot Learning (RoL), and this thesis aims to explore how they can be used to solve DOM problems.

In summary, DOM is the application area of interest and RoL is the family of methods which are applied to address it. A more detailed description of the research objective of this thesis is given in Section 1.3. Before that, this chapter introduces the wider research context, where Section 1.1 presents an overview of the field of Deformable Object Manipulation and Section 1.2 continues with the Robot Learning landscape. This chapter concludes with the thesis outline, in Section 1.4.

1.1 Deformable Object Manipulation

DOM is a rapidly growing field of robotics. It encompasses robot manipulation problems in which the object being manipulated undergoes deformation. To date, the majority of robotics research has been limited to rigid object manipulation, thus requiring the assumption that objects would not change shape. Naturally, this has narrowed the scope of robotic applications, leaving many industrial tasks involving deformable objects still to be performed by humans. One of the most compelling evidence to this fact is an example from the automotive industry. Even though this industry started to become automated already in the early 1970s, until recently nearly 100% of the installations of car wiring systems was done manually [2].

Besides benefiting a variety of industrial applications, there are other sectors which stand to gain from advances in DOM, as highlighted in the previous section. However, DOM is characterized by several technical challenges, which according to Zhu et al. [3] can be summarized as:

- deformation is complex to model
- deformation is difficult to sense
- deformation gives rise to infinite degrees of freedom

Though these challenges are shared by all deformable objects, depending on the type of object they can have varying degrees of impact on the manipulation problem. For example, sensing deformation of a rope is particularly difficult because of its *low compression strength*, i.e. low resistance when two opposing points are pushed together. While grasping one end of the rope, very little information can be deduced about the rest of its shape through force-torque measurements alone. The same is not true for a metal wire with *high strain*, i.e. high resistance when it starts to deform. Since the material opposes deformation, there will be high forces which can be measured through force-torque sensors. This difference led Sanchez et al. [4] to categorize deformable objects according to these two physical properties[2], as shown in Figure 1.1.

Objects were also grouped by their approximate geometry [4]. The rope and wire mentioned above are examples of Deformable Linear Objects (DLOs)[3],

[2] In [4], low compression strength is referred to as *no compression strength*, and high strain is referred to as *large strain*.

[3] Sometimes referred to as Deformable One-dimensional Objects (DOOs).

which are characterized by having one dimension considerably larger than the other two. More types of *linear* objects include cables and threads. Objects can also be considered *planar*, when one dimension is considerably smaller than the other two, e.g. metal sheets, clothes and paper. Finally, *volumetric* objects have no dimension significantly larger/smaller than the others, much like the sponge in Figure 1.1.

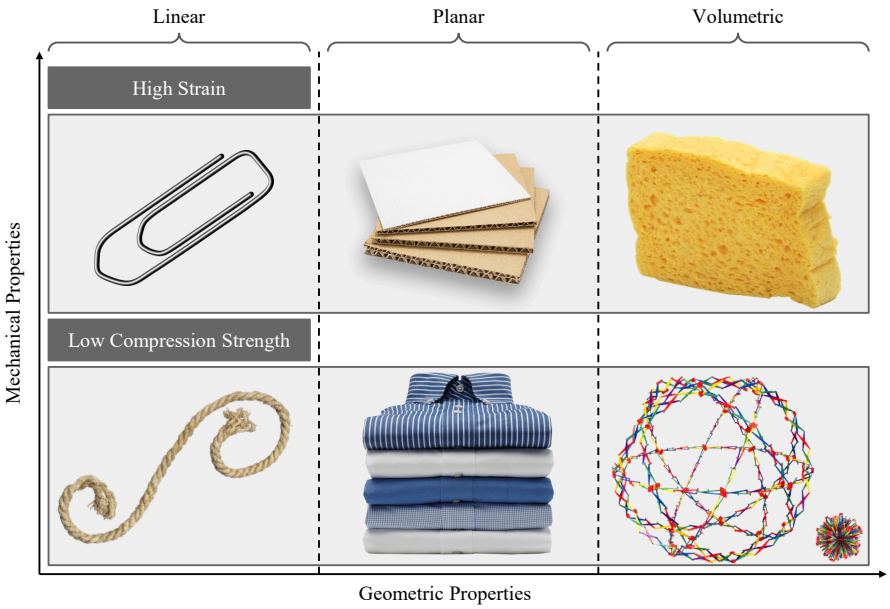


Figure 1.1: Classification of deformable objects proposed by Sanchez [4]. Based on mechanical properties, objects may have *high strain* or *low compression strength*. Based on an approximate geometry, these may be *linear*, *planar* or *volumetric*[4].

Attempting to find a manipulation strategy which generalizes to all classes presented in Figure 1.1, would require nothing short of human level intelligence and dexterity. Most research has therefore been focused on specific classes and even more often, on a particular task [4]. This thesis will focus on DLO manipulation, which will be covered in Chapter 2, through the lens of two representative tasks, introduced in Section 2.6.

[4] Though a Hoberman sphere is given as the volumetric example, it is technically not a deformable object but rather an articulated structure, with low compression strength.

1.2 Robot Learning

RoL can be defined as the application of ML to robotic tasks. However, this is a very broad definition since robotics is an interdisciplinary field, and ML can be applied to many different sub-problems. To name a few: **i.** for human-robot interaction through speech, one can use advances from Natural Language Processing (NLP); **ii.** to endow robots with visual comprehension, there are methods from Computer Vision (CV); **iii.** to find model parameters based on sampled data, there is model learning; **iv.** to be able to copy a human skill, Learning from Demonstrations (LfD) can be implemented; and **v.** in order to have robots learn directly by interacting with the environment, there is Reinforcement Learning (RL). Note that while examples **i-ii** are separate research areas that are relevant to robotics, **iii-v** are the three methods described by Peters et al. [5] as the core of RoL. Figure 1.2 illustrates the RoL landscape, while a short description of the methods is provided below.

Model Learning When ML is used for system identification, this is referred to as model learning. While classical approaches use statistical methods to learn specific classes of mathematical models, ML algorithms aim to learn more general mappings from inputs to outputs. There are two main types of models which can be learned: *forward* and *inverse*. The former aim at predicting the evolution of the system based on the current observation or a history of past observations. These predictions can then be used for control, which is what the latter type of model does directly. That is, inverse models attempt to predict the input required to achieve a desired output [5].

Learning from Demonstrations LfD includes two main strategies, namely Imitation Learning (IL) and Apprenticeship Learning (AL) [5]. In IL, which is also sometimes referred to as Behavior Cloning (BC), the robot estimates a policy from a teacher's demonstration in order to reproduce it. In AL, on the other hand, a reward function is used to assign scores to the demonstrations, with the intent to encode the true objective and go beyond pure imitation. This reward function needs to be chosen such that a perfect score is attributed to an optimal demonstration. Based on this reward function, it is then possible to find the optimal policy through RL. Consequently, this approach is also known as inverse reinforcement learning [5].

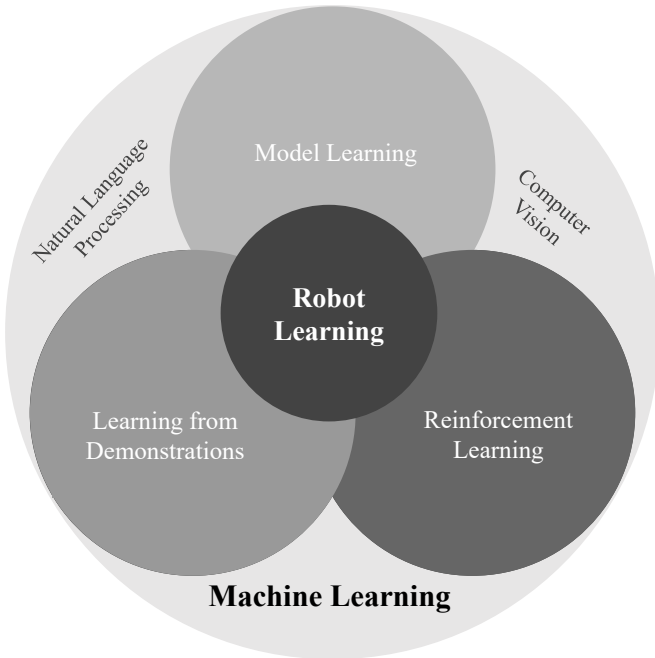


Figure 1.2: Overview of the RoL landscape, with three main research areas: learning from demonstrations, model learning and reinforcement learning. ML has also found many useful applications in robotics, from other research fields such as CV and NLP.

Reinforcement Learning Model learning and IL approaches mainly use algorithms from the Supervised Learning (SL) branch of ML, i.e. they learn a mapping function based on ground truth data. RL, on the other hand, is considered to be a separate branch which aims to learn by trial-and-error. Contrary to SL, RL is centered around an agent which interacts with the environment and makes decisions that influence what data it receives from future experience. Moreover, while in SL there are ground truth labels, in RL there is only a scalar reward signal which the agent attempts to maximize. This reward is also what sets it apart from unsupervised learning methods, where no signal of correctness is given. Although all RoL techniques can be applied to DOM problems, this thesis will focus on reinforcement learning in particular. Hence, Chapter 3 will introduce RL theory in greater detail.

After this overview of the three central RoL approaches, it is important to ask the question: why should one use any of them? According to Connell and Mahadevan [6], RoL is particularly important for problems in which:

- the environment is nonstationary
- it may be prohibitively hard to program a robot
- not all situations, as well as goals, may be foreseeable

Let us consider a DOM task we are all familiar with: folding laundry. Programming a robot to fold our clothes is extremely challenging. Since these are low compression strength materials, sensing has to rely mostly on vision data. This, coupled with the fact that the clothes we own change over time, make it a nonstationary environment. Furthermore, there is high variability in the geometrical and fabric properties of garments, which potentially requires a unique robot program for each individual item, i.e. the steps to fold a cotton t-shirt are not the same as to fold a pair of leather pants. Therefore, hard-coded approaches may quickly become impractical. Finally, the state of our laundry when taken straight out of the dryer is truly unforeseeable.

Unsurprisingly, the first complete pipeline for autonomous folding of clothes using a dual-armed robot proposed by Doumanoglou et al. [7] made use of several ML techniques to achieve a 79% success rate, for a small subset of garments. It should be noted that the grippers used in this work were specialized for handling clothes [8]. Besides RoL software, it is believed that another major bottleneck in DOM is the hardware [3]. This includes not only the robot, but also the grippers and sensors chosen for a particular task. Learning itself may become much simpler depending on the type of sensing and actuation used while interacting with the environment.

The work by Doumanoglou et al. [7] is evidence that ML plays an important role in DOM. Nevertheless, their approach applied mostly CV algorithms to solve sub-problems of the task, leaving the aforementioned RoL techniques still to be explored. Generally, robotic manipulation approaches fall in-between two extremes: either a single RoL method is applied in an *end-to-end* approach, or a *modular* framework is implemented using different algorithms to solve each individual sub-problem. This thesis aims to apply RL to learn robot-agnostic policies in task space, assuming a modular implementation with CV and haptic techniques for tracking the DLO and inverse kinematics algorithms for controlling the manipulator.

1.3 Research Objective

This Doctoral thesis presents the work carried out towards the **development of Artificial Intelligence (AI) for robotic manipulation of deformable objects**, as part of a research project within the Wallenberg AI, Autonomous Systems and Software Program (WASP). The project can be divided into two key components, one which refers to the *core technology* of AI, and the other to the DOM *application*. More specifically, this was to be achieved by combining:

- data-driven modeling of robot-object interaction, based on Deep Neural Networks (DNNs) and vision/force data.

- design of control policies based on reinforcement learning principles and testing them in real-world and simulated robotic setups.

This helped narrow the focus for the *core technology* of this project to RL, although other ML methods were also explored to a lesser extent. Indeed this means that a greater focus has been placed on the second item, with little work done on explicit model learning. As highlighted in Section 1.1, DOM is a large and heterogeneous problem which would be difficult to tackle all at once. Therefore, the DOM *application* was narrowed to Deformable Linear Object manipulation tasks. As will be made clear in Chapter 2, DLOs encapsulate a lot of the general challenges in DOM, making them good case studies on the effectiveness of RL for other types of deformable objects.

The research objective was tackled in stages. First, a simulation sandbox with several DLO manipulation tasks was developed in Paper A, which provided a flexible environment to test online RL algorithms. Then, this sandbox was used to tackle a DLO manipulation task with more challenging mechanical properties, leading to Paper B. In the final stage of this project, there was a shift towards offline RL algorithms to address a real-world task using an ABB Dual-Arm YuMi robot, resulting in Paper C. Instead of attempting to learn a DNN model of the robot-object interaction, Papers D and E both make use of more classical control techniques based on robot kinematics, which do not require a dynamical model. In both works, state estimation of the DLO was implemented, using vision in Paper D and force in Paper E. Furthermore, both papers employ ML algorithms to solve different sub-problems, however applying RoL methods is left as future work.

1.4 Thesis Outline

The goal of Part I is to provide an overview of DOM and RoL, needed to understand the research context of the papers compiled in Part II. Therefore, the theoretical concepts introduced here are meant to be a complement to the papers, not a repetition. The rest of Part I is organized into two main chapters: Chapter 2 introduces relevant related work on DLO manipulation and defines the two representative tasks that are addressed in this thesis, while Chapter 3 presents a brief but comprehensive coverage of RL theory. Finally, Chapter 4 provides a summary of the research contributions of the appended papers, followed by some concluding remarks, provided in Chapter 5.

CHAPTER 2

Deformable Linear Object Manipulation

DLOs were one of the first classes of deformable objects to be studied within robotic manipulation. This is due in part to their comparatively simple geometry, with respect to planar and volumetric objects [4]. More importantly, DLOs are crucial components in numerous industrial, healthcare and service applications. Consider the countless electrical cables which are installed in electronic devices, and the variety of hoses and tubes found across so many industrial machines. Previous work on robotic DLO manipulation has tackled specific tasks, such as USB cable insertion [9], hot-wire cutting [10], knot tying [11] as well as untangling [12], wire harness assembly [13], surgical suturing [14], etc. Other works have focused on important subproblems such as state estimation [15], modeling [16] and shape/deformation control [17].

In this chapter, five key subproblems of robotic manipulation are introduced: **i.** modeling, **ii.** simulation, **iii.** sensing, **iv.** planning and **v.** control. Section 2.1 gives an overview of modeling techniques used for DLO manipulation. Section 2.2 covers simulation with an emphasis on software. In Section 2.3, three major sensing modalities are summarized. Planning and control are addressed in Sections 2.4 and 2.5, respectively. Finally, the two representative tasks addressed in this thesis are presented in Section 2.6.

2.1 Modeling

In DOM problems, modeling can be applied to describe either the geometry or the dynamics of a deformable object. The former is useful for defining a shape representation, while keeping dimensionality low. Dynamical models on the other hand, attempt to describe the behavior of the object when external forces are applied and are mostly used for prediction and control. Note that both geometric and physics-based models may be analytical or learned via ML methods. This section begins by presenting the problem of shape representation in Section 2.1, followed with an overview of modeling techniques for deformation physics in Section 2.1. For a more extensive tutorial on deformable object modeling, refer to the review by Arriola-Rios et al. [18].

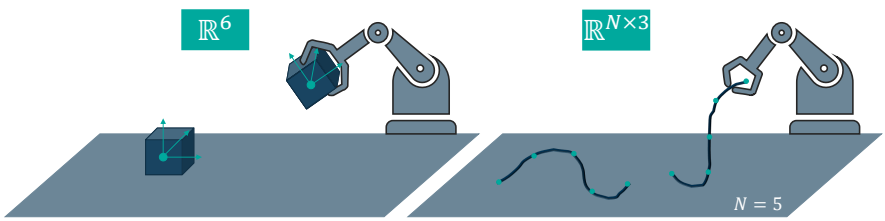


Figure 2.1: Comparison of state representations for a rigid object and a DLO. While the rigid cube can be represented by its pose, the DLO is approximated as a point cloud with N points.

Geometric Modeling

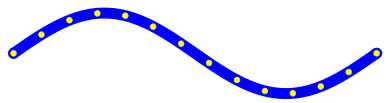
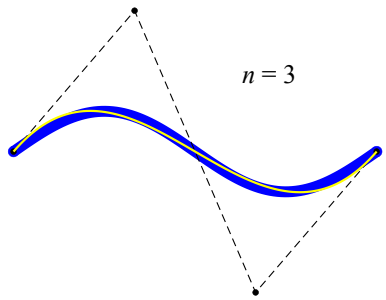
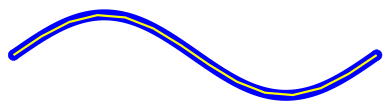
In robotic manipulation, it is often sufficient to define the state of a rigid object by its pose vector in $\mathbb{R}^6$, i.e. position and orientation, as illustrated in Figure 2.1. If the geometry, mass and surface characteristics of the object are also known, that provides a complete description. Conversely, there is no obvious choice of representation for a DLO since the state must also include information about its shape [19]. One straightforward option is to discretize the continuous object as a set of points in $\mathbb{R}^{N\times3}$. Despite simple, this has a couple of drawbacks: firstly, it is an approximation which will be increasingly accurate as $N \to \infty$, also leading to a larger state space; secondly, it cannot describe torsion unless the orientation of each point is also included in the state, which would further increase the state space to $\mathbb{R}^{N\times6}$.

Geometric representations are intimately related to state estimation and tracking. For example, Tang et al. [11] developed a framework for DLO manipulation using Coherent Point Drift (CPD) for tracking the object. Since CPD is a point set registration method i.e. it aligns a point cloud with another, they discretized the DLO as a set of uniformly spaced points, i.e. a **point cloud**. Paper D uses the Structure Preserved Registration (SPR) [13] tracking method which improves on the CPD principle through a local regularization term, proposed by the same authors. Zea et al. [20] on the other hand, chose **Bézier curves** to represent the DLO for a Bayesian state estimation method. Yet another approach was used by Wnuk et al. [21] who represented the DLO's state by its **skeleton line**, from which a kinematic multi-body model was derived for control. These three geometric models are shown in Table 2.1, though there are many more summarized by Yin et al. [22].

All aforementioned methods rely on vision data and therefore require several preprocessing steps of RGB and/or depth measurements. Segmentation, i.e. separating objects from the background, is particularly important for the success of DLO tracking. CPD and SPR only require this image processing phase to obtain the point cloud representation. However, the other two methods employ continuous shape representations, further requiring a curve fitting algorithm. Zea et al. [20] constructed a rectangle chain approximation of the Bézier curves, to simplify the fitting process. Much like many other algorithms, their work also relied on a dynamical model to better predict the object state. Furthermore, Wnuk et al. [21] modeled the skeleton curve as a weighted sum of Radial Basis Functions (RBFs), for which they computed weights minimizing the error between the point cloud and the RBF.

Besides analytical modeling approaches, there have also been attempts to learn shape representations through black box models, e.g. Artificial Neural Networks (ANNs). Yan et al. [15] proposed a self-supervised learning approach for estimating the DLO as an ordered sequence of points. Alternatively, Sundaresan et al. [23] learned Dense Depth Object Descriptors (DDODs). Once a shape representation is obtained, it can be used either as **i.** a feedback signal or as **ii.** state variables. In the first case, the representation should be robust to noise. In the second case, the goal is to find a good trade-off between lower dimensionality and higher accuracy [3]. Finally, in end-to-end approaches a latent shape representation is implicitly learned directly from sensory data. Sensing modalities are discussed in Section 2.3.

Table 2.1: Examples of DLO state representations used in related work. This table presents the formal definition on the left and an illustrative plot on the right, where the DLO is drawn in blue and the representations in yellow. The cubic Bézier curve ($n = 3$), includes the set of $n + 1$ control points $\mathbf{p}_i$ and the respective Bézier polygon as the dashed line, in black.

A **point cloud** [11] represents the DLO as a discrete set of N points: $$X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \in \mathbb{R}^{N\times 3}$$ where $\mathbf{x}_n \in \mathbb{R}^3$ is the n-th point's position in Cartesian space.	
A **Bézier curve** [20] of order n is defined as a weighted sum of Bernstein polynomials, $B_i^n(u)$, with a set of $n + 1$ control points, $\mathbf{p}_i$: $$\mathbf{B}(u) = \sum_{i=0}^{n} \mathbf{p}_i B_i^n(u)$$ where, $$B_i^n(u) = \binom{n}{i}(1-u)^{n-i}u^i$$ with $u \in [0, 1]$.	
A **skeleton line** [21] is modeled as a continuous spatial curve in Euclidean space, $$f(s) : \mathbb{R} \mapsto \mathbb{R}^3$$ where $s \in [0], [1]$ are local coordinates running along the DLO's length, L. The curve in turn is described by the Frenet-Ferret frame: $$\mathbf{R}(s) = [f'(s) \quad f''(s) \quad f'(s) \times f''(s)]$$	

As-Rigid-As-Possible Deformation A particularly important method in the context of this thesis is the As-Rigid-As-Possible (ARAP) [24], [25] deformation model. While in Paper B the point cloud of the DLO can be obtained directly from the ReForm simulation, for the real-world experiments of Paper C, there needed to be a tracking method from which to extract the point cloud of the object. To that end, the ARAP-based algorithm developed by Shetab-Bushehri et al. [26] was used.

The ARAP model, originated in the field of computer graphics, to provide a way to deform geometric shapes while still preserving local rigidity. Although there are several formulations of the ARAP deformation problem, Sorkine and Alexa [25] present a simple energy formulation for surface modeling of meshes, which is described here in more detail.

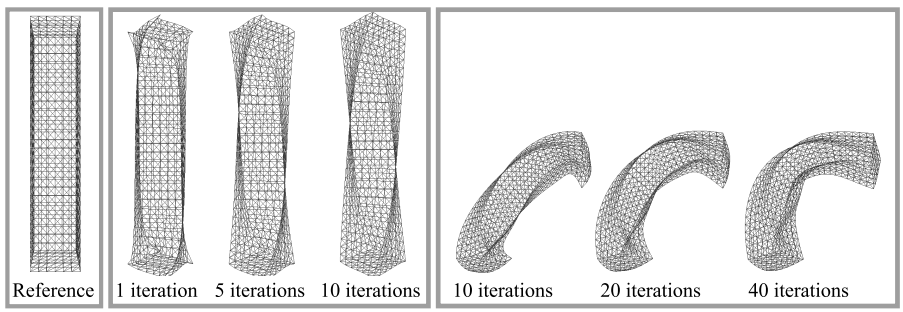


Figure 2.2: Illustration of ARAP model with two consecutive deformations. The selected *handle* points are the 8 extreme vertices of the bar. For the twist deformation both top and bottom vertices are rotated by 45° in opposing directions. For the bend deformation, only the top vertices were controlled to create a 90° angle.

Consider a triangle mesh $\mathcal{S}$, with n vertices and m triangles, where $\mathcal{N}(i)$ denotes the one-ring neighbors of a vertex i, i.e. the set of vertices connected to i. The geometric embedding of $\mathcal{S}$ is defined by the vertex positions $\mathbf{p}_i \in \mathbb{R}^3$. The reference mesh $\mathcal{S}$ is deformed into $\mathcal{S}'$ with the same connectivity but different geometric embedding, $\mathbf{p}'_i$. In order to describe the energy function for local rigidity, they define a cell $\mathcal{C}_i$ as the set of triangles incident upon a vertex i, and its deformed version $\mathcal{C}'_i$. If the transformation $\mathcal{C}_i \rightarrow \mathcal{C}'_i$ is rigid, there exists a rotation matrix $\mathbf{R}_i$, such that:

$$\mathbf{p}'_i - \mathbf{p}'_j = \mathbf{R}_i \left(\mathbf{p}_i - \mathbf{p}_j\right), \quad \forall j \in \mathcal{N}(i) \tag{2.1}$$

For non-rigid transformations we can find the least-squares approximation of $\mathbf{R}_i$, by minimizing the energy:

$$E(\mathcal{C}_i, \mathcal{C}_i') = \sum_{j \in \mathcal{N}(i)} w_{ij} || \left(\mathbf{p}_i' - \mathbf{p}_j'\right) - \mathbf{R}_i \left(\mathbf{p}_i - \mathbf{p}_j\right) ||^2 \tag{2.2}$$

where w_{ij} is a per-edge weight which aims to make the energy function as mesh-independent as possible. Choosing $w_{ij} = 0.5 \left(\cot \alpha_{ij} + \cot \beta_{ij}\right)$, where α_{ij} and β_{ij} are the two angles opposite of the edge (i, j), compensates for non-uniformly shaped cells [25].

Finally, the measure of deformation rigidity of the whole mesh can be defined as the sum over all the rigidity deviations per cell, resulting in:

$$E(\mathcal{S}') = \sum_{i=1}^{n} E(\mathcal{C}_i, \mathcal{C}_i') \tag{2.3}$$

which only depends on the vertex positions $\mathbf{p}'$, since the reference mesh geometry, determined by $\mathbf{p}$, is constant and $\mathbf{R}$ is a well defined function of $\mathbf{p}'$.

In the proposed modeling framework, they then solve for the positions $\mathbf{p}'$ of $\mathcal{S}'$ that minimize $E(\mathcal{S}')$, under some user-defined constraints. To that end, the gradient of $E(\mathcal{S}')$ is computed with respect to positions $\mathbf{p}'$ and the result set to zero, leading to the following linear system of equations:

$$\sum_{j \in \mathcal{N}(i)} w_{ij} \left(\mathbf{p}_i' - \mathbf{p}_j'\right) = \sum_{j \in \mathcal{N}(i)} \frac{w_{ij}}{2} \left(\mathbf{R}_i - \mathbf{R}_j\right) \left(\mathbf{p}_i - \mathbf{p}_j\right) \tag{2.4}$$

Constraints are incorporated into the above system of equations, by setting fixed positions for the desired vertices, i.e. $\mathbf{p}_j' = \mathbf{c}_k$ for $k \in \mathcal{F}$, where $\mathcal{F}$ is the set of indices of the constrained points. These may be either *static* or *handle* vertices, depending on if they are controlled by the user or not.

The authors propose an iterative minimization strategy, that starts with an initial guess for positions $\mathbf{p}_i'(0)$, which can be used to estimate the local rotations $\mathbf{R}_i(0)$, using equation (2.2). These estimates are then used to compute new positions $\mathbf{p}_i'(1)$, by solving equation (2.4). These two steps are interleaved until the local energy minimum is reached. Figure 2.2 shows an illustration of the ARAP method in action over several iterations, for two consecutive deformations, first by twisting and then by bending a bar.

Physical Modeling

Deformation occurs when an object changes shape due to the application of external forces. In addition, there are internal forces that neighboring particles of a continuous material exert on each other, which are expressed as the stress, σ. When a force F is applied, this results in a dimensional change along the force direction, namely the strain ε. There is a wide range of deformation behavior depending on the material properties of an object. Stress tests are typically performed to analyze such properties, by applying tensile, compressive, bending, torsion or shear forces. Stress and strain are computed differently depending on the type of force and geometry of the object. For a tensile force, the stress and strain of a rod are given by:

$$\sigma = \frac{F}{A_0} \quad \text{and} \quad \varepsilon = \frac{L - L_0}{L_0}$$

where A_0 and L_0 are the original cross-sectional area and length, respectively and L is the measured length. Note that as the rod extends, its cross-sectional area will shrink since the volume of the object remains constant. The relationship between such axial and lateral strains is given by **Poisson's ratio**, ν [4]. Figure 2.3 shows an illustration of tensile stress tests applied to three rods with different material properties and the corresponding stress-strain curves.

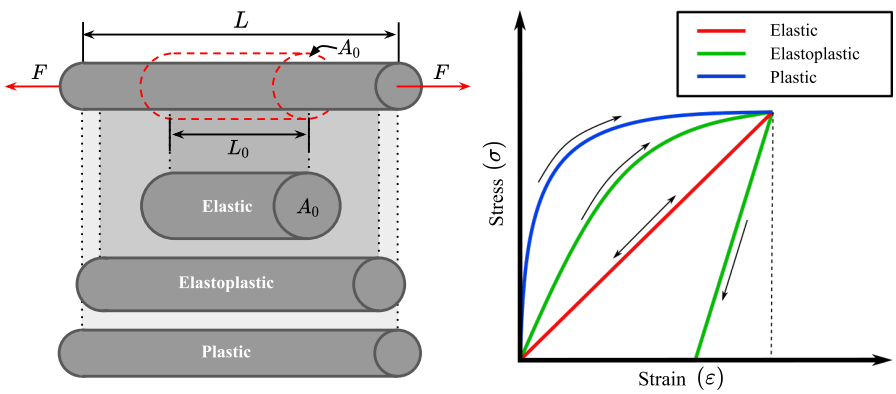


Figure 2.3: Illustration of different types of deformation. The stress-strain curves on the right show the behavior up to a certain strain, followed by the removal of the force. The resulting deformations are shown on the left.

When an object undergoes *elastic* deformation it returns to its original shape once the force is removed. An isotropic material, i.e. uniform in all orientations, with **Young's modulus** E, is said to be linearly elastic if the strain is proportional to the stress, $\sigma = E\varepsilon$. In contrast, when an object undergoes *plastic* deformation it will become permanently deformed. An intermediate behavior is exhibited by *elastoplastic* materials which become permanently deformed after a certain threshold but are able to partially recover once the force is removed [4], as shown in Figure 2.3. These may act elastic within a certain range of stresses and plastic when a **yield point** is crossed. Once this happens, the resulting shape becomes strongly history-dependent. The effect of these properties are studied in Paper B. Finally, while for the aforementioned types of deformation the stress rate is of no importance, the same is not true for viscous materials. When the material is *viscoelastic*, the strain is a function of the stress rate. If this time dependent behavior is accompanied by permanent deformation, then the material is *viscoplastic* [27].

Due to this large variability, it becomes necessary to choose a modeling technique which can adequately describe an object's behavior. While the properties defined above relate to mechanics of materials, physical models appear in other contexts as well. Given their simplicity and computational efficiency, *discrete* models have been used extensively, of which mass-spring systems are the most common formulation. Lv et al. [28] modeled a DLO as a series of masses connected by linear, bending and torsion springs. However, discrete models are not as physically accurate as *continuum* ones, which are more complex and therefore also more computationally heavy. As a consequence, these tend to be used outside of robotics for detailed material analysis e.g. finite element modeling of Warrington-Seale rope [29]. Finite Element Methods (FEM) are numerical techniques commonly used to solve partial differential equations of continuum-based models, by splitting the object into discrete elements that approximate its geometry [4].

Other works have used simpler energy-based models [16], [30]–[32], for simulating and controlling DLOs. Despite being physically inspired and computationally efficient, they do not explicitly model material properties. Recently, high-fidelity physics engines which support real-time execution have become readily available. That has enabled the use of more advanced models of DLOs which can be easily utilized out of the box. An overview of simulation using physics engines is presented in Section 2.2.

Catenary In the context of this thesis, specifically in Paper E, the *catenary* curve is an important model. Since this work consisted of a blind implementation for cable-routing, the state of the DLO needed to be estimated through endpoint FT measurements, assuming a catenary model.

Catenaries are used to describe flexible cables, within the statics branch of classical mechanics. A uniform and inextensible DLO, suspended from its endpoints A and B while hanging under its own weight assumes the shape with least potential energy, namely a catenary curve. Figure 2.4 shows an illustration of such a curve, along with the free-body diagram of a DLO segment.

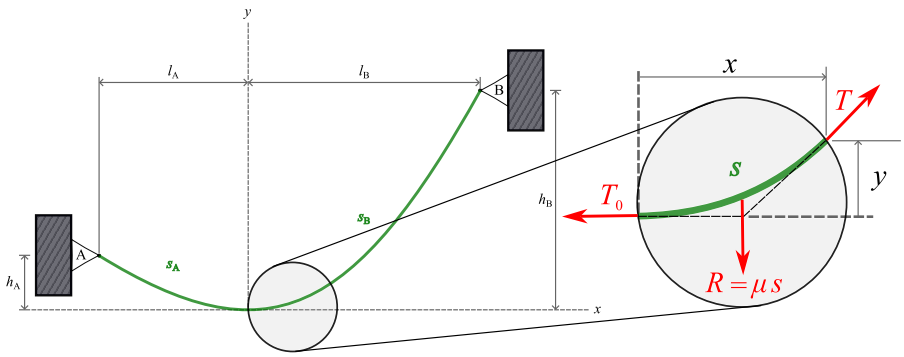


Figure 2.4: DLO modeled as a catenary curve. The free-body diagram on the right shows relevant forces such as the minimum tension T_0, found at the lowest point of the DLO, a tension along the curve T and the resultant force R, which depends on the weight per unit length μ. Catenary expressions need to be evaluated for each side, A and B.

Assuming that the DLO has a weight μ per unit length, the resultant force of the load is given by $R = \mu s$, with incremental vertical load $\mu\, ds$, leading to the following differential equation:

$$\frac{d^2y}{dx^2} = \frac{\mu}{T_0}\frac{ds}{dx} = \frac{\mu}{T_0}\sqrt{1 + \left(\frac{dy}{dx}\right)^2} \tag{2.5}$$

where the arc length s is a function of the coordinates, $s = f(x, y)$. Equation (2.5) can be solved by integration to obtain a hyperbolic function representing the shape of the curve in Cartesian space, expressed as:

$$y = \frac{T_0}{\mu}\left(\cosh\frac{\mu x}{T_0} - 1\right) \tag{2.6}$$

From the free-body diagram, it can be observed that the change in slope is given by $dy/dx = \mu s/T_0$, leading to the following expression for the arc length:

$$s = \frac{T_0}{\mu}\sinh\frac{\mu x}{T_0} \tag{2.7}$$

Furthermore, from the equilibrium triangle of the free-body diagram, the following relationship for the tension T can be determined:

$$T^2 = \mu^2 s^2 + T_0^2 \tag{2.8}$$

Finally, the expression for the tension along the curve is obtained by combining equation (2.7) and (2.8): $T = T_0 \cosh(\mu x/T_0) = T_0 + \mu y$. These relations make it possible to determine the shape of a DLO with known length and weight, through its endpoint position and tension, as shown in Figure 2.5.

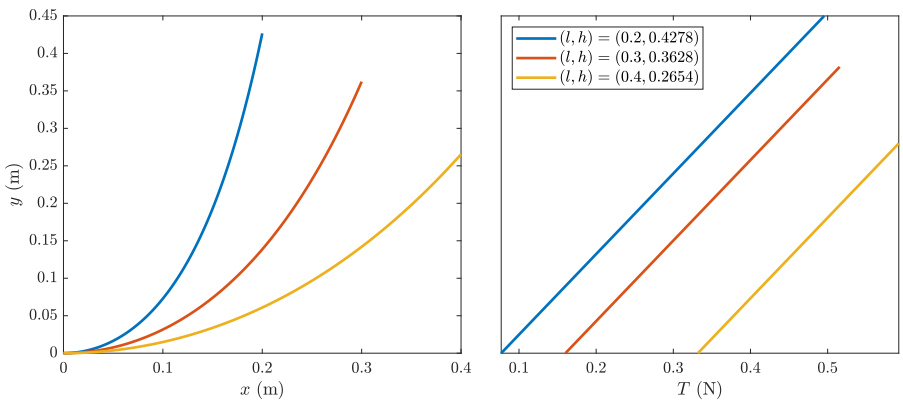


Figure 2.5: Example of catenary with weight per unit length $\mu = 0.1$ N/m and arc length $s = 0.5$ m from its lowest point until its endpoint. The left plot shows the shape of the DLO computed through equation (2.6), based on three different endpoint positions (l, h). The right plot shows the corresponding tension T along each curve, starting with T_0.

Finally, if the sag-to-span ratio of a catenary is small, i.e. $h/l \ll 1$, which means that the DLO is taut, approximations can be made [33, Chapter 5].

2.2 Simulation

The models described in the previous section can be used to simulate the object. However, developing specialized simulators can be time-consuming and the result will be quite limited, since modeling the object is only the first step. Indeed the object's interaction with the robot and the environment also play an important role, and lead to complex dynamic behaviors. Therefore, in this thesis, a greater focus was placed into finding an appropriate physics simulation software which could be harnessed for our purpose.

In the field of robotics, there are several simulators which are commonly used such as Gazebo [34], CoppeliaSim (f.k.a. V-REP) [35], PyBullet [36], and MuJoCo [37]. It is important to distinguish between a simulation software and the underlying physics engine. For example, both Gazebo and CoppeliaSim support Open Dynamics Engine (ODE) or Bullet as the physics engine, among others. On the other hand, MuJoCo and (Py)Bullet are standalone physics engines. Since they all offer rigid body kinematics and dynamics it is possible to approximate a DLO as a series of rigid links connected by ball-joints, which can be viewed as an underactuated robot. Unfortunately, simulating more complex deformation is still limited [3].

Bullet supports soft body dynamics for cloth, rope and deformable volumes, using `btSoftBody` objects. To create a DLO, Bullet offers the `CreateRope` function in `btSoftBodyHelpers`, but more complex DLOs require a specialized implementation [38]. For both of these engines, the mechanical properties of a DLO need to be set through constraint parameters connecting rigid bodies. Even though such parameters can be tuned to approximate the behavior of a real DLO, they are not based on deformation properties and are better suited to represent low compression strength materials.

MuJoCo 2.0 supported simulation of `composite` objects like particle systems, ropes, cloth and other soft bodies [37]. With the recent MuJoCo 3.0[1] release, a new low-level model element called `flex` was added which can model stretchable lines, triangles and tetrahedra. A DLO that was modeled as a `rope` object in version 2.0, should be modified to and `cable` object in version 3.0, since the former was removed [39].

[1] MuJoCo was purchased by Google DeepMind in October 2021, and made Open Source. In October 2023, the 3.0 release shows the company's efforts to provide better support for deformable objects, with a new `flex` element. However, this is still in an early stage, as they write: "This feature is still under development and subject to change".

Alternative simulators allow to set actual material properties, with parameters such as Young Modulus and Poisson's Ratio. This is true for the Simulation Open Framework Architecture (SOFA) [40] which was initially developed for medical applications and supports both mass-spring and FEM deformable models. SOFA has also been used for soft robotics, e.g. to control elastic soft robots [41]. AGX Dynamics, the simulator used in Paper A and B, offers similar possibilities. There are two main DLO classes provided with this software: `Cable` and `Wire`. The former is aimed for fixed-length DLOs, which may exhibit elastoplastic behavior, while the latter is better fitted for DLOs where torsion is not relevant and length may vary [42]. In ReForm only `Cable` objects were used, and Figure 2.6 shows how they are modeled.

Although the aforementioned simulators are the most prominent in robotics research, there are many others. For example, Blender is a 3D animation software that also supports Bullet for more advanced physics simulation. Sundaresan et al. [23] used Blender to generate synthetic depth data for rope manipulation through DDODs. Physics engines, such as NVIDIA PhysX and FleX, have also been applied to robotic contexts with simulated environments like SoftGym [43], SAPIEN [44] and ThreeDWorld [45]. More recently, NVIDIA has released Isaac Gym [46], which uses these two physics engines as backend, and provides more utilities that are useful for RL research. Section 3.6 covers simulation environments for RoL research in further detail.

Finally, there are also many other smaller scale simulators for specific applications, such as the Elastica [47] library based on Cosserat rod theory, which has been used with RL but is not intended for robotics.

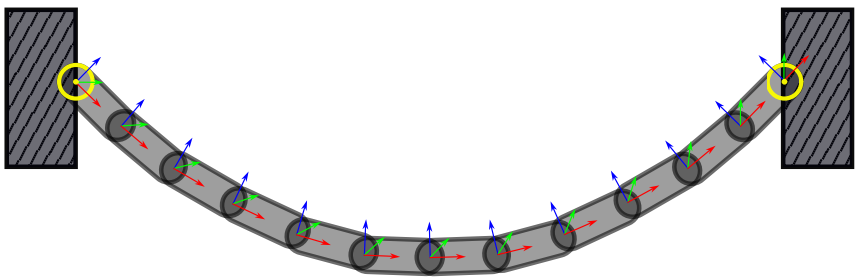


Figure 2.6: Illustration of a `Cable` object in AGX Dynamics, modeled as a set of rigid capsules connected by constraints. Consecutive capsules are exempted from contact generation. Global material properties set on the `Cable`, are translated into local constraint properties.

2.3 Sensing

Sensing is considered as one of the most important areas of research when it comes to DOM [3]. There are several sensing modalities which may be used for robotic manipulation, e.g. vision, tactile, olfactory, thermal and force. According to Yin et al. [22], vision seems to be the predominant sensing modality in DOM applications. However, the choice of modality depends on the task, as well as the state representation which needs to be estimated. For example, with low compression strength DLOs which incur large deformations, vision is preferable to force-torque sensing since the latter only provides local information about the object [3]. Conversely, for contact rich tasks where there may be occlusion by the gripper, perhaps tactile sensing is a better choice [48]. There are three main sensing modalities which have been used for DLO manipulation: vision, force-torque, and tactile. These will be presented below together with relevant related work.

Vision Naturally, when it comes to DOM applications, vision is an almost indispensable sensing modality. However, since RGB data is high dimensional and contains information about the whole scene, it is often fed into tracking algorithms to obtain much smaller state representations of the object of interest. Deformation increases the difficulty of object tracking considerably, but some algorithms have shown promising results for DLO manipulation [11], [13]. Alternatively, RGB and Depth maps (RGB-D) have been used directly in end-to-end implementations [49]. A unique challenge for DLO manipulation in particular is the need to perceive topological information, such as knots and loops which lead to self-occlusion [12], [50], [51]. Papers C and D are both real-world implementations relying on RGB-D sensing to track the DLO.

Force-torque One of the most commonly used sensing modalities in rigid object manipulation problems is force-torque (FT) sensing. Tough it does not provide much usable information for low compression strength objects, it may be useful for high strain DLOs [52]. Force-torque measurements can be obtained directly using sensors, or estimated based on proprioceptive signals, i.e. the joint torques of the robot. Sanchez et al. [53] used FT sensing to estimate the location and magnitude of the contact forces applied to a deformable bar [54]. In Paper E, only FT measurements are used to track the state of the DLO, without any visual information.

Tactile The third and least mature modality is tactile sensing [55]. Even though there are many types of tactile inputs, in general they encode local information on the contact surface, such as forces, pressure or texture. There is a large variability in the sensing technology used, and consequently also in the capabilities of the sensors. The most common approaches are based on optical e.g. GelSight sensors [48], capacitive e.g. tactile sensors installed on the PR2 [56], or resistive technologies, e.g. the specialized sensors developed by Drimus et al. [57]. Sensors using GelSight technology have been successfully applied to DLO manipulation by She et al. [9], and shown to improve dexterity. There are also some examples of capacitive [58] and resistive [59] technologies being used for cable manipulation tasks. Alternatively, the BioTac sensor [60] is a biologically inspired sensing modality which makes use of vibrations to extract tactile information and was used by Sanchez et al. [53] in a similar setting to the aforementioned FT implementation, by the same author.

Finally, multiple sensing modalities may be combined for state estimation or identification of model parameters, through sensor fusion techniques [61].

2.4 Planning

Planning is a very broad term, extending well beyond the field of robotics. It is also closely related to control, which is shortly addressed below. Although it is difficult to draw a line separating the two concepts, planning can be seen as a higher level process than control [62]. In DOM problems, it is often the case that both intelligent high-level planning and low-level control are needed. Generally speaking, a complete strategy for robotic automation has to include both *manipulation* and *grasping* operations. Determining the sequence of such discrete tasks that leads to successfully reaching an objective, is considered a **task planning** problem. Moreover, the continuous motion of each task to be executed by the robot needs to be determined, i.e. how to move the arm and the gripper. This problem is refers to **motion planning**, which LaValle [62, Part II] defines as "determining what motions are appropriate for the robot so that it reaches a goal state without colliding into obstacles". While this definition is generally complete for reaching motions, when thinking of DOM tasks many more constraints need to be considered, such as not overstretching a DLO. Both types of planning problems need to be solved for the cable-routing task, which will be introduced in Section 2.6.

2.5 Control

Similarly to planning, control theory also extends to many fields. However, in the context of robotics it refers to designing feedback policies that determine the execution of lower level robot actuation. Robotic manipulators generally consist of a mechanical structure with a set of rigid links connected by a set of joints, e.g. revolute, prismatic, etc. There are many configurations used for manipulators, e.g. Cartesian, anthropomorphic, etc. Typically a manipulator is also fitted with an end-effector specific for the task, e.g. gripper, suction plate, etc. [63]. Therefore, a control strategy is highly dependent on the combination of manipulator and end-effector. Also, depending on the desired task, one may define the control inputs in **task space**, i.e. in terms of end-effector trajectories, or in **joint space**, i.e. in terms of joint trajectories. In the end, control policies defined in task space still need to be translated into joint space, via Inverse Kinematics (IK) algorithms. Papers C-E make use of the Hierarchical Quadratic Programming (HQP) [64] algorithm to solve such a problem, subject to a hierarchy of constraints.

There are some DOM tasks that can be solved as a pure control problem (without planning), when *grasping* is not considered. Indeed, all DLO tasks in Paper A are formulated in terms of low-level control policies. In both Paper A and Paper B, the dynamics of a robot manipulator are not modeled in the simulation, since points on the DLOs can either be directly controlled in Cartesian space, or pushed by a separate rigid object. Furthermore, the shape-servoing task introduced in Section 2.6, is formulated as a control problem.

RL for Planning and Control This thesis places a big emphasis on Reinforcement Learning, as explained in Section 1.3. RL principles are generalizable across a wide range of domains, being able to capture low-level controllers, higher level planners and everything in between. There is a great deal of freedom when it comes to formulating an RL problem, as will be made clear in Chapter 3. Broadly speaking, RL assumes there is an **environment** with which an **agent** interacts by observing a **state** and performing an **action**. Crucially, the agent also receives a **reward** indicating a measure of how good a certain action is when in a certain state. By trial-and-error learning, a policy can be gradually improved to maximize the expected future reward. While this versatility is one of the greatest appeals of RL theory, it simultaneously can be seen as one of its biggest drawbacks.

2.6 Representative Tasks

Each DLO manipulation task comes with unique challenges. Since it is not feasible to cover every task in one thesis, two representative tasks have been selected to be addressed in more detail, namely shape-servoing and cable-routing. These tasks are prevalent in DLO manipulation literature, and share many of the common challenges of DOM, highlighted in Section 1.1. In particular, both can be classified as a form of **shape control** problems, as opposed to DOM tasks where controlling the deformation of the object is not the primary objective. For example, in robotic food cutting [65], a knife may cause an undesired compression of the object while the goal is to be able to slice through it. In such tasks, deformation acts more like a disturbance.

Paper A classifies shape control problems as either **implicit** or **explicit**. The former class describes tasks where the goal is to deform a DLO into a specific shape, while the latter class describes tasks where the shape of the DLO must be controlled so that a more general condition is satisfied, e.g. wrapping a DLO around a cylinder. This distinction is particularly relevant in the context of RL, since adequately defining the goal through a reward function is key to its success. The simulation sandbox proposed in Paper A includes three tasks of each class, shown in Figure 2.7.

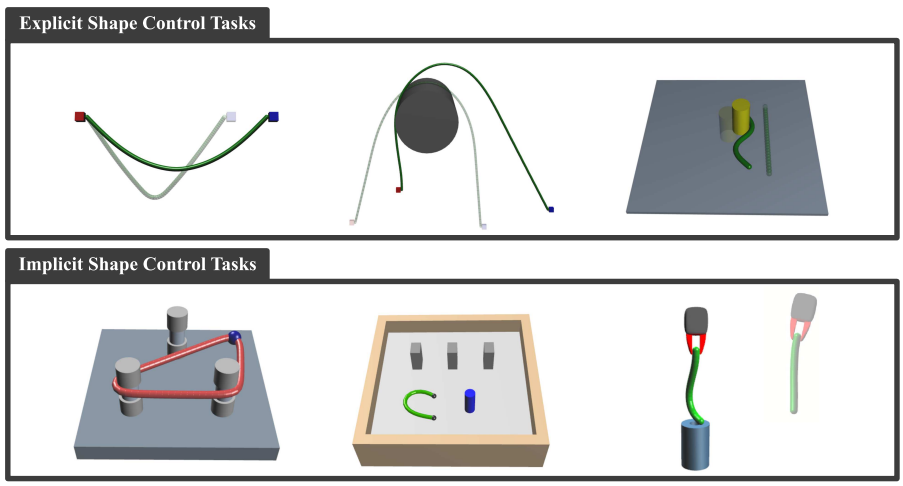


Figure 2.7: Shape control tasks available in ReForm simulation sandbox.

Shape-Servoing

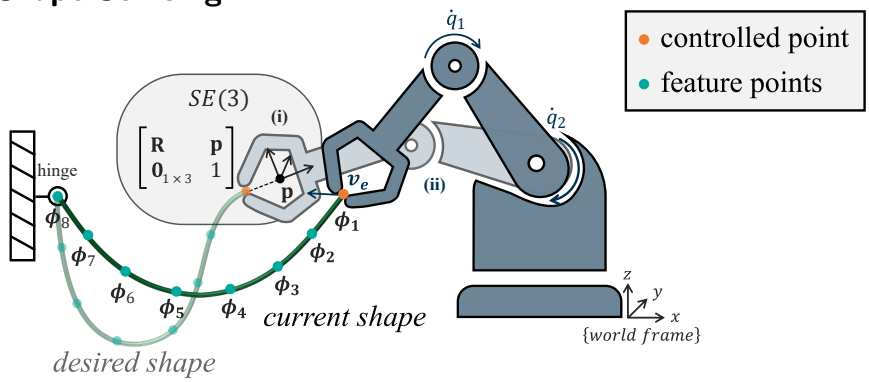


Figure 2.8: Illustration of a DLO shape-servoing task. As an explicit shape control problem, the goal is to deform the DLO into a *desired shape*. The image shows a task space control approach, with **(i)** a Cartesian end-effector, moving independently in $SE(3)$; and **(ii)** an anthropomorphic arm with gripper $\boldsymbol{v}_e$ and joint $\dot{q}_1, \dot{q}_2$ velocities. Both are rigidly attached to a controlled point $\boldsymbol{\phi}_1$ of the DLO with feature points $\boldsymbol{\phi}_i$, $\forall i \in [1, 8]$.

Shape-servoing describes tasks in which the goal is to drive a deformable object into a specified shape. The term servoing usually refers to a real-time closed-loop control policy for the robot motion, based on a sensor input, e.g. visual. Typically, servoing problems are formulated in terms of the desired velocity of the manipulator, driving an instantaneous error to zero. However, the same DLO shape control task can be formulated as a sequence of pick and place motions [15], [66], making it more of an online planning problem.

Given the ubiquity of IK control in robotics, several DLO shape-servoing strategies have followed a similar approach, by modeling the object with a deformation Jacobian $\mathbf{J}_{\text{DLO}}(\boldsymbol{\phi})$, where $\boldsymbol{\phi}$ is some representation of the DLO's shape [67]–[69]. One of the key obstacles for such ***model-based*** approaches is the modeling complexity of deformable objects, as highlighted in Section 2.1. Since this Jacobian matrix characterizes how the feature points $\boldsymbol{\phi}_i$ move given a certain end-effector velocity $\boldsymbol{v}_e$, it depends on the physical properties of the DLO. Therefore, this type of approach can require some calibration procedure to obtain modeling parameters, i.e. system identification [70]. As a consequence, such methods have mostly been applied to objects with simpler dynamics, e.g. fairly stiff objects, and are likely to fail in cases with more complex dynamics, such as elastoplastic DLOs. Alternatively, the Jacobian can be

learned directly by function approximation, however this requires sufficiently varied training data, otherwise overfitting may lead to poor generalization. Yu et al. [71] proposed a DL method to learn $\mathbf{J}_{\text{DLO}}(\boldsymbol{\phi})$, and applied an adaptive control strategy for shape-servoing a DLO. For reference, Almaghout et al. [72] provide a comprehensive review on related methods.

Berenson [17] proposed a shape-servoing method which approximates the deformation Jacobian, using a *diminishing rigidity* heuristic, i.e. the farther the end-effector is from a feature point, the less displacement it induces on that point. This approach does not require a physical model, but in principle it still relies on a geometric model. A simplified instance[2] is presented here in terms of a single controlled point $\boldsymbol{\phi}_1 \in \mathbb{R}^3$ rigidly attached to the end-effector, as depicted in Figure 2.8, which is assumed to be able to move independently in $SE(3)$. The DLO velocities are given by $\dot{\boldsymbol{\phi}} = \tilde{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}}$, where $\mathbf{q} \in \mathbb{Q}$ is the end-effector configuration and $\tilde{\mathbf{J}} \in \mathbb{R}^{3P\times 6}$ is the approximated Jacobian:

$$\tilde{\mathbf{J}}_i(\mathbf{q}) = \overbrace{e^{-k\cdot d_i}}^{rigidity}\Big[\ \underbrace{\mathbf{I}_{3\times 3}}_{translation}\ ,\ \underbrace{\mathbf{R}_1\times\mathbf{r},\,\mathbf{R}_2\times\mathbf{r},\,\mathbf{R}_3\times\mathbf{r}}_{rotation}\ \Big] \tag{2.9}$$

where k is the rigidity parameter and $d_i = d(\boldsymbol{\phi}_i, \boldsymbol{\phi}_1)$, $\forall i \in [1, P]$ is the geodesic distance between each feature and the controlled point; $\mathbf{r} = \boldsymbol{\phi}_1 - \mathbf{p}$, is the displacement vector relative to the controlled point; finally, the rotation matrix $\mathbf{R} \in SO(3)$ and position $\mathbf{p} \in \mathbb{R}^3$ encode the end-effector configuration, in the world frame. Jacobian $\tilde{\mathbf{J}}(\mathbf{q}) = \left[\tilde{\mathbf{J}}_1^\intercal(\mathbf{q}), \ldots, \tilde{\mathbf{J}}_P^\intercal(\mathbf{q})\right]^\intercal$ is then used to compute the end-effector velocity:

$$\dot{\mathbf{q}} = \tilde{\mathbf{J}}^{+}(\mathbf{q})\Delta\boldsymbol{\phi} \tag{2.10}$$

where $\Delta\boldsymbol{\phi}$ encodes the objective as the desired DLO displacement[3].

Yet another approach is to learn a control policy directly, without learning an explicit model or deriving a deformation Jacobian [66]. *This thesis explores such* ***model-free*** *methods to solve shape-servoing tasks with multiple problem formulations. Paper B explores online RL on a shape-servoing task with an elastoplastic DLO, using the ReForm simulation sandbox. Paper C applies an offline RL method to a real-world shape-servoing task with, in which Berenson's approach is used as a baseline for comparison.*

[2] The original formulation is valid for an arbitrary number of grippers and included two additional mechanisms to prevent overstretching and allow obstacle avoidance [17].

[3] Notation: $\mathbf{R}_j$ indicates column j and $+$ denotes the Moore-Penrose pseudo-inverse.

Cable-Routing

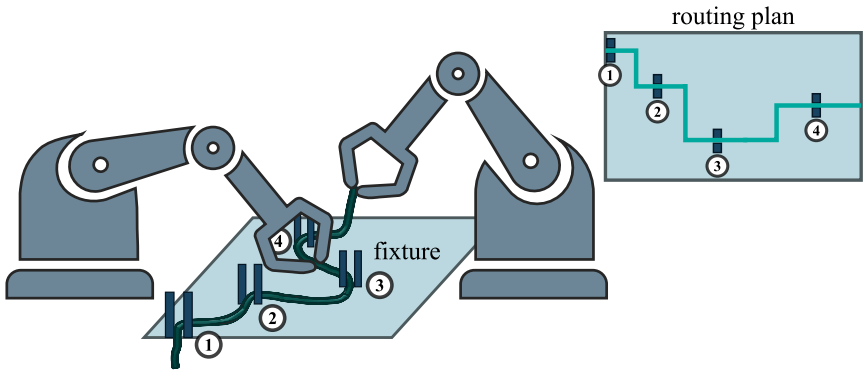


Figure 2.9: Illustration of cable-routing task. It can be considered as a planning problem which alternates between grasping tasks and implicit shape control tasks, for which the objective is to deform the DLO so that it passes through a fixture. In this example, the ultimate goal is to have the DLO passing through all four fixtures in the correct order, as defined by the *routing plan*.

Cable-routing describes tasks in which a cable is to be manipulated through a series of desired positions, without caring about an explicit DLO shape. While shape-servoing is a general task with no particular industrial application in mind, routing a DLO is most often researched due to its potential for **cable and wire harness production** applications. To date, wires and cables are mostly produced by assemblers who manually execute a lot of the tasks required to manufacture wiring systems, e.g. cutting, crimping, etc. However, one of the most challenging tasks to automate in this pipeline is the routing of several cable assemblies to produce a single harness.

Cable harness production is typically organized in a similar arrangement as the one illustrated in Figure 2.9. To facilitate the assembly process, there is often a board with fixtures through which the cables need to be routed. Workers either know the path each cable has to go through, or there are markings on the board, showing the desired routing configuration. The final task in harness production is to hold the cables together using zip ties or cable sleeves. Most related work has focused on the DLO routing task, using fairly simple cables, wires or even ropes. Moreover, while cable harnesses involve routing multiple DLOs, many works consider only one.

Besides the fact that most cable-routing methods rely on planning sequences of *motion primitives* (i.e. low-level skills that serve as building blocks for higher-level tasks), there is very little overlap between related works. Cable-routing implementations proposed in the literature vary in terms of planning algorithm, robot manipulators, end-effectors, sensing modality, control approach, etc. Table 2.2 provides a summary of some of the variations found in the literature. Zhu et al. [73] proposed a vision-based approach, assuming circular contact points instead of fixtures. To that end, a manipulator was fitted with a custom-made end-effector capable of both holding the DLO in place and allowing it to slide when pulled. This was vital, since in their implementation the DLO was never released by the end-effector. Galassi et al. [74] used a similar approach, using tactile sensing instead to control how tightly the DLO was grasped. Monguzzi et al. [58] also explored using tactile sensing for a cable-routing application, with some additional robot skills. Conversely, both Jin et al. [75] and Keipour et al. [50] proposed vision-based spacial representations for cable-routing. Most recently, Luo et al. [76] proposed the first RoL approach, using hierarchical IL to develop a multi-stage implementation.

The two final papers in this thesis dive into cable-routing tasks with a dual-arm robot. They follow similar trends as the work in Table 2.2, with Paper D proposing a complete planning framework using visual information and Paper E focusing on more robust motion primitives using FT sensing.

Table 2.2: Reference table for related works on cable-routing.

[X]	X-Arm	Grasping	Sensing	Motion primitives
[73]	Single	No	Visual	Rotate; Pull.
[74]	Single	No	Tactile	Fix; Corner Fix; Corner Rounding.
[50]	Single	Yes	Visual	Pick; Place.
[75]	Dual	Yes	Visual	Stretch; Cross; Insert.
[13]	Dual	Yes	Visual	*Not specified.*
[58]	Single	Yes	Tactile	Axial alignment; Planar alignment; Contour following; Cable routing.
[76]	Single	Yes	Visual	Route; Perturb; Pick up; Go next.

CHAPTER 3

Reinforcement Learning

RL was briefly introduced in Section 1.2, within the context of robot learning. However, reinforcement learning can be applied to a wide range of sequential decision making problems. Indeed one of the most mediatic successes of RL has been the achievement of superhuman performance in the game of Go. First in 2016 when DeepMind's AlphaGo [77] algorithm defeated grandmaster Lee Sedol, and replicated one year later when it came out victorious over Ke Jie, the highest ranked player at the time. RL has been exceptionally efficient at learning to play a variety of board games like chess [78] and backgammon [79] as well as video games like Atari [80] and StarCraft II [81].

But why did these methods suddenly become so effective? RL had been around for decades, and using ANNs as the main function approximation technique was common practice. The answer lies partially within advances in DL, which enabled more powerful Deep Reinforcement Learning (DRL) algorithms. A key turning point can be traced to 2012, when the Convolutional Neural Network (CNN) architecture later named AlexNet [82] far outperformed past results of the annual ImageNet Large-Scale Visual Recognition Challenge. Besides algorithmic improvements, the increase of available compute power made possible by GPU acceleration was also a catalyst.

Nevertheless, there are other important factors that contributed to the success of DRL in games which unfortunately do not hold true for robotic manipulation. Firstly, these environments are mostly fully observable and deterministic[1], with often discrete and fairly low-dimensional state and action spaces. Secondly, they enable the possibility of self-play, effectively learning to improve on past policies by competing against them. Thirdly, winning a game consists of a simple goal definition which provides an obvious reward signal. Finally, it is important to note that AlphaGo combined RL with planning by Monte Carlo Tree Search (MCTS) methods [77].

In contrast, the state and action spaces in robotic manipulation problems are typically continuous and *high-dimensional*. Furthermore, due to limitations in perception they are mostly partially observable. Indeed the majority of the obstacles to applying RL in robotics is due to interaction with the *real world*, where there are numerous sources of stochasticity and noise. While game agents can remain in a virtual world and therefore play through thousands of matches in seconds, robotic agents have to control physical systems which are limited to real-time execution. This problem is only exacerbated by conservative velocity constraints meant to prevent damage on such expensive pieces of equipment. There have been attempts to use multiple robots in parallel to collect more data [83], however the costs are prohibitive for most academic research institutions. Even the simplest task of resetting the environment is a challenge, given that it either requires some automated approach capable of handling unpredictable states or a human tediously standing by to do it [84]. Three main approaches have been used to overcome this issue: **i.** use LfD to initialize the RL algorithm with a good starting control policy and try to improve from there, **ii.** learn in simulation first and then transfer the learned policy to the real robot, and **iii.** use collected robot interaction data to learn a policy through offline RL, as is explored in Paper C.

In addition, for many robotic tasks there is no clear reward function which fully describes the *goal*. Though sparse rewards, similar to the ones used in games, can be applied to simpler robotic problems such as a peg-in-hole insertion, e.g. assign positive reward once the task is completed, more complex tasks tend to require *reward shaping* to lead the agent to the goal [84]. This problem is addressed in Paper B, for an explicit DLO shape control task.

[1] Backgammon is not fully deterministic since it includes dice rolls and Starcraft II is only partially observable with some randomness.

To summarize, the key challenges to applying RL in robotic applications have been succinctly described by Kober et al. [84] as four curses:

- Curse of dimension
- Curse of real-world samples
- Curse of goal specification
- Curse of under-modeling and model uncertainty

The last curse specifically referred to *modeling* accuracy in simulators. Given that these consist of simplifications of the real world, policies learned in such a setting may not work on the real robot. Consequently, addressing simulation-to-reality transfer is an open research topic, also termed sim-to-real. Despite these curses, the authors compiled a comprehensive list of robotic tasks to which RL was successfully applied. Since the survey's publication in 2013, there has been a growing interest in this field, fueled by the achievements of DRL. For a more recent overview, Hai and Lung [85] published a short review on the state of DRL for robotic manipulation problems, up until 2019.

RL has also been tested on a few DOM applications such as the ball-paddling example in Kober et al. [84], where an elastic string connects the two objects. More recently, low compression strength materials such as cloth and rope have been successfully manipulated without the use of demonstrations [86]. This is in contrast with previous work on cloth folding and hanging which combined RL and LfD [49]. Reinforcement learning has also been used to solve modified classical robotics tasks, such as a peg-in-hole task where the insertion is made of foam [87], and its converse where a soft cable is inserted into a rigid hole [88]. There have even been examples from robotic surgery applications, such as pattern cutting in gauze with DRL policies for tensioning [89].

For the rest of the chapter, the goal will be to introduce RL theory, starting from basic concepts and culminating in the methods relevant for Part II. At its core, RL is a computational approach for an **agent** to learn how to achieve a goal by interacting with the **environment**, through trial-and-error. This agent-environment interaction, illustrated in Figure 3.1, can be modeled as a Markov Decision Process (MDP). Notably, RL is intimately related to optimal control theory and Dynamic Programming (DP). However, these methods typically rely on an explicit model of the environment dynamics.

3.1 MDP Formulation

Markov decision processes provide a useful formalism to describe sequential decision making problems. For this reason, it is common practice to frame RL problems as MDPs. Major textbooks on RL mainly focus on MDPs with finite state and action spaces [90]–[92], however robotic control is better characterized by continuous spaces. Since both cases will be considered throughout this chapter, Definition 1 provides a general description of an MDP.

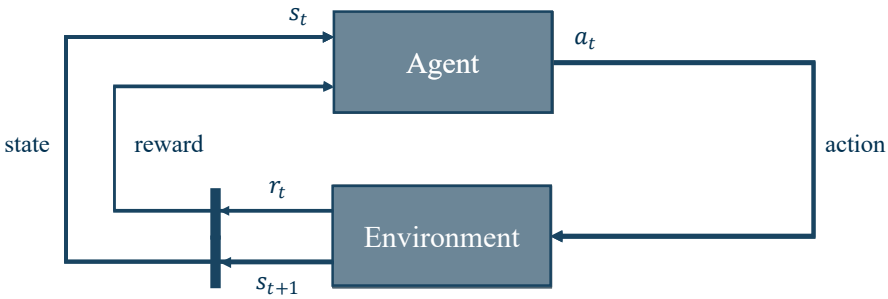


Figure 3.1: Illustration of agent-environment interaction. The agent is in state s_t and takes action a_t, leading to a reward r_t and new state s_{t+1}.

Definition 1: Markov Decision Process *An MDP is defined as a tuple* $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$*, where: Sets* $\mathcal{S}$ *and* $\mathcal{A}$ *represent the state and action spaces, respectively, which can be discrete or continuous. The action space may in some cases also be a function of the state,* $\mathcal{A}(s)$ *with* $s \in \mathcal{S}$*. For continuous spaces, it is assumed that* $\mathcal{S} \subseteq \mathbb{R}^{D_s}$ *and* $\mathcal{A} \subseteq \mathbb{R}^{D_a}$*, where* $D_s, D_a \in \mathbb{N}$ *are the dimensions. The aforementioned spaces characterize the MDP as finite or infinite, e.g. if both spaces are finite the MDP is finite. When the state space is continuous the dynamics of the environment are represented by the probability density function* $p(s_{t+1}|s_t, a_t)$*, which describes the probability of transitioning to state* s_{t+1}*, when in state* s_t *the agent takes action* a_t*. For a discrete state space, this is given by a probability mass function. The objective of the task is encoded by a scalar reward function, which may depend on the current state* $r(s_t)$*, both state and action* $r(s_t, a_t)$*, and even the successive state* $r(s_t, a_t, s_{t+1})$*. Finally,* $\gamma \in [0, 1]$ *is the discount factor which defines how much weight is placed on future rewards in the objective, where* $\gamma = 0$ *implies none, and* $\gamma = 1$ *implies equal weighting for all future rewards.*

For simplicity, discrete-time MDPs are considered here, though much of the theory which will be introduced can be extended to continuous-time [93]. Therefore, the agent-environment interaction leads to a discrete *trajectory*:

$$h_{1:T} = \{s_1, a_1, r_1, \ \dots \ , s_t, a_t, r_t, s_{t+1}, \ \dots \ , s_T, r_T\}$$

where T is the time of termination[2]. An MDP is said to be finite-horizon if $T < \infty$, otherwise the MDP is said to be infinite-horizon. Sutton and Barto [91] further classify tasks as *episodic* if they can be broken down into a sequence of episodes, such as finite-horizon cases, or *continuing* for infinite-horizon MDPs. It is possible to unify these classes if termination is assumed to be equivalent to reaching an absorbing state s_T, for which any action leads to a transition to itself, without any reward, i.e. $s_t = s_T, r_t = 0, \ \forall t > T$ [91].

The objective in MDP problems is defined in terms of the *return*, R_t:

$$R_t \doteq \sum_{k=t}^{T} \gamma^{k-t} r_k = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots + \gamma^{T-1} r_T = r_t + \gamma R_{t+1} \quad (3.1)$$

where r_k is the immediate reward at time k and $T \in [0, \infty]$[3]. The return R_t expresses the sum of future rewards at time step t, and can be related to the successive return R_{t+1}, leading to a recursive property which is exploited in DP algorithms. For a problem to be modeled as an MDP, the Markov property must hold, i.e. the state needs to be a sufficient statistic for predicting the future, independently from past observations. Furthermore, MDPs are assumed to be stationary, i.e. functions r and p are not time-dependent.

MDPs were initially used to formulate dynamic programming algorithms, which work by recursively finding solutions to sub-problems and eventually converging to an optimal policy for the global problem. In essence, a *policy* defines how the agent interacts with the environment. Therefore, a stochastic policy $\pi(a|s) = \mathbb{P}(a_t = a|s_t = s)$ describes the probability of taking action a in state s, at time step t. Policies can also be deterministic, i.e. $\pi(s) = a$, in which case they simply map states to actions, $\pi : \mathcal{S} \to \mathcal{A}$. An important construct in RL, which originated in DP, is the concept of a value function.

[2] While the convention throughout this thesis is that a reward at time t is the result of taking action a_t when in state s_t, this differs from the convention used in [91].

[3] With the caveat that when $T = \infty$ the discount factor must satisfy $\gamma < 1$, otherwise the sum becomes undefined.

The *state-value function*, $v_\pi(s)$, expresses the expected return conditioned on starting in a state s, and following policy π thereafter. Similarly, the *action-value function*, $q_\pi(s, a)$, expresses the expected return further conditioned on taking a specific action a. Both functions are defined below:

$$v_\pi(s) \doteq \mathbb{E}_\pi \left[R_t | s_t = s \right] \tag{3.2}$$

$$q_\pi(s, a) \doteq \mathbb{E}_\pi \left[R_t | s_t = s, a_t = a \right] \tag{3.3}$$

Value functions aim to guide the search for an optimal policy π_*, i.e. the policy which maximizes the expected return. These also lead to variants of the *Bellman equation*. Indeed, the two main DP algorithms, Policy and Value Iteration, are built on iterative updates using Bellman equations, and provide the theoretical basis for many RL algorithms. For instance, the Bellman optimality equation for the action-value function is given by:

$$q_*(s, a) = \mathbb{E}\left[r_t + \gamma \max_{a'} q_*(s_{t+1}, a') | s_t = s, a_t = a \right] \tag{3.4}$$

Since DP is *model-based*, it can only be applied when the environment dynamics $p(s_{t+1}|s_t, a_t)$ are known and the state and action spaces are relatively small. Alternatively, *model-free* RL can be explored through methods where the value function is approximated, as in Section 3.2, or a policy is directly optimized, as in Section 3.3. While both alternatives are less sample efficient than model-based algorithms, learning a model is also a challenging task [94].

Goal-Conditioned RL A particularly relevant MDP formulation which is not represented in Definition 1 is the one used for Goal-Conditioned Reinforcement Learning (GCRL). While the standard MDP formulation assumes a single goal is to be achieved by the agent, e.g. win a game of Go, there are many robotic problems that are better characterized as multi-goal, e.g. a simple reaching task will be dependent on what location or object is to be reached.

GCRL differs from standard RL since it involves augmenting the state with an additional goal $g \in \mathcal{G}$, where $\mathcal{G}$ denotes the goal space. This leads to a goal-augmented MDP formulation defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{G}, p, r, \gamma)$, which has a goal-dependent reward function, $r(\cdot, g)$. Moreover, the learned policy also depends on the goal, for instance in the deterministic case $\pi(s, g)$ is a mapping $\pi : \mathcal{S} \times \mathcal{G} \to \mathcal{A}$. Liu et al. [95] provide a survey which covers the challenges of GCRL and some of the algorithms proposed to address them.

3.2 Value Function Approximation

Monte Carlo (MC) methods provide a simple strategy to approximate value functions by sampling complete trajectories from the agent-environment interaction, which can be real or simulated. MC algorithms work by estimating the average return, based on the rewards observed throughout an episode until termination. Therefore, they can only be applied to episodic tasks, for which the return can be explicitly computed by moving backwards in time, i.e. $R_t = r_t + \gamma(r_{t+1} + \gamma(\dots(r_{T-1} + \gamma R_T))$, with $t < T - 2$.

Temporal Difference (TD) learning, is an alternative approach which does not require sampling full episodes. While MC methods use the episode return R_t to update the value estimate, TD(n) methods *bootstrap* using current value estimates as the target, e.g. $R_t^{(1)} = r_t + \gamma q(s_{t+1}, a_{t+1})$ or $R_t^{(2)} = r_t + \gamma r_{t+1} + \gamma^2 q(s_{t+2}, a_{t+2})$, where the (n) denotes the bootstrapping depth. Note that a one-step bootstrap update is also referred to as TD(0). The Sarsa algorithm uses such an update to approximate the action-value function. This method is presented below (in red) together with the Q-learning algorithm (in blue) proposed by Watkins [96].

Algorithm 1 TD Control (Sarsa & Q-learning)

Randomly initialize $\boldsymbol{Q}(s,a), \forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s)$, except $\boldsymbol{Q}(s_T, \cdot) = \mathbf{0}$
for `episode` $= 1 : M$ **do**
  Obtain initial state s_1
  Choose a_1 from current state s_1 using policy derived from $\boldsymbol{Q}$ S
  for $t = 1 : T$ **do**
    Choose a_t from current state s_t using policy derived from $\boldsymbol{Q}$ Q
    Execute action a_t, observe reward r_t and new state s_{t+1}
    $\boldsymbol{Q}(s_t, a_t) \leftarrow \boldsymbol{Q}(s_t, a_t) + \alpha\left[r_t + \gamma \max_{a'} \boldsymbol{Q}(s_{t+1}, a') - \boldsymbol{Q}(s_t, a_t)\right]$ Q
    Choose a_{t+1} from next state s_{t+1} using policy derived from $\boldsymbol{Q}$ S
    $\boldsymbol{Q}(s_t, a_t) \leftarrow \boldsymbol{Q}(s_t, a_t) + \alpha\left[r_t + \gamma \boldsymbol{Q}(s_{t+1}, a_{t+1}) - \boldsymbol{Q}(s_t, a_t)\right]$ S
    $a_t \leftarrow a_{t+1}$ S
    $s_t \leftarrow s_{t+1}$

Algorithm 1 facilitates the comparison of TD control methods based on action-values. Sarsa uses soft updates $\boldsymbol{Q}(s_t, a_t) \leftarrow \boldsymbol{Q}(s_t, a_t) + \alpha\delta_t$, where $\delta_t = R_t^{(1)} - \boldsymbol{Q}(s_t, a_t)$ is the TD error and α is the step-size. While Sarsa has $R_t^{(1)}$ as the bootstrapping target, Q-learning has the return of the optimal policy $r_t + \gamma \max_{a'} \boldsymbol{Q}(s_{t+1}, a')$. Q-learning is an *off-policy* algorithm because it uses action-value estimates from π_* while following another policy π, whereas Sarsa is an *on-policy* method which uses estimates from the same policy π.

Contrary to DP updates that consider the whole distribution of possible states, MC and TD methods only update values of visited states. Therefore, they must guarantee sufficient exploration of the state space. Consequently, the greedy policies from DP methods, which exploit value estimates, need to be modified to include exploratory actions. This introduces a trade-off referred to as the *exploration-exploitation dilemma.* A simple approach is given by the ε-greedy policy, with $\varepsilon \in (0, 1)$:

$$\pi(s) = \begin{cases} a^* = \operatorname{argmax}_{a \in \mathcal{A}} \boldsymbol{Q}(s, a), & \text{with probability } 1 - \varepsilon \\ a \in \mathcal{A}, & \text{with probability } \varepsilon \end{cases} \tag{3.5}$$

which means the total probability of choosing an optimal action is $\pi(a^*|s) = \frac{\varepsilon}{|\mathcal{A}|} + (1 - \varepsilon)$ and any suboptimal action is $\pi(a|s) = \frac{\varepsilon}{|\mathcal{A}|}$, with $a \neq a^*$.

This type of approach effectively embeds exploration into the algorithm and on-policy methods, e.g. Sarsa, rely on it for exploration. Conversely, off-policy algorithms have a *behavior policy* μ, which selects actions and a separate *target policy* π, which is actually updated. This allows the use of deterministic target policies, e.g. greedy, since the behavior policy will continue to sample random actions. For example, in the Q-learning algorithm, actions may be selected based on a separate behavior policy like ε-greedy.

It is possible to unify TD learning with MC methods given that as $n \to T$, the TD updates become equivalent to Monte Carlo updates. However, the optimal value of n is not known *a priori* because it is problem and algorithm dependent. The TD(λ) approach addresses this issue by averaging over several n-step returns to compute an update, where $\lambda \in [0, 1]$ refers to the eligibility trace-decay parameter [91, Chapter 12]. The target of this type of update is the λ-return, defined as the geometrically weighted average of all n:

$$R_t^\lambda \doteq (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)} = (1 - \lambda) \sum_{n=1}^{T-t} \lambda^{n-1} R_t^{(n)} + \lambda^{T-t} R_t \tag{3.6}$$

For the methods presented so far, the state- and action-value functions have been estimated as a vector $\boldsymbol{v} \in \mathbb{R}^{|\mathcal{S}|}$ and a matrix $\boldsymbol{Q} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}$, respectively. Nevertheless, this is only possible if the state and action spaces are bounded and discrete. Otherwise, they must either be quantized or *function approximation* needs to be employed. There are many alternative function approximation techniques however, this work focuses on DNNs that will be briefly introduced in Section 3.4. To apply these methods, the state-value function denoted $v(s|\mathbf{v})$, is parameterized with $\mathbf{v} \in \mathbb{R}^{D_\mathbf{v}}$, where $D_\mathbf{v} \in \mathbb{N}$ is the dimension of the parameter vector. For action-values, the parameterized function may be *action-in*, denoted $q(s, a|\mathbf{w})$ which outputs a scalar, or *action-out*, denoted $q(s, \cdot|\mathbf{w})$ which instead outputs a vector. Choosing between these two formulations depends on whether the RL algorithm requires the q value for a particular action a or for all possible actions in a given state s. Note that the *action-out* formulation implies a finite action space. Action-value function parameters are denoted $\mathbf{w} \in \mathbb{R}^{D_\mathbf{w}}$ with dimension $D_\mathbf{w} \in \mathbb{N}$.

3.3 Policy Approximation

The previous two sections covered *value-based* methods, in which policies are only implicitly defined through the value function, i.e. the *critic*, requiring maximization over actions to either select an action or update the value estimates. Consequently, the algorithms presented so far are not directly applicable to problems with continuous action spaces. An alternative approach is found in *policy-based* methods, which search for an explicit policy i.e. an *actor*, and do not suffer from the same limitation. This section will introduce the key idea behind policy search, with a focus on gradient methods. These will also be combined with value-based strategies in what is referred to as *actor-critic* methods, which improve efficiency by reducing variance.

Policy-based methods are a natural choice for robotic applications, not only due to their applicability to continuous MDPs but also because learning in policy space often requires fewer parameters than in value space. Furthermore, they offer a direct approach to incorporate prior knowledge through the choice and initialization of the policy representation, as well as the inclusion of constraints. Moreover, in value-based methods a small change in the value function may result in large discontinuous changes to the policy which might lead to dangerous actions [97].

To approximate an optimal policy, one needs to define a parameterized policy $\pi_{\boldsymbol{\theta}}$, with parameters $\boldsymbol{\theta} \in \mathbb{R}^{D_{\boldsymbol{\theta}}}$ of dimension $D_{\boldsymbol{\theta}}$. This section will focus on **Policy Gradient** (PG) methods, which further require the policy to be differentiable with respect to its parameters. However, there are also gradient-free optimization methods such as evolutionary algorithms, which may be used otherwise. Depending on the algorithm, this policy can be stochastic $\pi(a|s, \boldsymbol{\theta})$ or deterministic $\pi(s|\boldsymbol{\theta})$. Once a policy parameterization has been defined, it is possible to treat this as an optimization problem, by defining a performance measure $J(\pi_{\boldsymbol{\theta}})$. This measure is defined differently for episodic tasks where the *starting-state* formulation is used and for continuing tasks which require the *average-reward* formulation.

Starting-state formulation The goal is to maximize the total reward over an episode, from a non-random starting-state s_1 to a terminal state s_T:

$$J_{s_0}(\pi_{\boldsymbol{\theta}}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[R_1] = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_{k=1}^{T} \gamma^k r_k \middle| s_1\right] \tag{3.7}$$

which corresponds to the state-value function $v_{\pi_{\boldsymbol{\theta}}}(s_1)$.

Average-reward formulation The goal is to maximize the expected average reward per time step:

$$J_{\bar{r}}(\pi_{\boldsymbol{\theta}}) = \frac{1}{T}\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_{k=1}^{T} r_k\right] \tag{3.8}$$

which for infinite horizon MDPs can be evaluated as a limit:

$$J_{\bar{r}}(\pi_{\boldsymbol{\theta}}) = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_{k=1}^{T} r_k\right] \tag{3.9}$$

For simplicity, this section considers the starting-state formulation for continuous state and action spaces. In either case, the goal is to improve the parameterized policy $\pi_{\boldsymbol{\theta}}$ by updating its parameters $\boldsymbol{\theta}$ through (stochastic) *gradient ascent*:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_a \nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}})$$

where α_a is the step-size parameter and the subscript a stands for actor.

The problem with such PG objectives is that the policy $\pi_{\boldsymbol{\theta}}$ affects not only the actions taken in each state, but it also indirectly affects the state distribution $\rho_{\pi_{\boldsymbol{\theta}}}(s)$, through interaction with the unknown environment dynamics $p\left(s'|s,a\right)$. This makes gradient computation problematic since the effect of policy parameters on the state distribution is not known.

A simple numerical approach to compute the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is by **finite-difference methods**. These approximate derivatives with finite differences by perturbing each parameter θ_k with a small amount ϵ, in each step:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_k} \approx \frac{J(\boldsymbol{\theta} + \epsilon \mathbb{1}_k) - J(\boldsymbol{\theta})}{\epsilon} \tag{3.10}$$

where $\mathbb{1}_k$ is a unit vector with 1 in the k-th component and 0 elsewhere [97].

A more powerful strategy is used by **likelihood ratio methods**, which allow for an analytical computation of the gradient using Theorem 1.

Theorem 1: Policy Gradient *For the objective function $J(\pi_{\boldsymbol{\theta}})$ and any differentiable **stochastic** policy, the gradient is given by*

$$\begin{aligned}\nabla_\theta J(\pi_{\boldsymbol{\theta}}) &\propto \int_{\mathcal{S}} \rho_{\pi_{\boldsymbol{\theta}}}(s) \int_{\mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta}) q_{\pi_{\boldsymbol{\theta}}}(s,a) \mathrm{d}a \mathrm{d}s \\ &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \log \pi(a|s,\boldsymbol{\theta}) q_{\pi_{\boldsymbol{\theta}}}(s,a)]\end{aligned}$$

where $\rho_{\pi_{\boldsymbol{\theta}}}(s)$ is the on-policy distribution under $\pi_{\boldsymbol{\theta}}$. The proportionality sign $\propto$ becomes an equality for the continuing case, while in the episodic case the proportionality factor is the average episode length.

*The theorem was extended to **deterministic** policies by Silver et. al. [98], assuming the underlying MDP satisfies some regularity conditions:*

$$\begin{aligned}\nabla_\theta J(\pi_{\boldsymbol{\theta}}) &\propto \int_{\mathcal{S}} \rho_{\pi_{\boldsymbol{\theta}}}(s) \nabla_{\boldsymbol{\theta}} \pi(s|\boldsymbol{\theta}) \nabla_a q_{\pi_{\boldsymbol{\theta}}}(s,a)|_{a=\pi_{\boldsymbol{\theta}}(s)} \mathrm{d}s \\ &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\nabla_{\boldsymbol{\theta}} \pi(s|\boldsymbol{\theta}) \nabla_a q_{\pi_{\boldsymbol{\theta}}}(s,a)|_{a=\pi_{\boldsymbol{\theta}}(s)}]\end{aligned}$$

which was shown to be a special (limiting) case of the stochastic policy gradient.

The key feature of the PG theorem is that $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ does not include the gradient of the on-policy state distribution $\rho_{\pi_{\boldsymbol{\theta}}}(s)$, despite the fact that this depends on the policy parameters $\boldsymbol{\theta}$.

Therefore, PG methods based on the likelihood ratio only need to find an estimate of the action-value $q_{\pi_{\boldsymbol{\theta}}}(s, a)$. One of the first PG algorithms used the sample return R_t for that purpose, making it an MC implementation. This algorithm, appropriately named REINFORCE, was proposed by Williams [99] and is presented in Algorithm 2.

Algorithm 2 REINFORCE (with baseline)

Randomly initialize policy $\pi(a|s, \boldsymbol{\theta})$, with parameters $\boldsymbol{\theta}$
Randomly initialize differentiable value function $v(s|\mathbf{v})$, with parameters $\mathbf{v}$
for `episode` $= 1 : M$ **do**
  Generate an episode h following $\pi(a|s, \boldsymbol{\theta})$
  for $t = 1 : T$ **do**
    $R_t \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k$
    $A \leftarrow R_t - v(s_t|\mathbf{v})$
    $\mathbf{v} \leftarrow \mathbf{v} + \alpha_b \nabla_{\mathbf{v}} v(s_t|\mathbf{v}) A$
    $R_t \leftarrow A$ ▷ i.e. use advantage function in place of return
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_a \nabla_{\boldsymbol{\theta}} \log \pi(a_t|s_t, \boldsymbol{\theta}) R_t$ ▷ PG update

Since REINFORCE is an MC method, it suffers from high variance and is slow to learn. To help reduce variance, Williams proposed a variant of the algorithm with a *baseline function*, $b(s)$. This can be any function which does not depend on the actions a. The baseline is subtracted from the state-action value in the policy gradient theorem, so that the expectation remains intact:

$$\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[\nabla_{\boldsymbol{\theta}} \log \pi(a|s, \boldsymbol{\theta}) \left(q_{\pi_{\boldsymbol{\theta}}}(s, a) - b(s) \right) \right] \tag{3.11}$$

This difference is called an *advantage function*, $A_{\pi_{\boldsymbol{\theta}}}(s, a)$, typically denoted by uppercase letter A to differentiate it from actions a and action space $\mathcal{A}$. A good baseline is the state-value function $b(s) = v_{\pi_{\boldsymbol{\theta}}}(s)$ which results in:

$$A_{\pi_{\boldsymbol{\theta}}}(s, a) = q_{\pi_{\boldsymbol{\theta}}}(s, a) - v_{\pi_{\boldsymbol{\theta}}}(s) \tag{3.12}$$

In essence, $A_{\pi_{\boldsymbol{\theta}}}$ is a measure of how much better $\pi_{\boldsymbol{\theta}}$ can become. If for some state-action pair $A_{\pi_{\boldsymbol{\theta}}}(s, a) > 0$, this means that the policy would improve if action a was to be selected in state s. For an optimal policy, π^*, all state-action pairs have $A_{\pi_{\boldsymbol{\theta}}}(s, a) < 0$, except the optimal ones for which $A_{\pi_{\boldsymbol{\theta}}}(s, a) = 0$.

Algorithm 2 shows REINFORCE with baseline in blue. Note that the step-size of the value function has subscript b for baseline and not c for critic. This is because the value function is only considered a critic when it is used for bootstrapping from estimated values of successive states [91, Chapter 13.5].

Instead of the MC return R_t, it is possible to choose a different estimate of $q_{\pi_{\boldsymbol{\theta}}}(s,a)$, using a TD learning critic to reduce variance, i.e. $q_{\pi_{\boldsymbol{\theta}}}(s,a|\mathbf{w})$. This leads to the final group of algorithms which will be presented in this chapter, namely **actor-critic** methods. A baseline may still be used to further improve stability, leading to advantage actor-critic methods. However, replacing the true $q(s,a)$ by an approximation does not guarantee that it will represent the true gradient, unless Theorem 2 holds.

Theorem 2: Compatible Function Approximation *If the following two conditions are satisfied:*

1. *The value function approximator $q(s,a|\mathbf{w})$ is compatible with policy $\pi_{\boldsymbol{\theta}}$:*

$$\begin{aligned} \nabla_{\mathbf{w}} q(s,a|\mathbf{w}) &= \nabla_{\boldsymbol{\theta}} \log \pi(a|s,\boldsymbol{\theta})^\intercal \mathbf{w} && \text{stochastic policy} \\ \nabla_a q(s,a|\mathbf{w})|_{a=\pi_{\boldsymbol{\theta}}(s)} &= \nabla_{\boldsymbol{\theta}} \pi(s|\boldsymbol{\theta})^\intercal \mathbf{w} && \text{deterministic policy} \end{aligned}$$

2. *The value function parameters $\mathbf{w}$ minimize the mean-squared error:*

$$\begin{aligned} MSE_{sto} &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\left(q_{\pi_{\boldsymbol{\theta}}}(s,a) - q(s,a|\mathbf{w})\right)^2\right] \\ MSE_{det} &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\left(\nabla_a q_{\pi_{\boldsymbol{\theta}}}(s,a)|_{a=\pi_{\boldsymbol{\theta}}(s)} - \nabla_a q(s,a|\mathbf{w})|_{a=\pi_{\boldsymbol{\theta}}(s)}\right)^2\right] \end{aligned}$$

then the policy gradient is unbiased:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\nabla_\theta \log \pi(a|s,\boldsymbol{\theta}) q(s,a|\mathbf{w})] && \text{stochastic policy} \\ \nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\nabla_\theta \pi(s|\boldsymbol{\theta}) \nabla_a q(s,a|\mathbf{w})|_{a=\pi_{\boldsymbol{\theta}}(s)}] && \text{deterministic policy} \end{aligned}$$

What this theorem expresses is that function approximators have to be linear in the policy features: $\nabla_{\boldsymbol{\theta}} \pi(s|\boldsymbol{\theta})$ or $\nabla_{\boldsymbol{\theta}} \log \pi(a|s,\boldsymbol{\theta})$. Furthermore, the parameters should be the solution to the linear regression problem minimizing the *MSE*. However, in practice this second condition is relaxed to include value estimation methods such as TD learning [98]. Until Silver et al. [98] introduced the possibility of PG for deterministic policies, with the Compatible Off-Policy Deterministic Actor-Critic (COPDAC-Q) algorithm, it was believed that such a formulation could only be obtained using a model [100].

Since COPDAC-Q is an off-policy method with behavior policy $\mu(a|s) \neq \pi(s|\boldsymbol{\theta})$, the gradient becomes slightly different:

$$\begin{aligned}\nabla J_\mu(\pi_{\boldsymbol{\theta}}) &\approx \int_{\mathcal{S}} \rho_\mu(s) \nabla_{\boldsymbol{\theta}} \pi(s|\boldsymbol{\theta}) q_{\pi_{\boldsymbol{\theta}}}(s,a) \mathrm{d}s \\ &= \mathbb{E}_\mu \left[\nabla_{\boldsymbol{\theta}} \pi(s|\boldsymbol{\theta}) \nabla_a q_{\pi_{\boldsymbol{\theta}}}(s,a)|_{a=\pi_{\boldsymbol{\theta}}(s)} \right]\end{aligned} \tag{3.13}$$

where $\rho_\mu(s)$ is the state distribution under policy μ. Moreover, the Q in the name refers to the critic implementation which uses the action-value TD(0) update, as presented in Algorithm 3. Hence, condition 2 of Theorem 2 is technically not satisfied, since the critic parameters $\mathbf{w}$ are updated by TD learning. In practice, they implement the critic as a linear function approximator $q(s,a|\mathbf{w}) = \phi(s,a)^\intercal \mathbf{w}$, from state-action features defined as $\phi(s,a) = a^\intercal \nabla_{\boldsymbol{\theta}} \pi(s|\boldsymbol{\theta})$ to satisfy condition 1.

Algorithm 3 COPDAC-Q (Deterministic PG)

Randomly initialize policy $\pi(s|\boldsymbol{\theta})$, with parameters $\boldsymbol{\theta}$
Randomly initialize value function $q(s,a|\mathbf{w})$, with parameters $\mathbf{w}$
for `episode` $= 1 : M$ **do**
  Obtain initial observation state s_1
  for $t = 1 : T$ **do**
    $a_t \leftarrow \mu(a|s_t)$
    Execute action a_t, observe reward r_t and new state s_{t+1}
    $\delta_t \leftarrow r_t + \gamma q\left(s_{t+1}, \pi(s_{t+1}|\boldsymbol{\theta})\middle|\mathbf{w}\right) - q(s_t, a_t|\mathbf{w})$ ▷ TD-error
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_a \nabla_{\boldsymbol{\theta}} \pi(s_t|\boldsymbol{\theta}) \left(\nabla_{\boldsymbol{\theta}} \pi(s_t|\boldsymbol{\theta})^\intercal \mathbf{w}\right)$ ▷ PG update
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha_c \delta_t \, \phi(s_t, a_t)$

For the PG update, the gradient of the action-value with respect to a is given by: $\nabla_a q(s,a|\mathbf{w}) = \nabla_{\boldsymbol{\theta}} \pi(s_t|\boldsymbol{\theta})^\intercal \mathbf{w}$. Furthermore, for the critic update, the gradient of $q(s,a|\mathbf{w})$ is just the feature vector $\phi(s,a)$. Unlike the REINFORCE algorithm, the PG update is based on the current value of the critic, and then the critic is updated in the direction minimizing the TD-error, δ_t. The COPDAC-Q algorithm was also tested with function approximation using Multilayer Perceptrons[4] (MLPs) for both actor and critic [98].

[4] Tested on a goal-reaching task with a simulated 6-segment octopus arm, where $D_s = 50$ and $D_a = 20$. The MLPs had one hidden layer with $D_{\boldsymbol{\theta}} = 8, D_{\mathbf{w}} = 40$, and an output layer with sigmoidal and linear activation functions for the actor and critic, respectively.

3.4 Deep Reinforcement Learning

Deep reinforcement learning refers to RL methods where DNNs are used to represent a value function, the policy or even the model of the dynamics.

Deep Neural Networks DNNs are artificial neural networks with a multilayer architecture where each layer contains a set of neurons. Much like their biological counterparts, *neurons* are the functional units of an ANN. The perceptron [101] is an artificial neuron model which learns a mapping from input vectors $\mathbf{x}$ to binary outputs $y \in \{0, 1\}$, as a composition of two functions $H(f(\mathbf{x}))$, where $f(\mathbf{x}) = \boldsymbol{w} \cdot \mathbf{x}_i + b$ with weight vector $\boldsymbol{w}$ and bias b, and H is the activation function. For the perceptron algorithm, H is a Heaviside step, but other activation functions include the sigmoid or hyperbolic tangent, and more recently the Rectified Linear Unit (ReLU). In a SL context, the perceptron algorithm updates the parameters $\boldsymbol{w}, b$ in the direction which minimizes the error between the desired output and the current one $y_i - H(\boldsymbol{w} \cdot \mathbf{x}_i + b)$, using a set of sampled input-output pairs $\{(\mathbf{x}_i, y_i)\}$.

MLPs are networks made up of layers of perceptrons, typically with nonlinear activation functions. They are also referred to as *fully connected*, because each neuron in one layer is connected to all neurons in the following layer, as shown in Figure 3.2. CNNs are another type of *feedforward* architecture, which applies parameterized convolutional filters by strides along the input data, thus reducing each local region into a scalar value. For data with a sequential nature, Recurrent Neural Networks (RNNs) have been the predominant architecture. These are not considered feedforward, since they have an internal state that is used as input to the following layers. More recently, Transformer networks [102] have outperformed previous RNN architectures in numerous applications, thanks to their attention mechanism.

Architectures may vary, but their training procedure is relatively similar. Data is used to iteratively update the weights through *backpropagation*, to minimize an error encoding the objective. In SL, the error is defined for either regression or classification objectives. Updates may be done for each data point or in batches, i.e. stochastic or minibatch *gradient descent*, respectively. The stochasticity is determined by a pseudorandom number generator, initialized by a *random seed*. Many algorithmic improvements have also contributed to more efficient ANN training, such as the Adam algorithm for optimization [103] or batch normalization [104] and dropout [105] for regularization.

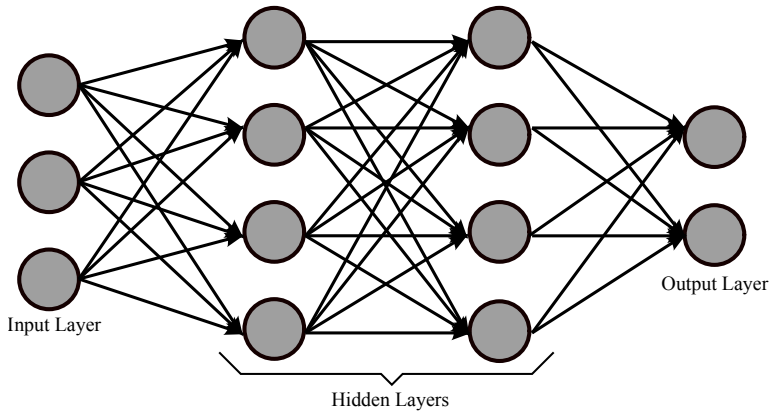


Figure 3.2: Illustration of MLP, i.e. feedforward fully connected neural network.

Despite being powerful function approximators, DNNs can lead to instabilities and even divergence of RL algorithms. This is because convergence of learned parameters to a fixed point is not guaranteed when a nonlinear function approximator is used for value-based algorithms, presented in Section 3.2. Even with simpler linear function approximators, off-policy algorithms such as Q-learning, are no longer guaranteed to converge. This is a consequence of **the deadly triad**, which refers to the combination of *function approximation*, *bootstrapping* and *off-policy training*. Whenever these three conditions are present, even DP algorithms become unstable. For more details on this subject, consult Sutton and Barto [91, Chapter 11.3].

One of the earlier successes of deep reinforcement learning was the proposal of the Deep Q-Network (DQN) algorithm [80]. As the name indicates, it consists of a modification of Algorithm 1, where the $\boldsymbol{Q}$ value is represented by a DNN. A key challenge in training neural networks using sampled RL data, is that optimization algorithms commonly require that samples are independently and identically distributed (iid). Naturally, this assumption does not hold for an RL trajectory, since the state and action distributions are correlated by the interaction between the policy and the dynamics of the environment. To address this issue and make use of the increased efficiency of minibatch gradient methods, Mnih et al. [80] proposed that sampled experience tuples (s_t, a_t, r_t, s_{t+1}) be stored in a *replay buffer* $\mathcal{D}$. For each step, a minibatch $\mathcal{B}$ of uniformly sampled tuples is then used to update the Q-network. The buffer was designed as a first-in, first-out system, with a fixed memory.

Another challenge that needed to be addressed, is that implementing the Q-learning critic as an ANN was unstable. This is due to the bootstrapping nature of the algorithm which means that the network being updated $q(s, a|\mathbf{w})$ by gradient descent $\mathbf{w} \leftarrow \mathbf{w} + \alpha_c \nabla_{\mathbf{w}} L$ is also used to calculate the target value of the *MSE* loss:

$$L \leftarrow \frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \left(y_i - q(s_i, a_i|\mathbf{w})\right)^2$$

where $y_i \leftarrow r_i + \max_{a'} q(s_t, a'|\mathbf{w})$ is the target. To overcome this problem, Mnih et al. [80] proposed using a *target Q network*, with separate parameters $\mathbf{w}'$, so that $y_i \leftarrow r_i + \max_{a'} q(s_t, a'|\mathbf{w}')$. The target network can then be updated at a slower rate, by setting $\mathbf{w}' \leftarrow \mathbf{w}$ only every C steps. Furthermore, the authors also found that clipping the *MSE* term from the update to be between -1 and 1, helped improve stability. However, this was fairly specific to the test domain, as discussed by van Hasselt et al. [106].

Based on the algorithmic changes described above, DQN was shown to reach human-level performance in 49 of the Atari 2600 games. The critic network was modeled as an action-out CNN architecture, which was trained using pixels and games scores as inputs. However, if DQN is an off-policy TD method which uses function approximation, how does the deadly triad affect its performance? Van Hasselt et al. attempt to answer that question in [107].

Deep Deterministic Policy Gradient

As a *value-based* method, DQN is only suitable for discrete action spaces. Building on the COPDAC-Q algorithm and the observations from the DQN training strategy, Lillicrap et al. [108] proposed the Deep Deterministic Policy Gradient (DDPG) algorithm, for continuous action spaces. DDPG also makes use of the replay buffer and target network ideas. They found that in order to avoid divergence, target networks were required both for the critic and the actor. While this slows down the learning, it is highly compensated by its stability. Unlike the DQN strategy, the target networks are updated through soft updates controlled by parameter τ, in order to have them slowly track the learned networks, as shown in item 3 of Algorithm 4. In the original implementation, the exploratory behavior policy μ was obtained by adding noise $\omega_t \sim \mathcal{W}$ to the current policy $\pi(s_t|\boldsymbol{\theta})$. For their experiments, $\mathcal{W}$ was an Ornstein-Uhlenbeck process which enables temporally correlated exploration.

Algorithm 4 Deep Deterministic PG (DDPG)

Initialize actor network $\pi(s|\boldsymbol{\theta})$, with weights $\boldsymbol{\theta}$
Initialize critic network $q(s, a|\mathbf{w})$, with weights $\mathbf{w}$
Initialize target networks q' and π', with weights $\mathbf{w}' \leftarrow \mathbf{w}$, $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}$
Initialize replay buffer $\mathcal{D} \leftarrow \varnothing$
for `episode` $= 1 : M$ **do**
  Initialize a random process $\mathcal{W}$ for action exploration
  for $t = 1 : T$ **do**
    $a_t \leftarrow \pi(s_t|\boldsymbol{\theta}) + \omega_t$, where $\omega_t \sim \mathcal{W}$
    Execute action a_t, observe reward r_t and new state s_{t+1}
    Store transition in replay buffer, $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
  if it is time for network updates **then**
    Sample minibatch of transitions $\mathcal{B} = \{(s, a, r, s')\}$ from $\mathcal{D}$
    1. Update critic networks using target networks:

$$y \leftarrow r + \gamma q'(s', \pi'(s'|\boldsymbol{\theta}')|\mathbf{w}') \qquad \triangleright \text{ compute targets}$$

$$L = \frac{1}{|\mathcal{B}|} \sum_{(s,a,r,s') \in \mathcal{B}} \left(y - q(s, a|\mathbf{w})\right)^2 \qquad \triangleright \text{ MSE}$$

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha_c \nabla_{\mathbf{w}} L \qquad \triangleright \text{ gradient descent step}$$

    2. Update actor network using critic network:

$$J = \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} q\left(s, \pi(s|\boldsymbol{\theta})\middle|\mathbf{w}\right) \qquad \triangleright \text{ mean action-value}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_a \nabla_{\boldsymbol{\theta}} J \qquad \triangleright \text{ PG step}$$

    3. Update target networks, with $\tau \ll 1$:

$$\boldsymbol{\theta}' \leftarrow \tau\boldsymbol{\theta} + (1 - \tau)\boldsymbol{\theta}'$$

$$\mathbf{w}' \leftarrow \tau\mathbf{w} + (1 - \tau)\mathbf{w}'$$

In addition, *batch normalization* [104] was used during network updates. This DL strategy normalizes a minibatch so that each dimension across samples has zero mean and unit variance[5].

[5] This was needed due to the test domain variability. While DQN was tested on the Atari suite with the same input type, DDPG was tested on the continuous control suite with varied inputs of different magnitudes.

Twin-Delayed DDPG

A common issue in *value-based* RL is that function approximation errors lead to value overestimation and consequently suboptimal policies. Fujimoto et al. [109] showed that the problem persists in *actor-critic* settings and proposed the Twin-Delayed DDPG (TD3) algorithm to mitigate it. TD3 is a variant of DDPG with three essential modifications: **i.** two *twin* critic networks are used to stabilize the learning, by taking the smaller of two q values to form the targets (i.e. Clipped Double Q-learning); **ii.** the policy updates are *delayed*, by being applied less frequently than the q function updates (every N-th iteration); **iii.** noise is added to the target actions, as a regularization strategy for the action-value function. This third modification serves to reduce the impact of actions whose value becomes overestimated. Such actions create sharp peaks in value space, which DDPG policies exploit, often leading to undesired behavior. In essence, adding noise helps smooth the value space around similar actions. All three modifications are indicated in Algorithm 5.

Choosing RL Algorithm DDPG (Algorithm 4) was used in Papers A-B. However, more recent PG methods such as TD3, have been shown to lead to comparatively better performance. Thus, a variant of TD3 (Algorithm 5) was used in Paper C. Choosing these methods over other algorithms listed in Table 3.1 was in part motivated by their relative simplicity, but also guided by the observations made by Henderson et al. [110]. In particular, they highlight the difficulties in comparing DRL algorithms across different publications. Specifically, they attempt to address the following questions:

- How much do *hyperparameter settings* influence the reported algorithm performance?
- How does the *network architecture* of the policy and value functions affect the performance?
- How does *reward scaling* affect results and why is it used?
- How do *random seeds* affect the reported performance? Can results be distorted by averaging an improper number of trials?
- How do the *environment* properties affect variability in performance?
- Are commonly used *codebase* implementations comparable?

Table 3.1: Reference table for state-of-the-art policy gradient methods, all with an actor-critic formulation. DDPG has been considered one of the more efficient off-policy DRL methods [111]. Since then, several algorithms have been proposed which are listed in this table, together with the policy learning approach and actor type.

[X]	Algorithm	X-Policy	X Actor
[108]	Deep Deterministic Policy Gradient (DDPG)	Off	Deterministic
[109]	Twin-Delayed DDPG (TD3)	Off	Deterministic
[112]	TD3 + 4 additions (TD7)	Off	Deterministic
[113]	Actor Critic with Experience Replay (ACER)	Off	Stochastic
[114]	Soft Actor-Critic (SAC)	Off	Stochastic
[115]	Advantage Actor Critic (A2C)	On	Stochastic
[116]	Trust Region Policy Optimization (TRPO)	On	Stochastic
[117]	Proximal Policy Optimization (PPO)	On	Stochastic

It is a well-known fact that *hyperparameter settings* and *network architecture* have significant effects on performance of any deep learning algorithm. Henderson et al. [110] further show that *reward scaling* can have a large impact, and recommend a more principled approach, e.g. Pop-Art [106]. They also demonstrate that the variance between trials with different *random seeds* is enough to create statistically different performance distributions. Moreover, algorithm performance can vary across *environments* and it is not clear which is the universally best performing method.

Most DRL papers present results in the form of learning curves, as some function of the return. However, without understanding what the returns actually indicate, these curves can be misleading since the algorithm may converge to local optima without ever reaching the desired goal. This is observed in Paper B, where some reward definitions lead to suboptimal solutions, as demonstrated in the qualitative plots. Consequently, the authors recommend the inclusion of a qualitative analysis of the results. Finally, they find that implementation differences between *codebases* of the same algorithm can have drastic effects on training performance. Thus, it is important to use standardized codebases or make new implementations public, with all hyperparameter details [110]. Therefore, Papers A-B used the *rlpyt* [118] codebase and Paper C, used the *d3rlpy* [119] library.

The difficulty in choosing RL algorithm is compounded by the various algorithmic improvements which can be combined with these methods. Indeed, several works have proposed such improvements, namely Prioritized (PER) [120] and Hindsight Experience Replay (HER) [121], Generalized Advantage Estimation (GAE) [122], TD-regularization [123] and Phasic Policy Gradient (PPG) [124]. As every algorithmic improvement often comes with additional hyperparameters to tune, it makes the search space even larger.

3.5 Offline RL

All of the algorithms presented so far are formulated for *online* learning, requiring agent interaction with the environment while iterating through different policies. Due to the challenges of applying RL in robotics, highlighted at the start of this chapter, there has been a push towards *offline* learning algorithms. The key idea is to use previously collected interaction data, and train a policy without any additional online interaction, similarly to behavior cloning. While the idea of offline RL is appealing, it also raises several challenges. The tutorial by Levine et al. [125] covers the subject in greater depth. Theoretically, any off-policy algorithm can be used for offline RL, however that alone is not sufficient to achieve good performance, and online fine-tuning is often required. In general, offline RL methods include algorithmic changes that keep the policy close to the data distribution, by preventing value overestimation of state-action pairs not represented in the dataset. Several algorithms have been proposed for that purpose, such as Conservative Q-Learning (CQL) [126] and Implicit Q-Learning (IQL) [127].

Fujimoto and Gu [128] propose the TD3+BC algorithm, which is a variant of Algorithm 5, with two minimal modifications. The main modification consists on regularizing the learned policy with a behavior cloning term, whose strength is controlled via a parameter λ. Striking balance between the RL (i.e. maximizing q) and the imitation (i.e. minimizing the BC term) objectives is dependent on the scale of the action-value, therefore λ is normalized:

$$\lambda = \frac{\alpha}{\frac{1}{|\mathcal{B}|}\sum_{(s,a)\in\mathcal{B}} |q(s,a)|} \tag{3.14}$$

where α is a hyperparameter and the mean action-value is computed over minibatches during training.

The secondary but impactful modification was to normalize the states over the offline dataset, to have zero mean and unit variance. Note that this normalization step is not shown in Algorithm 5. Although CQL and IQL were also tested, ultimately TD3+BC was used in Paper C.

Algorithm 5 Twin Delayed DDPG plus BC (TD3+BC) *online*

Initialize actor network $\pi(s|\boldsymbol{\theta})$, with weights $\boldsymbol{\theta}$
Initialize twin critic networks $q_k(s, a|\mathbf{w}_k)$, with weights $\mathbf{w}_k$ for $k = \{1, 2\}$ **i.**
Initialize target networks q'_k and π', with weights $\mathbf{w}'_k \leftarrow \mathbf{w}_k$, $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}$
Initialize replay buffer $\mathcal{D} \leftarrow \varnothing$
for `episode` $= 1 : M$ **do**
  for $t = 1 : T$ **do**
    $a_t \leftarrow \pi(s_t|\boldsymbol{\theta}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma)$
    Execute action a_t, observe reward r_t and new state s_{t+1}
    Store transition in replay buffer, $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
  if it is time for network updates **then**
    Sample minibatch of transitions $\mathcal{B} = \{(s, a, r, s')\}$ from $\mathcal{D}$
    1. Update critic networks using target networks:

$$\tilde{a} \leftarrow \pi'(s'|\boldsymbol{\theta}') + \epsilon, \text{ where } \epsilon \sim \text{clip}\left(\mathcal{N}(0, \tilde{\sigma}), a^{\min}, a^{\max}\right) \quad \textbf{iii.}$$

$$y \leftarrow r + \gamma \min_{k=1,2} q'_k(s', \tilde{a}|\mathbf{w}'_k) \quad \triangleright \text{ compute minimal targets}$$

$$L_k = \frac{1}{|\mathcal{B}|} \sum_{(s,a,r,s') \in \mathcal{B}} \left(y - q_k(s, a|\mathbf{w}_k)\right)^2$$

$$\mathbf{w}_k \leftarrow \mathbf{w}_k - \alpha_c \nabla_{\mathbf{w}_k} L_k \quad \triangleright \text{ gradient descent step}$$

    if t mod N **then** **ii.**
      2. Update actor network using critic network:

$$J = \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} \lambda \, q_1\left(s, \pi(s|\boldsymbol{\theta})\middle|\mathbf{w}_1\right) - \left(\pi(s|\boldsymbol{\theta}) - a\right)^2$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_a \nabla_{\boldsymbol{\theta}} J \quad \triangleright \text{ PG step}$$

      3. Update target networks, with $\tau \ll 1$:

$$\boldsymbol{\theta}' \leftarrow \tau\boldsymbol{\theta} + (1 - \tau)\boldsymbol{\theta}'$$

$$\mathbf{w}'_k \leftarrow \tau\mathbf{w}_k + (1 - \tau)\mathbf{w}'_k$$

3.6 RL Simulation Environments

Much like the ImageNet challenge for CV methods, several simulation environments were created in order to compare RL algorithms. To facilitate this comparison, OpenAI released the Gym [129] toolkit for developing RL simulation environments following the same formalism[6]. This standardization makes testing RL algorithms in different applications much easier, as all tasks have the same Python interface. Notably, the environments used for benchmarking DRL in continuous control tasks by Duan et al. [111], were made available as part of Gym [129]. These included classic control problems from RL literature, as well as more complex locomotion tasks. Though the agents in these environments can be viewed as simplified robots, they do not actually correspond to any real-world hardware. It was with the publication of HER [121] that environments including real robotic systems like Fetch [131] and the Shadow Hand [132] were added to Gym.

Many other third-party environments were created using the same standard. That is the case for the work introduced in Paper A which presents ReForm, a robot learning sandbox for DLO manipulation. Also included in the paper is an overview of environments for robotic manipulation, although some related work was missed such as robo-gym, which simulates multiple commercially available industrial robots using Gazebo [133]. A particularly important omission in the context of deformable object manipulation is the concurrent work by Huang et al. [134] which published the PlasticineLab environments after the final version deadline for Paper A. These, together with ReForm and SoftGym [43], consist of the only RL benchmarking environments currently available which address the challenges of DOM.

SoftGym, was released while the work on ReForm was underway. This library uses NVIDIA FleX as the physics engine, which can simulate soft objects such as clothes and ropes, as well as liquids. However, elastoplasticity is not available in any of the environments. Conversely, PlasticineLab does include an elastoplastic material model implemented using Taichi [135]. Nevertheless, all deformable objects in the provided environments behave like Plasticine. That includes the DLO manipulation task, where a rope is modeled as a long Plasticine piece. This is in contrast with the AGX Dynamics engine used for ReForm, which supports DLO models with a range of material properties.

[6] In 2022, maintenance of the Gym library was transferred to the Farama Foundation, in what is now Gymnasium [130].

CHAPTER 4

Research Contributions

While the full manuscripts of the appended publications are found in Part II, this chapter provides a brief summary of their respective research contributions. Before that, a detailed description of the author's individual contributions for each paper is given in Table 4.1.

Table 4.1: Overview of author contributions for each paper appended in Part II. Work is divided into three main areas, from the technical conceptualization, to the practical implementation and finally to the writing of the publication. **Legend:** ○ - nearly no contribution; ◔ - minor contribution; ◑ - contribution; ◕ - major contribution; ● - nearly sole contribution.

Paper	Concept	Implementation	Publication
A	●	◕	◑
B	●	●	●
C	◕	◕	●
D	◑	◔	◑
E	◑	○	◕

This thesis is a continuation of the work included in the previously published Licentiate thesis [1], which was built on *simulation* results from Papers A-B. Since then, there has been an effort to move to *real-world* experiments with Papers C-E. As a consequence, the literature study in Part I has been enhanced and expanded to incorporate the additional papers in Part II. Figure 4.1 shows a diagram indicating how the contributed papers fit within DOM research and reflecting the type of experiments carried out. Next, a summary of each publication is presented, together with its contribution towards the research objective introduced in Section 1.3.

4.1 Summary of Publications

Paper A presents ReForm, a new RoL sandbox to facilitate research on DLO manipulation in simulation. Paper B addresses a shape-servoing task from ReForm with elastoplastic properties, to achieve a single goal, through online RL. Paper C tackles a similar but more complex multi-goal problem on a real-world robotic setup using offline RL. Finally, Paper D robustly solves a cable-routing task with randomly placed fixtures based on vision information, while Paper E shows how a similar task can be solved using only force information for state estimation. While the author was the major contributor to the shape-servoing papers, in the cable-routing papers the author assumed a more supervisory role. The primary contribution was made towards formalizing the research concepts for publication, whereas the main development work was carried out by Master's students.

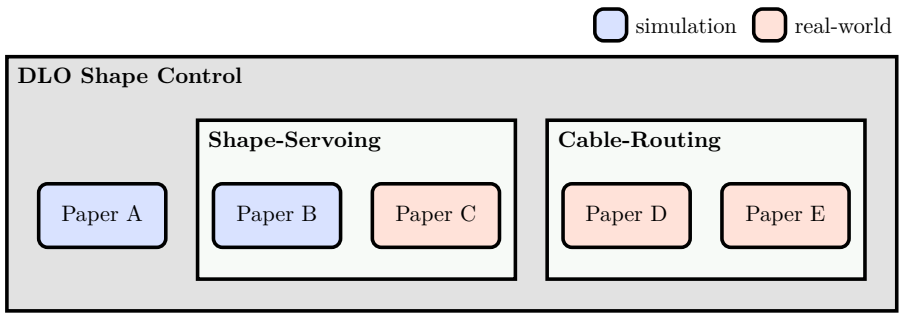


Figure 4.1: Diagram of contributions within DOM research.

Paper A

ReForm: A Robot Learning Sandbox for Deformable Linear Object Manipulation

Rita Laezza, Robert Gieselmann, Florian T. Pokorny, Yiannis Karayiannidis

Published in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4717–4723.

 DOI: 10.1109/ICRA48506.2021.9561766

This paper presents ReForm, a novel robot learning sandbox for deformable linear object manipulation. It is meant as a tool for testing and benchmarking DLO manipulation strategies, particularly through RoL. It consists of six environments representing important characteristics of deformable objects, with material properties ranging from pure elasticity to elastoplasticity. Three explicit and three implicit shape control tasks are included. Notably, the explicit shape control tasks can easily be used for multi-goal RL. The sandbox also includes problems such as self-occlusions and -collisions. ReForm is built as a modular framework which allows for the choice of parameters such as end-effector DoFs, reward function and type of observation. Since vision data is supported, CV methods can also be tested, for example to address self-occlusion in DLO tracking. For each DLO manipulation task in ReForm, initial benchmarking experiments were carried out using the DDPG algorithm to obtain a baseline for online RL performance.

The initial concept of ReForm was led by the author, with the development of the base Gym interface with AGX Dynamics and all three explicit shape control environments. As this work demanded more resources, it evolved into a joint effort with our partners at KTH, within the WASP collaboration project. Robert Gieselmann developed the implicit shape control environments and contributed to the final publication, resulting in a shared first authorship.

Research Objective While ReForm has been tested using an RL algorithm, it can easily be used to develop DNN models of robot-object interactions based on vision and force data, since both sensing modalities are available in the simulation sandbox.

Paper B

Learning Shape Control of Elastoplastic Deformable Linear Objects

Rita Laezza, Yiannis Karayiannidis

Published in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4438–4444.

 DOI: 110.1109/ICRA48506.2021.9561984

This paper introduces an explicit shape control task for DLOs with elastoplastic properties. The goal is to deform a DLO into a desired shape, with the added difficulty that a permanent deformation may occur. In such a case, it becomes challenging to define a distance measure which uniquely guides the manipulation objective. RL offers a data-driven approach to tackle this problem without needing a model. However, it still requires a reward definition that adequately describes the goal. The key contribution in this work, is the proposal of a reward function based on a shape representation using discrete curvature and torsion. The impact of this reward definition on the RL problem is studied using the DDPG algorithm. Notably, simulation experiments show that to converge to the correct shape, the reward must include the proposed representation.

Research Objective This work made use of ReForm to design an RL control policy in simulation. It mainly addressed the DLO shape representation problem and proposed a related reward shaping approach.

Paper C

Offline Goal-Conditioned Reinforcement Learning for Shape Control of Deformable Linear Objects

Rita Laezza, Mohammadreza Shetab-Bushehri, Gabriel A. Waltersson, Erol Özgür, Youcef Mezouar, Yiannis Karayiannidis

Submitted to *IEEE Robotics and Automation Letters (RA-L)*.

Available as a preprint on arXiv:2403.10290.

This paper focuses on a real-world DLO shape control task using offline GCRL techniques. More specifically, dual-arm shape-servoing of a DLO on a plane is addressed. Two types of material properties a evaluated, with an elastic cord and a soft rope. A modular implementation is proposed, using an ARAP-based method for visual state tracking of the DLO and an IK controller to execute the action from an RL policy on the robot (ABB dual-arm YuMi). The key contribution of this work is the ability to leverage limited real data to successfully train ANN policies. This is achieved through a data augmentation approach, inspired by the HER principle. By randomly sampling intermediate shapes, and setting them as goals, the learned policies are shown to achieve better results. Furthermore, the proposed method is compared with a well established shape-servoing algorithm, indicating its relative improvement for both DLOs. The TD3+BC offline RL method is used, which incorporates a BC regularization term that is also investigated in this work.

Research Objective This work also addressed the design of RL policies, but in a real-world robotic setup. Compared to Paper B, it moved from online to offline RL algorithms, and from a single- to a multi-goal MDP formulation.

Paper D

Planning and Control for Cable-Routing with Dual-Arm Robot

Gabriel A. Waltersson, **Rita Laezza**, Yiannis Karayiannidis

Published in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1046–1052.

 DOI: 10.1109/ICRA46639.2022.9811765

This paper presents a vision-based approach for a cable-routing task, with arbitrarily placed fixtures. The key contribution of the paper is a task-space planner that generates instructions, using a heuristic algorithm, and employs a replanning strategy, based on a Genetic Algorithm (GA), if problems occur. More specifically, the planner builds a roadmap from predefined motion primitives and generates trajectory parameters for the controller. If the trajectory parameters are unfeasible, a replanning module uses the GA to find a new feasible solution. As a dual-arm implementation is considered, both coordinated and individual control strategies are used to define the necessary motion primitives. The paper relies on a CV system for estimating the poses of the fixtures and tracking the DLO. The proposed framework is tested in real-world experiments with an ABB dual-arm YuMi robot, demonstrating a 90% success rate for three-fixture problems.

Research Objective As the first cable-routing paper in this thesis, the goal was to lay some groundwork towards future data-driven research. While no RL policy or DNN model were explored in this work, the GA is an alternative ML approach to solve the challenging replanning problem. Future work could replace the GA with an RL policy. This work was carried out using visual information on a real-world robotic setup. Importantly, the control architecture developed for this work was leveraged in both Paper C and Paper E.

Paper E

Feel the Tension: Manipulation of Deformable Linear Objects in Environments with Fixtures using Force Information

Finn Süberkrüb, **Rita Laezza**, Yiannis Karayiannidis

Published in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11216–11222.

 DOI: 10.1109/IROS47612.2022.9982065

This paper presents a purely force-based approach for a cable-routing task. Its first contribution is the proposed DLO graph model which enables blind manipulation when the object is kept under tension. The second contribution is the online model estimation procedure which keeps the model updated, based on the perceived DLO elasticity and contacts with the fixture environment. This paper further introduces a set of elementary sliding and clipping manipulation primitives, which can be defined based on the proposed model. The individual manipulation primitives as well as the model estimation method are tested in a real-world cable-routing experiment, using an ABB dual-arm YuMi robot.

Research Objective This was the only work to make use of force information on a real-world robotic setup. While the proposed model was not encoded as a DNN, its parameters were estimated using a data-driven method. Similarly to Paper C, the motion primitives proposed in this work could be used as building blocks for an RL-based planner.

CHAPTER 5

Concluding Remarks

To recapitulate, Part I provides an overview of the research context of the contributed papers, found in Part II. Chapter 1 started with a birds-eye view of deformable object manipulation and robot learning. This was followed by a more in-depth treatment of DLO manipulation in Chapter 2 and RL theory in Chapter 3. After reading these three chapters, readers should be better equipped to understand both the research context and the contributions of the papers, summarized in Chapter 4.

For concluding remarks on the research contributions of this thesis, refer to Section 4.1. Instead, Section 5.1 provides more general reflections on RoL for DOM applications, aimed at researchers in the field that may stumble across this thesis. Much like RL, research requires a lot of trial-and-error learning. Indeed, a lot of efforts made towards this thesis are not visible in the appended papers. One could say that several exploratory "actions" were taken to investigate if a particular research direction could be fruitful, which led to important insights but not "rewarded" with a publication. Therefore, some of the lessons learned in this process are discussed in more detail. Finally, we look towards the future with Section 5.2, which covers upcoming research in DOM applications, as well as healthcare applications.

5.1 Reflection

In the final chapter of the Licentiate thesis [1], there was already a moment of reflection regarding the goal of this research project, presented in Section 1.3. Specifically, a few relevant questions were posed regarding how we humans *approach DOM problems in our daily lives*:

"Is trial-and-error learning such as RL, responsible for all human dexterity? Or are many manipulation skills learned by imitation? How are the goals for each task defined? When folding laundry, does the exact position of each fold matter, or is the purpose to flatten clothes while reducing storage area? • How many tasks would humans be able to solve if instead of having compliant five-fingered hands, they had rigid two-fingered parallel grippers? Would tying shoelaces be easy or even possible? Which tasks are strictly dependent on vision or tactile feedback?"

The first few questions related to human motor learning in general. There is evidence that some form of RL is behind human motor skills, at least by regulating an underlying motor learning process [136]. There has also been significant research on mirror neurons, which have been shown to contribute towards imitation behavior, having a key role in sensorimotor learning [137]. What is more unclear is how our reward system impacts our learning of motor skills and how goals are set. Extrinsic rewards have been shown to improve motor skill learning. Interestingly, monetary rewards have a higher impact on motor skill performance than performance-based feedback [138]. Indeed money is considered to be a *secondary reward* since it is directly associated with *primary rewards* e.g. food, that are related to survival and reproduction. While actions taken towards some primary or secondary reward may not feel rewarding, e.g. going to work, there are activities that feel rewarding in and of themselves, which is believed to be a result of an intrinsic reward system [139]. Perhaps as we learn more about ourselves, new ideas may lead to better RoL algorithms.

The final questions related to how much the human body has evolved to make dexterous motor control possible, including DOM. Perhaps more importantly, how much does the human anatomy and physiology facilitate motor learning as compared to a robot. There is still a hardware bottleneck in robotics, due both to the lack of dexterous end-effectors and robust sensing

technologies, as mentioned in Chapter 1. Even the way manipulators are constructed is far from the biological arms they aim to emulate. While biologically inspired robotics is a growing field, the aim is often to understand the underlying principles and adapt them to simpler mechanical structures [140]. In the context of DOM, human-like robotic hands are likely not going to become the most practical solution, at least in the near future. This is supported by recent work from OpenAI towards solving a Rubik's cube with a Shadow hand [132] through RL [141][1]. Instead, soft grippers are a more practical approach, since they can adapt to the shape of deformable objects without needing overly complex software, e.g. for pick and place tasks [142]. Thus, solutions with soft robotics are more likely to become commonplace for many DOM problems, e.g. in food processing and agriculture.

Indeed, more robust solutions with simpler mechanics, such as suction grippers, are likely to continue to be chosen over AI-based dexterous robot hands for a few years to come. This is in line with Moravec's paradox [143], which postulates that, reasoning requires very little computation when compared to sensorimotor skills, contrary to human intuition. In other words, abilities that we classify as complex, such as language, are actually simpler than those that we take for granted, like dexterous manipulation. Recent successes in Large Language Models (LLMs)[2] while robots are still struggling to even walk reliably on two feet[3], seem to support this idea.

DOM is truly an example of Moravec's paradox, since so many tasks that we do without even thinking, are extremely difficult to be executed by a robot. While there are some industrial tasks which can perhaps be solved using current technology, many will require general manipulation abilities. Therefore, when it comes to DOM research, the argument by Demis Hassabis[4] likely applies: *"solve intelligence, and then use that to solve everything else"*. In the context of DOM, it makes more sense to solve general robotic manipulation, and then use that to solve DOM. For the most part, looking for general solu-

[1] In 2021, OpenAI disbanded its robotics team. Co-founder, Wojciech Zaremba, points to the lack of training data as the main motivation behind decision.

[2] For example, ChatGPT by OpenAI which was released in November, 2022.

[3] While Honda's ASIMO (Advanced Step in Innovative Mobility), introduced already in 2000, was the first commercially available bipedal robot, it was meant for showcasing purposes and not of any practical utility. Since then, not many other legged robots have been released. For example, Spot, the four-legged robot by Boston Dynamics, only became commercially available in September, 2019.

[4] Co-founder and CEO of DeepMind.

tions to DOM is impractical. Until robotic manipulation is solved, it is best to look for specific solutions to particular tasks, such as cable-routing.

RL is still a promising approach for achieving general robotic manipulation abilities. While the MDP formulations used in this thesis were specific to DLO manipulation, the learning mechanism is truly general. As discussed in Chapter 3, many recent algorithms attempt to improve the performance of RL. However, there is a big divide between researchers working on novel ML methods and those applying them to real-world robotic applications. Indeed, most algorithms are tested in benchmarks which are far from representing the complexities of a robotic system, making translational research a challenge.

There are many fundamental RoL questions that still do not have a good answer. Considering the structure of an MDP, there are several design choices that influence the learning problem: **i.** what should the state representation be? (e.g. RGB-D data or task-specific representations, how should other sensor information be combined?); **ii.** what should the action representation be? (e.g. instantaneous inputs or full trajectories, joint or task space, position or velocity control); **iii.** how can we define the reward? (e.g. sparse or dense, learned or engineered, how are multiple goals addressed?); **iv.** how should we sample the environment? (e.g. simulation or real-world, offline or online); and **v.** how should we model the policy? (e.g. value-based or policy-based, what ANN architecture should be used?). Finally, is it possible that even the currently available RL algorithms such as DDPG and PPO are able to solve complex robotic manipulation tasks, if we find the answers to all of the above?

As the aim was to tackle DOM problems, it was not feasible to attempt to answer all of the previous questions. Several state and reward formulations were investigated, which are specific to DLO tasks. Rewards based on Signed Distance Functions (SDFs) were also investigated, but is not included in this thesis. After the release of ReForm, there was an effort to make the explicit shape control tasks multi-goal, by randomly generating physically plausible target shapes. Furthermore, our collaboration project was awarded additional funding through the WASP Bridge initiative, that aimed at including the YuMi robot in the simulation environments for joint space control. While these efforts were initially planned as a continuation of the work in simulation, there was a decision to move toward real-world applications. While this was a great learning experience, working with real robots in a data driven approach was a time-consuming endeavor, which made progress quite slow.

Plans often change in research, as new knowledge is acquired and collaborations are formed. The future work section in the Licentiate thesis [1], is great evidence of that:

"Based on observations made during the initial phase of this graduate project, the aim of upcoming work will be twofold: ***i.*** *transfer policies learned in simulation to the real world and* ***ii.*** *attempt to reduce sample inefficiency by exploring model-based methods."*

While sim-to-real was explored, offline RL was the main research direction chosen in the end. Furthermore, no work was carried out on model-based methods as on several occasions it has been made clear that they are exceptionally difficult to make work. This knowledge did not come from a specific publication, but rather expert lectures during RL Summer Schools organized by Inria and CIFAR. The general idea seems to be that model-based RL is nice in theory but not yet in practice. In the following section, we look to the future again, with full conscience that it may never come to be.

5.2 Future Work

Using currently available hardware solutions, there are several DOM tasks that could be automated with the help of AI methods. An important job for future research would be to identify such tasks and attempt to solve them in all their complexities, not just in a simplified version. The cable-routing task presented in Section 2.6 is a great example of this. While most research has focused on stripped-down versions of this problem, it would be interesting to progressively move towards a complete cable harness production solution, e.g. the task boards for the assembly performance tests proposed by NIST, as part an effort to standardize *Robotic Hand Performance Testing* [144].

While this thesis lays some of the groundwork to accomplish fully autonomous cable-routing, the next steps are to combine the principles of Papers D and E with RoL methods. Recently, Luo et al. [76] applied RoL for cable-routing to learn both a high-level planner and low-level vision-based motion primitives, through an end-to-end hierarchical IL approach. In contrast, we aim to use both the vision- and FT-based motion primitives proposed in this thesis, which would be executed following a heuristic rule-based plan. No-

tably, Luo et al. [76] modeled the low-level primitives as MLPs, mapping raw camera feeds to a 4D Cartesian twist. Instead, our vision is to keep predefined primitives, but replace the GA replanning strategy in Paper D with an RL approach. Furthermore, in Paper E the observed failures in tensioning primitives were caused by poor estimation of the DLO elasticity parameter, while in Paper D the main source of failure was the DLO tracking algorithm, which caused unsuccessful grasps. Thus, replacing or enhancing these methods with SL could further help mitigate the DLO state estimation problem.

Healthcare Robotics Although this thesis has focused on domain-agnostic tasks, like shape-servoing and cable-routing, a key motivation to dive into the field of DOM was its potential for healthcare robotics. A recent review by Wang and Zhu [145] highlights many such applications in caregiving scenarios. Moreover, the RoL knowledge acquired during this doctoral project can be applied to many other problems from Biomedical Engineering. Some of the work not included in this thesis has been carried out exactly in such a biomedical context, namely bionic limb control via myoelectric signals. Future research directions will likely continue moving towards healthcare areas, like robotic rehabilitation.

References

[1] R. Laezza, "Robot learning for manipulation of deformable linear objects," Licentiate thesis, Chalmers University of Technology (Sweden), Oct. 2021.

[2] P. Heisler, P. Steinmetz, I. S. Yoo, and J. Franke, "Automatization of the cable-routing-process within the automated production of wiring systems," in *Energy Efficiency in Strategy of Sustainable Production III*, ser. Applied Mechanics and Materials, vol. 871, Trans Tech Publications Ltd, Nov. 2017, pp. 186–192.

[3] J. Zhu, A. Cherubini, C. Dune, *et al.*, "Challenges and outlook in robotic manipulation of deformable objects," *IEEE Robotics & Automation Magazine*, vol. 29, no. 3, pp. 67–77, 2022.

[4] J. Sanchez, J.-A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: A survey," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 688–716, 2018.

[5] J. Peters, R. Tedrake, N. Roy, and J. Morimoto, "Robot learning," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2010, pp. 865–869, ISBN: 978-0-387-30164-8.

[6] J. H. Connell and S. Mahadevan, "Introduction to robot learning," in *Robot Learning*, Springer, 1993, pp. 1–17.

[7] A. Doumanoglou, J. Stria, G. Peleka, *et al.*, "Folding clothes autonomously: A complete pipeline," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1461–1478, 2016.

[8] M. J. Thuy-Hong-Loan Le, A. Landini, M. Zoppi, D. Zlatanov, and R. Molfino, "On the development of a specialized flexible gripper for garment handling," *Journal of Automation and Control Engineering Vol*, vol. 1, no. 3, 2013.

[9] Y. She, S. Wang, S. Dong, N. Sunil, A. Rodriguez, and E. H. Adelson, "Cable manipulation with a tactile-reactive gripper," in *Robotics: Science and Systems (RSS)*, 2020.

[10] S. Duenser, R. Poranne, B. Thomaszewski, and S. Coros, "Robocut: Hot-wire cutting with robot-controlled flexible rods," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 98–1, 2020.

[11] T. Tang, C. Wang, and M. Tomizuka, "A framework for manipulating deformable linear objects by coherent point drift," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3426–3433, 2018.

[12] J. Grannen, P. Sundaresan, B. Thananjeyan, *et al.*, "Untangling dense knots by learning task-relevant keypoints," in *Conference on Robot Learning (CoRL)*, 2020.

[13] T. Tang and M. Tomizuka, "Track deformable objects from point clouds with structure preserved registration," *The International Journal of Robotics Research*, vol. 41, no. 6, pp. 599–614, 2022.

[14] S. Sen, A. Garg, D. V. Gealy, S. McKinley, Y. Jen, and K. Goldberg, "Automating multi-throw multilateral surgical suturing with a mechanical needle guide and sequential convex optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 4178–4185.

[15] M. Yan, Y. Zhu, N. Jin, and J. Bohg, "Self-supervised learning of state estimation for manipulating deformable linear objects," *IEEE Robotics and Automation Letters*, 2020.

[16] H. Wakamatsu and S. Hirai, "Static modeling of linear object deformation based on differential geometry," *The International Journal of Robotics Research*, vol. 23, no. 3, pp. 293–311, 2004.

[17] D. Berenson, "Manipulation of deformable objects without modeling and simulating deformation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2013, pp. 4525–4532.

[18] V. E. Arriola-Rios, P. Guler, F. Ficuciello, D. Kragic, B. Siciliano, and J. L. Wyatt, "Modeling of deformable objects for robotic manipulation: A tutorial and review," *Frontiers in Robotics and AI*, vol. 7, p. 82, 2020.

[19] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, "Learning predictive representations for deformable objects using contrastive estimation," in *Conference on Robot Learning*, PMLR, 2021, pp. 564–574.

[20] A. Zea, F. Faion, and U. D. Hanebeck, "Tracking elongated extended objects using splines," in *2016 19th International Conference on Information Fusion (FUSION)*, IEEE, 2016, pp. 612–619.

[21] M. Wnuk, C. Hinze, A. Lechler, and A. Verl, "Kinematic multibody model generation of deformable linear objects from point clouds," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 9545–9552.

[22] H. Yin, A. Varava, and D. Kragic, "Modeling, learning, perception, and control methods for deformable object manipulation," *Science Robotics*, vol. 6, no. 54, 2021.

[23] P. Sundaresan, J. Grannen, B. Thananjeyan, *et al.*, "Learning rope manipulation policies using dense object descriptors trained on synthetic depth data," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 9411–9418.

[24] T. Igarashi, T. Moscovich, and J. F. Hughes, "As-rigid-as-possible shape manipulation," *ACM transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 1134–1141, 2005.

[25] O. Sorkine and M. Alexa, "As-rigid-as-possible surface modeling," in *Symposium on Geometry processing*, Citeseer, vol. 4, 2007, pp. 109–116.

[26] M. Shetab-Bushehri, M. Aranda, Y. Mezouar, and E. Özgür, "Lattice-based shape tracking and servoing of elastic objects," *IEEE Transactions on Robotics*, 2023.

[27] A. C. Ugural and S. K. Fenster, *Advanced mechanics of materials and applied elasticity*. Pearson Education, 2011.

[28] N. Lv, J. Liu, X. Ding, J. Liu, H. Lin, and J. Ma, "Physically based real-time interactive assembly simulation of cable harness," *Journal of Manufacturing Systems*, vol. 43, pp. 385–399, 2017.

[29] V. Fontanari, M. Benedetti, and B. D. Monelli, "Elasto-plastic behavior of a warrington-seale rope: Experimental analysis and finite element modeling," *Engineering Structures*, vol. 82, pp. 113–120, 2015.

[30] A. Remde and D. Henrich, "Direct and inverse simulation of deformable linear objects," in *Robot manipulation of deformable objects*, Springer, 2000, pp. 43–70.

[31] H. Wakamatsu, K. Takahashi, and S. Hirai, "Dynamic modeling of linear object deformation based on differential geometry coordinates," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2005, pp. 1028–1033.

[32] H. Wakamatsu, T. Yamasaki, A. Tsumaya, E. Arai, and S. Hirai, "Dynamic modeling of linear object deformation considering contact with obstacles," in *2006 9th International Conference on Control, Automation, Robotics and Vision*, IEEE, 2006, pp. 1–6.

[33] J. M. L. Kraige, *Engineering Mechanics Statics*. John Wiley & Sons, In, 2000.

[34] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sendai, Japan, Sep. 2004, pp. 2149–2154.

[35] E. Rohmer, S. P. N. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 1321–1326.

[36] E. Coumans and Y. Bai, *Pybullet, a python module for physics simulation for games, robotics and machine learning*, `http://pybullet.org`, 2016.

[37] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 5026–5033.

[38] E. Coumans, *Bullet 2.38 physics sdk manual*, Bullet Physics Library, 2021.

[39] E. Todorov, *Mujoco: Multi-joint dynamics with contact*, 2024.

[40] F. Faure, C. Duriez, H. Delingette, *et al.*, "SOFA: A Multi-Model Framework for Interactive Physical Simulation," in *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, Y. Payan, Ed., vol. 11, Springer, Jun. 2012, pp. 283–321.

[41] C. Duriez, "Control of elastic soft robots based on real-time finite element method," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2013, pp. 3982–3987.

[42] Algoryx, *Agx dynamics user manual*, Umeå, Sweden, 2021.

[43] X. Lin, Y. Wang, J. Olkin, and D. Held, "Softgym: Benchmarking deep reinforcement learning for deformable object manipulation," in *Conference on Robot Learning (CoRL)*, 2020.

[44] F. Xiang, Y. Qin, K. Mo, *et al.*, "Sapien: A simulated part-based interactive environment," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 097–11 107.

[45] C. Gan, J. Schwartz, S. Alter, *et al.*, "ThreeDWorld: A platform for interactive multi-modal physical simulation," 2021.

[46] V. Makoviychuk, L. Wawrzyniak, Y. Guo, *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[47] M. Gazzola, L. Dudte, A. McCormick, and L. Mahadevan, "Forward and inverse problems in the mechanics of soft filaments," *Royal Society open science*, vol. 5, no. 6, p. 171 628, 2018.

[48] W. Yuan, S. Dong, and E. H. Adelson, "Gelsight: High-resolution robot tactile sensors for estimating geometry and force," *Sensors*, vol. 17, no. 12, p. 2762, 2017.

[49] J. Matas, S. James, and A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," in *Conference on Robot Learning (CoRL)*, 2018, pp. 734–743.

[50] A. Keipour, M. Bandari, and S. Schaal, "Deformable one-dimensional object detection for routing and manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4329–4336, 2022.

[51] K. Shivakumar, V. Viswanath, A. Gu, *et al.*, "Sgtm 2.0: Autonomously untangling long cables using interactive perception," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 5837–5843.

[52] A. X. Lee, H. Lu, A. Gupta, S. Levine, and P. Abbeel, "Learning force-based manipulation of deformable objects from multiple demonstrations," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 177–184.

[53] J. Sanchez, C. M. Mateo, J. A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, "Online shape estimation based on tactile sensing and deformation modeling for robot manipulation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 504–511.

[54] J. Sanchez, K. Mohy El Dine, J. A. Corrales, B.-C. Bouzgarrou, and Y. Mezouar, "Blind manipulation of deformable objects based on force sensing and finite element modeling," *Frontiers in Robotics and AI*, vol. 7, p. 73, 2020.

[55] Z. Kappassov, J.-A. Corrales, and V. Perdereau, "Tactile sensing in dexterous robot hands," *Robotics and Autonomous Systems*, vol. 74, pp. 195–220, 2015.

[56] J. M. Romano, K. Hsiao, G. Niemeyer, S. Chitta, and K. J. Kuchenbecker, "Human-inspired robotic grasp control with tactile sensing," *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1067–1079, 2011.

[57] A. Drimus, G. Kootstra, A. Bilberg, and D. Kragic, "Classification of rigid and deformable objects using a novel tactile sensor," in *2011 15th International Conference on Advanced Robotics (ICAR)*, IEEE, 2011, pp. 427–434.

[58] A. Monguzzi, M. Pelosi, A. M. Zanchettin, and P. Rocco, "Tactile based robotic skills for cable routing operations," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 3793–3799.

[59] G. Palli and S. Pirozzi, "A tactile-based wire manipulation system for manufacturing applications," *Robotics*, vol. 8, no. 2, p. 46, 2019.

[60] J. A. Fishel and G. E. Loeb, "Sensing tactile microvibrations with the biotac—comparison with human sensitivity," in *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, IEEE, 2012, pp. 1122–1127.

[61] S. Cui, R. Wang, J. Wei, F. Li, and S. Wang, "Grasp state assessment of deformable objects using visual-tactile fusion perception," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 538–544.

[62] S. LaValle, "Planning algorithms," *Cambridge University Press*, vol. 2, pp. 3671–3678, 2006.

[63] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[64] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming:: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[65] I. Mitsioni, Y. Karayiannidis, and D. Kragic, "Modelling and learning dynamics for robotic food-cutting," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, IEEE, 2021, pp. 1194–1200.

[66] A. Caporali, P. Kicki, K. Galassi, R. Zanella, K. Walas, and G. Palli, "Deformable linear objects manipulation with online model parameters estimation," *IEEE Robotics and Automation Letters*, 2024.

[67] J. Zhu, B. Navarro, P. Fraisse, A. Crosnier, and A. Cherubini, "Dual-arm robotic manipulation of flexible cables," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 479–484.

[68] M. Ruan, D. Mc Conachie, and D. Berenson, "Accounting for directional rigidity and constraints in control for manipulation of deformable objects without physical simulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 512–519.

[69] R. Lagneau, A. Krupa, and M. Marchal, "Automatic shape control of deformable wires based on model-free visual servoing," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5252–5259, 2020.

[70] J. Smolen and A. Patriciu, "Deformation planning for robotic soft tissue manipulation," in *2009 Second International Conferences on Advances in Computer-Human Interactions*, IEEE, 2009, pp. 199–204.

[71] M. Yu, H. Zhong, F. Zhong, and X. Li, "Adaptive control for robotic manipulation of deformable linear objects with offline and online learning of unknown models," *arXiv preprint arXiv:2107.00194*, 2021.

[72] K. Almaghout, A. Cherubini, and A. Klimchik, "Robotic co-manipulation of deformable linear objects for large deformation tasks," *Robotics and Autonomous Systems*, p. 104 652, 2024.

[73] J. Zhu, B. Navarro, R. Passama, P. Fraisse, A. Crosnier, and A. Cherubini, "Robotic manipulation planning for shaping deformable linear objects with environmental contacts," *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 16–23, 2019.

[74] K. Galassi and G. Palli, "Robotic wires manipulation for switchgear cabling and wiring harness manufacturing," in *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*, IEEE, 2021, pp. 531–536.

[75] S. Jin, W. Lian, C. Wang, M. Tomizuka, and S. Schaal, "Robotic cable routing with spatial representation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5687–5694, 2022.

[76] J. Luo, C. Xu, X. Geng, *et al.*, "Multi-stage cable routing through hierarchical imitation learning," *IEEE Transactions on Robotics*, 2024.

[77] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[78] D. Silver, T. Hubert, J. Schrittwieser, *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[79] G. Tesauro, "Td-gammon, a self-teaching backgammon program, achieves master-level play," *Neural computation*, vol. 6, no. 2, pp. 215–219, 1994.

[80] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[81] O. Vinyals, I. Babuschkin, W. M. Czarnecki, *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[82] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[83] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[84] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[85] H. Nguyen and H. La, "Review of deep reinforcement learning for robot manipulation," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, IEEE, 2019, pp. 590–595.

[86] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel, "Learning to Manipulate Deformable Objects without Demonstrations," in *Robotics: Science and Systems (RSS)*, Corvalis, Oregon, USA, Jul. 2020.

[87] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, and A. M. Agogino, "Deep reinforcement learning for robotic assembly of mixed deformable and rigid objects," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 2062–2069.

[88] M. Vecerik, T. Hester, J. Scholz, *et al.*, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[89] B. Thananjeyan, A. Garg, S. Krishnan, C. Chen, L. Miller, and K. Goldberg, "Multilateral surgical pattern cutting in 2d orthotropic gauze with deep reinforcement learning policies for tensioning," in *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 2371–2378.

[90] C. Szepesvári, "Algorithms for reinforcement learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 4, no. 1, pp. 1–103, 2010.

[91] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[92] D. Bertsekas, *Reinforcement and Optimal Control*. Athena Scientific, 2019.

[93] K. Doya, "Reinforcement learning in continuous time and space," *Neural computation*, vol. 12, no. 1, pp. 219–245, 2000.

[94] T. Wang, X. Bao, I. Clavera, *et al.*, "Benchmarking model-based reinforcement learning," *arXiv preprint arXiv:1907.02057*, 2019.

[95] M. Liu, M. Zhu, and W. Zhang, "Goal-conditioned reinforcement learning: Problems and solutions," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, L. D. Raedt, Ed., Survey Track, International Joint Conferences on Artificial Intelligence Organization, Jul. 2022, pp. 5502–5511.

[96] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[97] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2006, pp. 2219–2225.

[98] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *ICML*, 2014.

[99] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

[100] J. Peters, "Policy gradient methods," *Scholarpedia*, vol. 5, no. 11, p. 3698, 2010.

[101] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[102] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[103] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[104] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning (ICML)*, PMLR, 2015, pp. 448–456.

[105] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[106] H. van Hasselt, A. Guez, M. Hessel, V. Mnih, and D. Silver, "Learning values across many orders of magnitude," *Advances in Neural Information Processing Systems*, vol. 29, pp. 4287–4295, 2016.

[107] H. van Hasselt, Y. Doron, F. Strub, M. Hessel, N. Sonnerat, and J. Modayil, "Deep reinforcement learning and the deadly triad," *arXiv preprint arXiv:1812.02648*, 2018.

[108] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[109] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning (ICML)*, PMLR, 2018, pp. 1587–1596.

[110] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[111] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning (ICML)*, PMLR, 2016, pp. 1329–1338.

[112] S. Fujimoto, W.-D. Chang, E. J. Smith, S. S. Gu, D. Precup, and D. Meger, "For sale: State-action representation learning for deep reinforcement learning," *arXiv preprint arXiv:2306.02451*, 2023.

[113] Z. Wang, V. Bapst, N. Heess, *et al.*, "Sample efficient actor-critic with experience replay," *arXiv preprint arXiv:1611.01224*, 2016.

[114] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning (ICML)*, PMLR, 2018, pp. 1861–1870.

[115] V. Mnih, A. P. Badia, M. Mirza, *et al.*, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning (ICML)*, PMLR, 2016, pp. 1928–1937.

[116] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning (ICML)*, PMLR, 2015, pp. 1889–1897.

[117] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[118] A. Stooke and P. Abbeel, "Rlpyt: A research code base for deep reinforcement learning in pytorch," *arXiv preprint arXiv:1909.01500*, 2019.

[119] T. Seno and M. Imai, "D3rlpy: An offline deep reinforcement learning library," *Journal of Machine Learning Research*, vol. 23, no. 315, pp. 1–20, 2022.

[120] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[121] M. Andrychowicz, F. Wolski, A. Ray, *et al.*, "Hindsight experience replay," in *31st International Conference on Neural Information Processing Systems (NIPS)*, 2017, pp. 5055–5065.

[122] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[123] S. Parisi, V. Tangkaratt, J. Peters, and M. E. Khan, "Td-regularized actor-critic methods," *Machine Learning*, vol. 108, no. 8, pp. 1467–1501, 2019.

[124] K. W. Cobbe, J. Hilton, O. Klimov, and J. Schulman, "Phasic policy gradient," in *International Conference on Machine Learning (ICML)*, PMLR, 2021, pp. 2020–2027.

[125] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[126] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[127] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *arXiv preprint arXiv:2110.06169*, 2021.

[128] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," *Advances in neural information processing systems*, vol. 34, pp. 20 132–20 145, 2021.

[129] G. Brockman, V. Cheung, L. Pettersson, *et al.*, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[130] M. Towers, J. K. Terry, A. Kwiatkowski, *et al.*, *Gymnasium*, Mar. 2023.

[131] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, "Fetch and freight: Standard platforms for service robot applications," in *Workshop on autonomous mobile service robots*, 2016.

[132] S. Robotics, "Shadow dexterous hand technical specifications," *Shadow Robot Company, London. www.shadowrobot.com*, 2013.

[133] M. Lucchi, F. Zindler, S. Mühlbacher-Karrer, and H. Pichler, "Robogym–an open source toolkit for distributed deep reinforcement learning on real and simulated robots," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[134] Z. Huang, Y. Hu, T. Du, *et al.*, "Plasticinelab: A soft-body manipulation benchmark with differentiable physics," in *International Conference on Learning Representations (ICLR)*, 2021.

[135] Y. Hu, T.-M. Li, L. Anderson, J. Ragan-Kelley, and F. Durand, "Taichi: A language for high-performance computation on spatially sparse data structures," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–16, 2019.

[136] T. Sugiyama, N. Schweighofer, and J. Izawa, "Reinforcement learning establishes a minimal metacognitive process to monitor and control motor learning performance," *Nature Communications*, vol. 14, no. 1, p. 3988, 2023.

[137] C. Heyes and C. Catmur, "What happened to mirror neurons?" *Perspectives on Psychological Science*, vol. 17, no. 1, pp. 153–168, 2022.

[138] P. Vassiliadis, G. Derosiere, C. Dubuc, *et al.*, "Reward boosts reinforcement-based motor learning," *Iscience*, vol. 24, no. 7, 2021.

[139] B. Blain and T. Sharot, "Intrinsic reward: Potential cognitive and neural mechanisms," *Current Opinion in Behavioral Sciences*, vol. 39, pp. 113–118, 2021.

[140] N. Savage, "Bioinspired robots walk, swim, slither and fly.," *Nature*, 2022.

[141] I. Akkaya, M. Andrychowicz, M. Chociej, *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[142] W. Wang, Y. Tang, and C. Li, "Controlling bending deformation of a shape memory alloy-based soft planar gripper to grip deformable objects," *International Journal of Mechanical Sciences*, vol. 193, p. 106 181, 2021.

[143] H. Moravec, *Mind children: The future of robot and human intelligence*. Harvard University Press, 1988.

[144] *Assembly performance metrics and test methods*, `https://www.nist.gov/el/intelligent-systems-division-73500/robotic-grasping-and-manipulation-assembly/assembly`, National Institute of Standards and Technology (NIST) - United States Department of Commerce.

[145] L. Wang and J. Zhu, "Deformable object manipulation in caregiving scenarios: A review," *Machines*, vol. 11, no. 11, p. 1013, 2023.