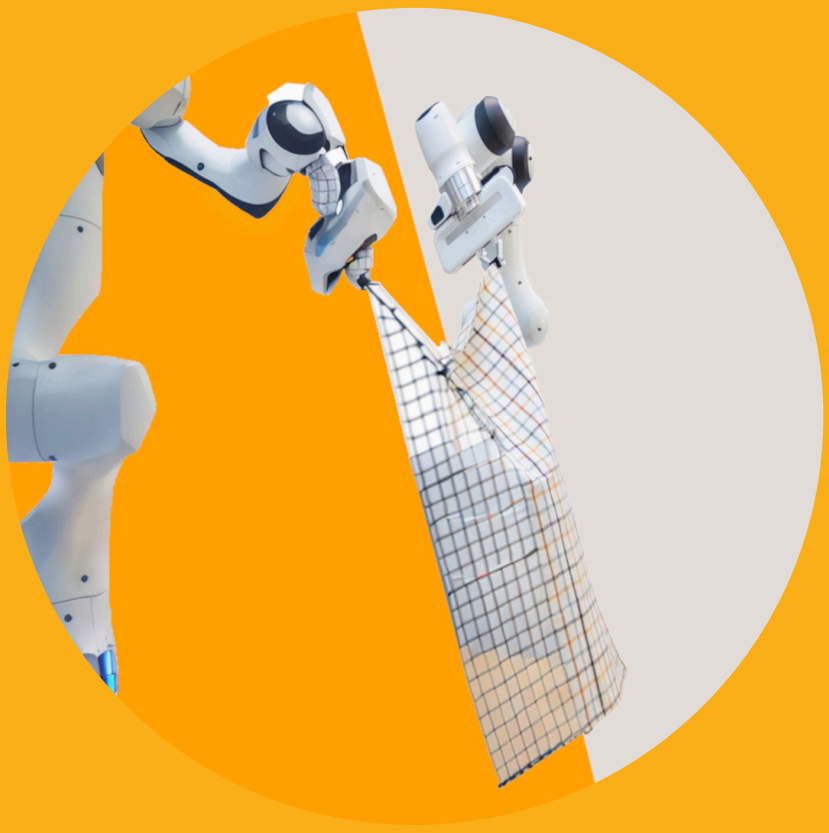
Department of Electrical Engineering and Automation

Towards Efficient Robotic Manipulation of Deformable Objects by Learning Dynamics Models and Adaptive Policies

David Blanco-Mulero



Aalto University

DOCTORAL THESES

Aalto University publication series
DOCTORAL THESES 64/2024

Towards Efficient Robotic Manipulation of Deformable Objects by Learning Dynamics Models and Adaptive Policies

David Blanco-Mulero

A doctoral thesis completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Electrical Engineering, at a public examination held at the lecture hall AS2 of the school on 19th April 2024 at 12:00.

Aalto University
School of Electrical Engineering
Department of Electrical Engineering and Automation
Intelligent Robotics

Supervising professor
Professor Ville Kyrki, Aalto University, Finland

Thesis advisor
Assistant Professor Gokhan Alcan, Tampere University, Finland

Preliminary examiners
Assistant Professor Daniel Seita, University of Southern California, USA
Professor Francis Wyffels, University of Ghent, Belgium

Opponent
Professor Yiannis Demiris, Imperial College London, United Kingdom

Aalto University publication series
DOCTORAL THESES 64/2024

ISBN 978-952-64-1737-0 (printed)
ISBN 978-952-64-1738-7 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-64-1738-7

Unigrafia Oy
Helsinki 2024

Finland

Printed matter
4041-0619

Author
David Blanco-Mulero

Name of the doctoral thesis
Towards Efficient Robotic Manipulation of Deformable Objects by Learning Dynamics Models and Adaptive Policies

Publisher School of Electrical Engineering

Unit Department of Electrical Engineering and Automation

Series Aalto University publication series DOCTORAL THESES 64/2024

Field of research Automation, Systems and Control Engineering

Manuscript submitted 30 November 2023 **Date of the defence** 19 April 2024

Permission for public defence granted (date) 1 March 2024 **Language** English

☐ **Monograph** ☒ **Article thesis** ☐ **Essay thesis**

Abstract

Recent years have witnessed significant progress in developing intelligent robotic systems that are able to perform manipulation tasks. One reason for this success has been the advent of learning-based approaches, which driven by improvements in deep learning techniques, have endowed robots with greater generalisation capabilities to manipulate objects varying in shape, size, and texture. However, the majority of these accomplishments have been restricted to the domain of rigid objects, while our world is replete with diverse objects that deform when manipulated. This introduces a new set of challenges, such as the need for representing their deformation and adapting the robotic manipulation actions accordingly. Nevertheless, attempts have been made to improve the efficiency of current approaches by either reducing the number of interactions required to succeed in these tasks or reducing the amount of data collected in the real world using simulation engines. Although methods have been proposed for learning to manipulate deformable objects such as garments, their adaptation capabilities still remain limited.

Therefore, this dissertation proposes methods to bridge the gap in the adaptive capabilities of robotic systems for manipulating a variety of materials and objects. More specifically, it investigates methods that can learn to efficiently manipulate deformable objects in simulation, transfer the learnt skills to the real world, and examine the challenges that arise when transferring these skills. To accomplish this, the thesis first investigates the representation and modelling of deformable object dynamics using data-driven approaches, resulting in two methods for modelling the dynamics using graph-based representations. Subsequently, the thesis continues by investigating methods for enabling the learning of policies that can adapt and generalise to different objects and material properties. Thus, the dissertation proposes two approaches: adapting manipulation primitives when performing high-level planning and implementing closed-loop feedback for adapting the actions according to the object's deformation. Finally, this thesis studies a major challenge limiting approaches that learn to manipulate deformable objects in simulation: the reality gap. Here, a benchmark data set is proposed to evaluate the gap when performing a dynamic manipulation task.

The results of the work comprising this dissertation show that policies learnt in simulation can adapt to a wide variety of deformable objects and can efficiently manipulate them, where closed-loop feedback can mitigate the reality gap in these approaches. Consequently, approaches based on learning in simulation can enhance the adaptability of manipulation systems, where closed-loop feedback plays a vital role in successfully transferring the learnt skills to the real world.

Keywords Robotics, Machine Learning, Deformable Object Manipulation

ISBN (printed) 978-952-64-1737-0 **ISBN (pdf)** 978-952-64-1738-7

ISSN (printed) 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Helsinki **Location of printing** Helsinki **Year** 2024

Pages 126 **urn** http://urn.fi/URN:ISBN:978-952-64-1738-7

Preface

After four years of navigating the hills of the PhD process, amidst lakes, snow, robots, and great people, here I am, concluding this journey. In these lines, I would like to thank all of those that were involved and helped me in this process.

First, I want to thank the Academy of Finland and Business Finland. Their financial support provided me the chance to do what I love the most, which is conducting research, being creative, and solving problems with robots. I would also like to thank the Aalto Foundation, who funded my research visit and helped me meet the people I can now call colleagues. Moreover, I want to thank the Nokia Foundation for their encouragement grant, and the Aalto Science IT project, whose computational resources enabled most of the works carried out through my dissertation.

This journey was made possible thanks to two researchers that I highly esteem, to whom I am grateful to refer as my supervisor and advisor for my doctoral thesis. First, I would like to express my sincere gratitude to Professor Ville Kyrki, my supervisor, for his invaluable guidance and support during this endeavour. He helped me become a better researcher, encouraged me to be creative, and pursue innovative ideas. Even when I doubted myself, he helped me to be critical and managed to keep me motivated. Then, I would like to thank my advisor, Professor Gokhan Alcan. I am grateful for our constant discussions of research ideas, restless hours working on the tiniest detail, and all the nitpicking. I cannot find words to describe how much I appreciate all the support both of you have given me. With all my heart, thank you.

I want to thank my pre-examiners, Professor Daniel Seita and Professor Francis Wyffels, whose comments helped me refine my dissertation and also gave me new insights on my work. I would like to express my gratitude to all the co-authors with whom I had the pleasure to collaborate: Markus Heinonen, whose assistance was invaluable and helped me understand better Gaussian Processes; to Julius Hietala and Neea Tuomainen, who dedicated significant time and effort to their Master's thesis and contributed to my research; to Fares J. Abu-Dakka, for all the support for improving QDP; to Oriol Barbany and Adrià Colomé, for all the dedication to make the benchmarking possible; and finally to Professor Carme Torras, who on top of the contribution to the benchmarking

article, gave me the huge opportunity to carry out a research visit at the Institut de Robòtica i Informàtica Industrial perception and manipulation's group.

Now I would like to thank all my colleagues at Aalto University, who made every day of this journey special. To my Reinforcement Learning course buddies, Oliver and Karol, thank you for those extensive hours grading and discussing the exercises, but also for all the laughing you brought on those days. To all the members of the group, past and present, and in particular to Mattia, Kevin, Shivam, and Matti, for all the work and life conversations. To Ricardo, even though your time in the group was short, I still miss our short breaks. To Karol, for all the discussions on Reinforcement Learning and coding, as well as all your help without asking anything in return, you are an amazing person. To Oliver, for all the chats on the ups and downs throughout our PhD journey, we made it. To Joukko, I am very grateful to have shared this journey with you. To my deformable objects manipulation buddy Tran, I really appreciate all the discussions and the time we have enjoyed both in Kyoto and Detroit, with more exciting times ahead. And finally, to my office mate and climbing buddy, Francesco. Not only did you help me with details about research, presentations, or figures, but also supported me and cared for my mental well-being. I cannot thank you enough and I am grateful to call you a friend.

All this was only possible thanks to the people I hold dearest. To my brother Santi, my mother Carmen, and my uncle Jose, I only got here because you always believed in me. To Carlos and Lucero, whom I can call my second parents and cannot thank enough for their support. To Pablo, Marcos, Ruben, Kike, Mario, and Cristian. Even though the *pozo* has been there throughout our journey together, we have always managed to get through. Our toxic relationship cannot be better, thanks for always being there when I need you. To all the people that I am not mentioning but have been there as well. Despite all the distance, you have all been with me. This work is also yours.

Finally, to my grandma. Abue, aunque he estado lejos, siempre has estado conmigo en este camino, te quiero.

Barcelona, March 4, 2024,

David Blanco-Mulero

Contents

List of Publications

This thesis consists of an overview and the following publications, which are referred to in the text by their Roman numerals..

I **David Blanco-Mulero**, Markus Heinonen and Ville Kyrki. Evolving-Graph Gaussian Processes. In *International Conference on Machine Learning, Time Series Workshop*, Virtual Workshop., July 24 2021.

II Neea Tuomainen*, **David Blanco-Mulero*** and Ville Kyrki. Manipulation of Granular Materials by Learning Particle Interactions. *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, issue 2, pp. 5663-5670, April 2022.

III **David Blanco-Mulero**, Gokhan Alcan, Fares J. Abu-Dakka and Ville Kyrki. QDP: Learning to Sequentially Optimise Quasi-Static and Dynamic Manipulation Primitives for Robotic Cloth Manipulation. Accepted for publication in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Detroit, US., October 2023.

IV Julius Hietala, **David Blanco-Mulero**, Gokhan Alcan and Ville Kyrki. Learning Visual Feedback Control for Dynamic Cloth Folding. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Kyoto, Japan. pp. 1455-1462, October 2022.

V **David Blanco-Mulero**, Oriol Barbany, Gokhan Alcan, Adrià Colomé, Carme Torras, Ville Kyrki. Benchmarking the Sim-to-Real Gap in Cloth Manipulation. Accepted for publication in *IEEE Robotics and Automation Letters (RA-L)*, vol. 9, issue 3, pp. 2981-2988, March 2024.

Author's Contribution

Publication I: "Evolving-Graph Gaussian Processes"

David Blanco-Mulero devised the method together with Markus Heinonen and Ville Kyrki. David Blanco-Mulero was responsible for the implementation of the proposed algorithm, performed the experiments and wrote the publication.

Publication II: "Manipulation of Granular Materials by Learning Particle Interactions"

This publication is an extension of the Master's thesis by the first author, Neea Tuomainen. David Blanco-Mulero served as the instructor for Neea Tuomainen's thesis. The methodology and the experiments were planned by all authors. Neea Tuomainen was responsible for the implementation and testing the data-driven dynamics model, under the regular guidance of David Blanco-Mulero. David Blanco-Mulero was responsible for the control approach and the real-world experiments. Both Neea Tuomainen and David Blanco-Mulero were co-first authors of the paper and were responsible for writing the publication.

Publication III: "QDP: Learning to Sequentially Optimise Quasi-Static and Dynamic Manipulation Primitives for Robotic Cloth Manipulation"

David Blanco-Mulero devised the method together with Gokhan Alcan and Ville Kyrki. Gokhan Alcan was responsible for implementing the dynamic manipulation primitive. Fares J. Abu-Dakka was responsible for implementing the robot control. David Blanco-Mulero was responsible for implementing the proposed algorithm, performing the experiments and writing the publication.

Publication IV: "Learning Visual Feedback Control for Dynamic Cloth Folding"

This publication is an extension of the Master's thesis by the first author, Julius Hietala. David Blanco-Mulero and Gokhan Alcan served as the instructors for Julius Hietala's thesis. The methodology and experiments were planned by all authors. Julius Hietala was responsible for the implementation and testing of the approach, under the regular guidance of David Blanco-Mulero and Gokhan Alcan. Julius Hietala, David Blanco-Mulero and Gokhan Alcan were responsible for writing the publication.

Publication V: "Benchmarking the Sim-to-Real Gap in Cloth Manipulation"

This publication is the result of David Blanco-Muleros's research visit at the Institut de Robòtica i Informàtica Industrial, under the supervision of Adrià Colomé and professor Carme Torras. David Blanco-Mulero, Oriol Barbany, Gokhan Alcan and Adrià Colomé defined the research problem together. David Blanco-Mulero held the main responsibility for performing the experiments and collecting the benchmark data set. Oriol Barbany was responsible for the implementation of the vision and system identification algorithms. David Blanco-Mulero and Oriol Barbany together implemented the simulation environments. David Blanco-Mulero analysed the results and was responsible for writing the publication.

Language check

The language of my dissertation has been checked by Kenneth Pennington. I have personally examined and accepted/rejected the results of the language check one by one. This has not affected the scientific content of my dissertation.

List of Acronyms

BO Bayesian Optimisation

CD Chamfer Distance

CG Conjugate Gradient

CMA-ES Covariance Matrix Adaptation Evolution Strategy

CMA-ES Covariance Matrix Adaptation Evolution Strategy

DL Deep Learning

DoF Degree of Freedom

e-GGP evolving-Graph Gaussian Process

FEM Finite Element Method

GAN Generative Adversarial Network

GN Graph Network

GNN Graph Neural Network

GP Gaussian Process

HLP high-level planning

IL Imitation Learning

LLC low-level control

MPC Model Predictive Control

MPM Material Point Method

MSS mass-spring system

NN Neural Network

ODE Ordinary Differential Equation

P-n-P Pick-and-Place

PBD Position Based Dynamics

QDP Quasi-Dynamic Parameterisable

RL Reinforcement Learning

SAC Soft-Actor Critic

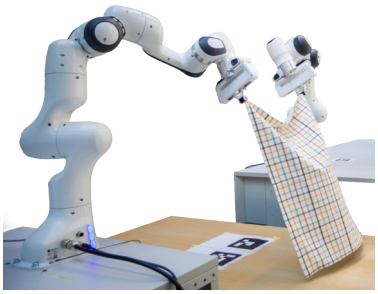
SDQN Sequential DQN

SPH Smoothed Particle Hydrodynamics

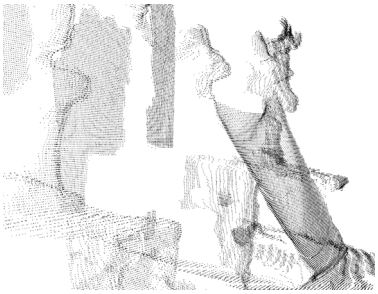
1. Introduction

Newborns, after the first month of birth, embark on their journey of learning to manipulate objects [1]. During consecutive months, humans gradually develop by trial-and-error different visuomotor skills [2], from grasping soft toys to building towers of blocks. Once they gain more control over their motor skills, toddlers start engaging in more complex tasks such as shaping plasticine. During this learning process, when faced with a manipulation task, humans use any prior information and *representation*, combined with visual input, to *adapt* their manipulation actions [3]. This enables humans to manipulate complex objects in the blink of an eye by refining their actions for tasks such as shaping and molding a lump of clay into a pottery piece. For example, humans can improve their performance in a task through sufficient practice of the task, as this allows them to optimise their skills to become as *efficient* as possible, finishing the task in a handful of trials [4]. But, what about robots? How can we endow robotic systems with the refined manipulation skills of a human?

In recent years, Deep Learning (DL) and Reinforcement Learning (RL) have enabled robotic systems to adapt and generalise some skills to unseen objects in tasks that require picking and placing different sets of objects. Despite recent advances in robotic manipulation, much of this success has been limited to the manipulation of rigid objects. Moreover, our world also contains objects that deform when manipulated, including flexible materials such as textiles, vegetables, or fruit, to name a few. When the manipulation of these objects involves taking into account its deformation, we refer to this type of manipulation as *precise* manipulation of deformable objects. For conferring robots with the skill of *precisely* manipulating deformable objects, robotic systems need to deal with challenges such as measuring the objects' deformation and adapting to deformation changes. In order to extend the capabilities of robots to precisely manipulate deformable objects, this dissertation studies techniques that adapt and manipulate these objects efficiently. Throughout the thesis, we will use the term *efficient* in the context of systems to refer to systems that are *task-efficient*; that is, they are capable of completing a task within a few trials. In the context of learning approaches, we will use the term *efficient* to refer to approaches that learn to perform a task without interacting with the real world by learning the



(a) Image representation.



(b) Point cloud representation.

Figure 1.1. Example of a dual robotic system manipulating a deformable object where the manipulation scene is depicted using two representations.

skills in simulation.

1.1 Background

Deformable objects present several challenges for robotic manipulation. Unlike rigid objects, the configuration of deformable objects cannot be solely represented by their position and orientation, as they have infinite Degrees of Freedom (DoFs). This representation problem becomes even more daunting when attempting to measure object deformation using a vision system (see Figure 1.1). Similar to rigid objects, occlusions lead to inaccurate measurements of the object configuration [5]. Some recent methods have been proposed to deal with occlusions by reconstructing the shape of the object and then fine-tuning its shape [6–8]. However, most methods consider the state to be partially observable, and the controller is able to perform the task, even though some information about the object remains unknown [9–13]. Throughout this dissertation, we assume that the state of the objects is fully known, provided by such means as a deformable object simulator; and that it is partially observable in the real world. Based on these assumptions, this dissertation focusses on methods that do not require complete knowledge of the object's configuration in the real world in order to *adapt* the manipulation actions of the robot to the object deformation. This deformation is crucial for precise deformable object manipulation, such as folding a piece of cloth [14], accurately grasping a strawberry [15], or planning precise manipulation of human tissue [16]. The tasks described in this dissertation are precise deformable object manipulation tasks, which differ from manipulation tasks where a deformable object is picked and displaced ignoring subtle changes to its deformation [17, 18].

Deformation of an object is a result of the complex dynamics of the object, thereby making manipulation a challenging task. Thus, to be able to represent the state of deformable objects and capture the dynamics of the system, researchers have envisioned a wide variety of methods to approximate their

dynamics for manipulation [19]. Initial work on deformable object manipulation focused on *analytical models* to approximate their dynamics [20–22]. As the computational power increased, more complex models have been proposed to simulate the dynamics of objects using techniques such as Finite Element Methods (FEMs) [23, 24] or Position Based Dynamics (PBD) [25, 26]. These simulators offer great benefits for *learning* to manipulate objects. First, they are inherently safe and *efficient*, as there is no interaction with the real systems. Second, they provide a huge amount of data for training *data-driven* approaches, which together with methods such as *domain randomisation*, can help generalise and *adapt* to different objects and materials. For this reason, the thesis uses data-driven approaches that are trained from data gathered in simulation engines to provide efficiency and adaptability.

With advances in DL, various research directions have emerged for planning manipulation actions using data-driven approaches. One option is to *learn* dynamics models using data from high-fidelity simulation engines, without the inherent computational burden required by these engines [27, 28], and then to leverage these models for planning [6, 11]. Another alternative is to *learn* a controller that, instead of using a dynamics model, utilises the experience gathered in simulation to decide on the next action. Nevertheless, learning in simulation and transferring the skills to the real-world might result in some unexpected outcomes. Often, the mathematical approximations of system dynamics in these simulators lead to inaccuracies, which, along with unmodelled phenomena, result in discrepancies from real world behaviour, also referred to as the *sim-to-real* gap or reality gap [29]. In this dissertation, we study both approaches that learn dynamics for planning and those that learn controllers, as well as the reality gap in deformable object manipulation.

Another challenge present in precise deformable object manipulation is that both the object position and shape are modified by the forces applied. This problem becomes apparent when performing *quasi-static manipulation* that neglect inertial forces. However, it becomes even more notable when performing *dynamic manipulation*, as manipulation inertial forces are crucial to succeed in tasks such as tossing an object [30, 31]. The use of dynamic manipulation for deformable object manipulation has recently gained increasing interest, as it has been shown to be more *task-efficient* than quasi-static manipulation by reducing the number of interactions required for tasks such as unfolding a piece of cloth [32, 33]. However, the state-of-the-art has been limited by the lack of methods for adapting manipulation actions. Methods that utilise quasi-static manipulation have mainly focused on using pre-defined manipulation primitives, such as Pick-and-Place (P-n-P), thus disregarding the effect of parameters such as the velocity used to execute the manipulation primitives. On the other hand, dynamic manipulation methods execute the actions in an open-loop manner, without taking into account the state of the object during the manipulation process. In order to move towards more generic robotic systems that can manipulate the wide variety of materials present in human environments, it is crucial to be

able to *adapt* both quasi-static and dynamic manipulation actions.

1.2 Research Aim and Questions

To address the aforementioned challenges in the modelling, perception, and manipulation of deformable objects, the objective of this thesis is to develop data-driven methods to efficiently model and precisely manipulate deformable objects. More specifically, the ultimate goal is to enhance the adaptive capabilities of robotic manipulation systems in order to enable them to cope with a wide variety of deformable objects in terms of their shape, colour, texture, and material properties. Such deformable objects include those commonly present in households or manufacturing industries.

To achieve these goals, the dissertation seeks to answer three Research Questions (RQs).

RQ 1 How can we *efficiently* learn to model the dynamics of deformable objects in order to manipulate them? (Chapter 2, Publication I and Publication II)

RQ 2 How can we learn robot control policies that can *adapt* the actions to deal with a variety of materials, shapes and sizes in precise deformable object manipulation? (Chapter 3, Publication III and Publication IV)

RQ 3 What is the impact of the *sim-to-real* gap on deformable object manipulation when learning manipulation policies entirely in simulation? (Chapter 4, Publication III, Publication IV and Publication V)

1.3 Contributions

The main contributions of this dissertation focus on efficiently learning dynamics models and task-efficient learning-based control of deformable objects, leading to five publications by the author. Initially, **RQ 1** was addressed by identifying suitable representations for deformable objects as well as proposing methods for learning their dynamics and interactions with rigid objects for manipulation. Subsequently, **RQ 2** was answered by developing methods for learning manipulation policies that could adapt their actions to different deformable object materials. Finally, **RQ 3** was addressed by evaluating the impact of the sim-to-real gap on the manipulation of deformable objects

The key scientific contributions of the dissertation are as follows:

- The evolving-Graph Gaussian Process (e-GGP): a novel autoregressive Gaussian Process model that learns the transition of vertices on the graph domain for predicting the graph evolution of physical dynamical systems. The performance of the model is demonstrated for learning the evolution of simulated deformable objects modeled as a graph.

- A data-driven approach for manipulating granular materials. The method represents the granular material as a graph, learning the dynamics and interactions with rigid objects using a Graph Neural Network (GNN). The manipulation trajectory is computed by minimising the Wasserstein distance between the distribution of the manipulated granular material and its target distribution.
- The *Quasi-Dynamic Parameterisable* approach (QDP): a novel Reinforcement Learning approach for learning a visual policy that can optimise the parameters of pre-defined manipulation primitives, including pick and place positions, as well as additional primitive parameters for cloth manipulation, to adapt manipulation actions.
- Analysing the performance of quasi-static and dynamic manipulation primitives in the real world on a public cloth unfolding benchmark using a single manipulation arm.
- A method for learning a closed-loop visual feedback policy to perform dynamic cloth manipulation. The method learns the policy entirely in simulation, given a single human demonstration on real fabrics, and can be directly transferred to the real world. Thus, the closed-loop visual feedback policy allows adaptation to a variety of fabrics.
- Implementing and releasing an open-source[1] benchmark for dynamic and quasi-static cloth manipulation on four well-established physics simulators: MuJoCo, Bullet, Flex, and SOFA.
- Open-source implementation of all the scientific contributions: e-GGP[2], the data-driven granular manipulation[3], QDP[4], the closed-loop visual feedback dynamic cloth manipulation[5], and the benchmarking of cloth manipulation in different simulation engines[1]. The open-source implementations also include multimedia material showcasing the real-world experiments for Publications II, III, IV and V.

1.4 Structure of the thesis

This dissertation is structured following the aforementioned research questions, dividing the chapters into three sub-problems: representation and dynamics modelling, adapting manipulation skills, and the sim-to-real gap.

[1] https://sites.google.com/view/cloth-sim2real-benchmark
[2] https://github.com/dblanm/evolving-ggp
[3] https://sites.google.com/view/granular-gnn-manipulation
[4] https://sites.google.com/view/qdp-srl
[5] https://sites.google.com/view/dynamic-cloth-folding

First, Chapter 2 discusses the representation and dynamics modelling of deformable objects. The chapter starts by depicting several options for this problem. Then, the chapter examines data-driven approaches to model the dynamics for manipulation tasks, as well as analyses their limitations.

Next, Chapter 3 addresses the problem of learning policies for manipulation through a literature review of current approaches and how they adapt robot actions to different object properties. After introducing the quasi-static and dynamic manipulation techniques that will be discussed throughout the thesis, the chapter examines approaches for precise the manipulation of deformable objects, dividing these into model-based and model-free methods. Later, the chapter discusses methods for learning visual feedback policies that can adapt to a variety of deformable object materials and shapes.

Chapter 4 presents the sim-to-real gap, one of the challenges when learning from simulation engines. The chapter also reviews techniques for quantifying and mitigating the sim-to-real gap in deformable object manipulation.

Finally, Chapter 5 ends the dissertation by highlighting the main contributions and suggesting future research directions.

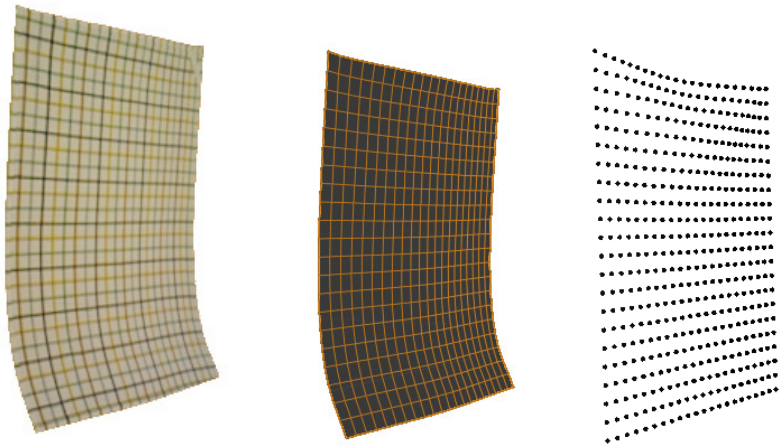
2. Representation and Learning of Deformable Objects Dynamics for Manipulation

When we manipulate a pottery mug, we use our mental image of it to apply the force necessary to hold it and move it. However, if the mug is still fresh and requires going through a kiln, it will deform when we apply enough strength. One of the major distinctions between manipulating these two objects is their dynamic behaviour. So, how do humans create a representation for each of these items? This representation is essential for understanding the deformation and dynamics of the object, enabling robots to perform manipulation tasks. Therefore, how can we represent deformable objects and model their dynamics to manipulate them with robotic systems?

This chapter discusses the problem of representation and learning of deformable object dynamics. First, it begins with an overview of options for representing the state of a deformable object. This is then followed by exploring several options for modelling their dynamics. The chapter continues by discussing prior work on learning dynamics models for performing robotic manipulation. These models have been divided into image-based and graph-based approaches. The chapter discusses how Publication I and Publication II have addressed some of the limitations described in the literature, closing with open questions for future work.

2.1 Representing Deformable Objects

As highlighted in Chapter 1, rigid objects can be represented with 6–DoF, whereas the state of deformable objects requires a different representation due to their infinite number of DoFs, thus making it difficult to represent their shape. The shape of deformable objects has been represented using various methods [19], which can be categorised as implicit curves and surfaces, explicit parameterised representations, free-forms, or discrete representations. Whereas much work has focused on learning control using implicit curves and surfaces [8, 34], as well as explicit parameterised representations [35, 36], this chapter will, for the sake of simplicity, focus on discrete representations, which have been used in the publications of this thesis.



(a) Image representation. **(b)** Mesh representation. **(c)** Point representation.

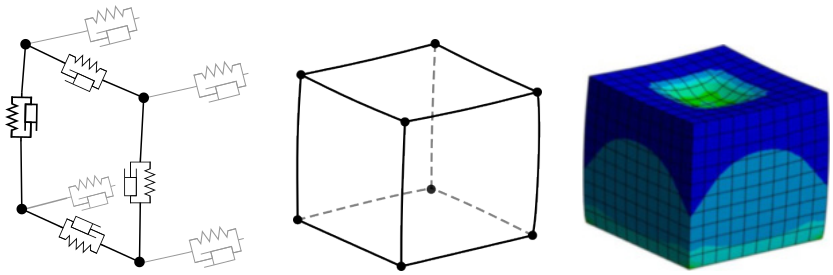
Figure 2.1. Image, mesh and point representations of a cloth. Note that the point representation can refer to a particle representation or a point cloud representation. In a particle representation, each point has associated attributes such as mass and material properties, while a point cloud representation associates the points only with their Cartesian position.

A discrete representation quantises the space, where the space is divided into a finite number of points, ranging from pixels to points and lines. These representations include, but are not limited to, images, meshes, particles, and point clouds (see Figure 2.1).

Images are 2-D matrices composed of pixels, also known as a pixel representation. The pixels represent a portion of space and can be associated with a colour, intensity, or depth value. This representation can be used for learning dynamics models using Neural Networks (NNs) [37] as well as for defining the state of granular materials using a depth or height map [38, 39].

Meshes are defined as a graph comprising nodes and edges. The meshes are made up of smaller structures formed by the vertices and edges of the graph, denoted as faces. These are usually defined as triangles or quadrilaterals for surfaces; and hexahedras or tetrahedras for the three-dimensional domain. This representation is used in simulation engines such as Bullet [40] or SOFA [41].

Particles can be defined as a set of points with attributes ranging from positions and masses to identifiers that specify the type of material associated with the particle [42, 43]. Particle representations have been extensively used in the animation of elastic and plastic objects [44], as well as fluids [45, 46], providing a wide variety of methods for computing the dynamics of deformable objects, such as Smoothed Particle Hydrodynamics (SPH) [46, 47] or the Material Point Method (MPM) [48, 49].



(a) Mass-spring system. **(b)** Particle system. **(c)** Constitutive model.

Figure 2.2. Example of different systems used for modelling the dynamics of deformable objects. The mass-spring and particle systems are discrete-based, whereas the constitutive model is a discretisation of the continuum mechanics via the Finite Element Method.

Point Clouds provide a 3-D representation of objects by defining them as sets of points in Cartesian space. This representation has been used for tasks such as garment generation [50], tracking [51], and reconstruction [52], as well as for grasping [53] and manipulation [54] of volumetric deformable objects.

2.2 Deformable Objects Dynamic Models

In order to capture the dynamics of deformable objects, different approaches have been presented to approximate their behaviour. Since the literature on these systems is quite extensive [19, 55–57], this section only covers analytical and data-driven approaches associated with a discrete representation, as discussed earlier. Throughout this thesis, these concepts will help us understand the different challenges faced when learning to manipulate different materials, as well as the limitations present in the simulation engines discussed throughout the dissertation.

2.2.1 Discrete Representation Models

Mass-Spring System

A common discrete-based model used for approximating the dynamics of a broad spectrum of deformable objects is the mass-spring system (MSS) (see Figure 2.2a), which includes objects ranging from cloth [58], soft tissues [59], and volumetric objects [60], to viscoelastic and viscoplastic materials [61, 62]. MSS represents deformable objects as a finite number of mass points connected by springs. Here, each mass point is defined by its position, velocity, acceleration, and mass. The system dynamics, that is, the temporal evolution of the points, can be determined following either Langrangian or Newtonian mechanics. These second-order Ordinary Differential Equations (ODEs) are typically solved using numerical integration methods [63], such as explicit or implicit methods. When

solved using Newtonian mechanics, the system includes constraints for the system to obey the laws of motion and to take into account external forces applied to the system. This gives rise to a convex optimisation problem, which can be solved by numerical algorithms such as the Conjugate Gradient (CG) or Newton-Raphson's method [64].

This type of model is used in simulation engines such as MuJoCo [65] and Bullet [40]. The MuJoCo simulation engine is used in both Publication IV and Publication V, while Bullet is used only in Publication V.

The main benefit of this type of model lies in its simplicity and lower computational complexity, which enable fast computation of the system dynamics for techniques such as closed-loop robotic manipulation [66]. However, optimising the spring coefficients to match real-world behaviour is a challenging task [67, 68]. Moreover, this model may not be suitable for handling large deformations, such as those present in granular materials [69, 70].

Particle system

Similar to the MSS, particle systems represent deformable objects as a collection of particles (see Figure 2.2b). Here, we discuss two methods for modelling the dynamics of deformable objects: Position Based Dynamics and the Material Point Method. In PBD, the particles are not connected via springs, unlike MSS. Instead, the particle interactions are enforced by multiple constraints, solved as an iterative optimisation problem [71]. Numerous constraints have been proposed to model volume preservation, stretching, or contact friction [72]. A positive aspect of PBD is its computational efficiency when accurately representing deformable object dynamics [73]. In addition, it is capable of modelling both elastic and plastic deformations.

The Material Point Method combines a particle-based and grid-based representation. This method represents the object as a set of particles with position, velocity, mass, and stress. We refer the reader to [74] for a more thorough explanation. One core strength of MPM is that it can compute the dynamics of objects that are subject to large deformations, which remain difficult to solve using other numerical methods such as the FEM. The MPM has been used to model a variety of materials, including fluids and granular materials [48, 75]. This method is used by the Taichi simulator [75], which was employed in Publication I and Publication II.

2.2.2 Continuum Mechanics

Unlike particle-based systems that model deformable objects as a discrete set of particles, continuum mechanics describe their behaviour in a continuous domain. The mathematical framework of continuum mechanics describes the relation between the forces, the strain, and the deformation of the objects. This framework uses constitutive models (see Figure 2.2c), which defines the relationship between the stress and strain of objects. These models can describe

the behaviour of a variety of materials, including viscoelastic, elastic, or plastic materials [76]. Due to the computational complexity of solving the equations of continuum mechanics systems, a common solution is to use the Finite Element Method, which divides the deformable object into smaller elements, enabling computing the deformation and stress to be computed within each element. This type of model is used in the simulation engine SOFA [41], which is used in Publication V.

Although these models produce simulations that are more precise and have a clear physical interpretation, solving the differential equations via FEM is computationally demanding.

2.2.3 Learned Dynamics Models

An alternative to the analytical models presented above is to use data-driven approaches that utilise Deep Learning methods for learning the dynamics of objects. In contrast to the aforementioned approaches, learning-based approaches, when trained using data from a high-fidelity simulator, can present a similar accuracy at a lower computational cost. In addition, if these models are trained using data from the real-world, they can capture phenomena that cannot be modelled using simulation engines, such as unmodelled friction or elasticity [77]. Table 2.1 provides an overview of earlier approaches, the deformable object dynamics they learn, and whether these models have been used in manipulation tasks.

Recently, approaches based on graph representations have attracted much attention due to their ability to intuitively represent particles of a deformable object as a set of nodes in a graph as well as the similarity of graph representations to analytical discrete dynamic models. Of the models using a graph representation in the literature, Graph Neural Networks have shown great promise as a model to learn the dynamics of deformable objects because of their capability to learn the relationships between the nodes and edges of a graph. One of the first approaches for representing deformable objects as particles and learning the dynamics using GNNs was proposed by Mrowca *et. al* [43]. Here, the graphs change dynamically over time, also referred to as dynamic graphs. This means that the relationship between the different particles and objects can change over time, replicating the physics behaviour of particle-based dynamic models. A key feature of dynamic graphs is that they can model discontinuities such as the contact between objects [78, 79]. One popular model is the Graph Networks, proposed by Sanchez-Gonzalez *et. al* [27]. This model exhibited significantly greater accuracy compared to previous methods, which sparked considerable attention, leading to multiple studies that leveraged this model for manipulating different materials (see Table 2.1).

Another potential approach is based on learning the dynamics of the object in either the pixel space [38, 91] or a latent space [37, 92] for planning robotic manipulation. Since data-driven methods using image-based representation

Table 2.1. Overview of recent work using graph representation for learning the dynamics of deformable objects (own work shown in bold). In this dissertation, we distinguish between the type of objects modelled: soft-bodies (SB), rope, cloth, granular materials, and heterogeneous materials (HM). Some works update online the model (MO), while others employ techniques for adapting to different materials (MA). The thesis also differentiates between those works that perform real-world experiments (RW), and those performing robotic manipulation experiments (RM).

	Year	Model	SB	Rope	Cloth	Granular	HM	Uncertainty	MA	MO	RW	RM
Mrowca *et. al* [43]	2018	HRN	✓	✗	✓	✓	✓	✗	✗	✗	✗	✗
Li *et. al* [78]	2019	DPI-Nets	✓	✗	✗	✓	✗	✗	✗	✓	✓	✓
Sanchez-Gonzalez *et. al* [27]	2020	Graph Networks (GNs)	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗
Pfaff *et. al* [28]	2020	GNs	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
Blanco-Mulero *et. al* [80]	2021	e-GGP	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗
Lin *et. al* [11]	2021	GNs	✗	✗	✓	✗	✗	✗	✗	✗	✓	✓
Tuomainen *et. al* [81]	2022	GNs	✗	✗	✗	✓	✗	✗	✗	✗	✓	✓
Salehi *et. al* [82]	2022	PhysGNN	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
Han *et. al* [83]	2022	SGNN	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
Shi *et. al* [84]	2022	GNs	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓
Wang *et. al* [85]	2022	GNs	✗	✓	✗	✗	✗	✗	✗	✓	✓	✓
Ma *et. al* [86]	2022	G-DOOM	✗	✗	✓	✗	✗	✗	✗	✗	✓	✓
Huang *et. al* [6]	2022	GNs	✗	✗	✓	✗	✗	✗	✗	✗	✓	✓
Wang *et. al* [87]	2022	GNs	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗
Deng *et. al* [88]	2022	Local-GNN	✗	✓	✓	✗	✗	✗	✗	✗	✓	✓
Driess *et. al* [89]	2023	Nerf GNN	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
Huang *et. al* [15]	2023	DefGraspNets (GNs)	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓
Longhini *et. al* [90]	2023	EDO-Net	✗	✗	✓	✗	✗	✗	✓	✗	✓	✗

often learn a dynamics model for planning manipulation, they will be discussed more thoroughly in the next section of this chapter.

2.3 Learning Dynamics Models for Deformable Objects Manipulation

This section provides an overview of state-of-the-art literature related to data-driven dynamics models for deformable object manipulation. The section discusses the variety of methods used for learning surrogate models, based on either images or graphs as a representation, along with their challenges and limitations. The reader can also refer to [93] for a general overview of approaches for learning the dynamics of cloth objects.

2.3.1 Image-based Representation

Early work on learning the dynamics of deformable objects from images used data collected in the real-world for training a surrogate model. For instance, Schenck *et al.* [38] use a height map to represent granular materials. They proposed learning a forward dynamics model, which given the current height map and an action, could be used to predict the next height map. Unlike [38], Nair *et al.* [94] proposed learning an inverse dynamics model in the pixel space, where given the current and target state of a rope, a NN infers the action required to

reach the desired next state. The authors collected data used for training the inverse dynamics model in the real-world, along with human demonstrations of how to manipulate the rope. Although the requirement for human demonstrations can be circumvented by using models such as Generative Adversarial Networks (GANs) [95] to generate the demonstrations [96], these approaches present several caveats.

First, gathering data in the real-world is time-consuming and might present safety concerns regarding the environment where the robot performs the manipulation tasks. Second, the data that can be obtained is restricted to the capabilities of the sensing systems. This is especially important for deformable objects, where sensing the deformation plays a vital role in robotic manipulation. Moreover, the collected data may not exhibit much variability. This can result in poor generalisation to different material properties, shapes, or environment variables, such as lighting conditions.

These issues can be alleviated by learning the dynamics from physics engines, which are inherently safe and provide a vast amount of training data. As an example, Tanaka *et al.* [92] proposed learning a forward dynamics model that predicts the next image state of a cloth after performing actions in a latent space. The model was first trained in simulation and then fine-tuned with real-world data.

It should be noted that the generalisation of data-driven models learnt from physics engines can be improved using techniques such as *domain randomisation* [97, 98]. This has been explored by approaches that learn a latent dynamics model using a contrastive loss [37] or a visual dynamics model from images [91]. In these works, the dynamics are learnt entirely in simulation and then transferred to the real-world by planning the manipulation actions using Model Predictive Control (MPC).

However, image-based approaches fail to account for the physical constraints of the object dynamics. Recent approaches solve this issue by using a particle-based representation rather than learning the dynamics in the image space. This enables adding physical priors provided by physics engines such as a differentiable physics simulator [99].

2.3.2 Graph-based Representation

In line with works that learn the latent dynamics of deformable objects using images, similar work has been proposed using a graph representation to learn the latent graph dynamics of a cloth defined by key points in an image [86]. However, an alternative option is to use a particle-based representation of the object to build a graph, where the nodes of the graph are the particles of the object (see Figure 2.3). This particle-based approach offers a more intuitive representation than images, as it resembles the particle-based dynamics models discussed earlier.

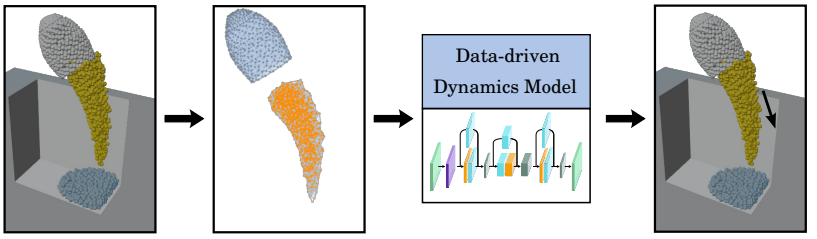


Figure 2.3. Example of a graph-based data-driven approach for modelling the dynamics of a granular material. In the left-hand image, the particles of the granular media (yellow) interact with the rigid-body (grey) for pouring the material into the desired shape (blue). The particles are then represented as a graph, where the data-driven dynamics model is used for predicting the acceleration of the particles and planning the respective manipulation actions. (Adapted from Publication II).

Extensive research has proposed diverse GNN models to learn cloth's dynamics for real-world manipulation (see Table 2.1). One of the first approaches was proposed by Lin *et al.* [11], building their work on the GNs model. Later, this work was extended to reconstruct a mesh from point cloud data [6], which improved the variety of garments that could be manipulated. A positive aspect of these approaches is that they can use incomplete point cloud data to reconstruct the cloth and plan the manipulation actions. More recently, Wang *et al.* [87] proposed using GNs to model the interaction between a cloth and a human in a dressing assistance task. In addition to predicting the cloth's dynamics, this work proposed learning a separate model of the force distribution exerted on the human while performing the manipulation task.

The standard approach for learning a dynamics model using a graph representation is to make use of ground-truth data from a physics engine, provided that the simulated material faithfully resembles reality. However, the success in manipulation tasks of these approaches deteriorates when transferred to the real world. This is a consequence of the sim-to-real gap, where some of the discrepancies come from unmodelled phenomena or using values for the material properties that do not capture the true object behaviour. For this reason, Longhini *et al.* [90] proposed a GNN model that incorporates an adaptation module to estimate the physical properties of the object that could bypass some of these issues in manipulation tasks. An alternative solution was proposed by Wang *et al.* [85], where a GNN is used to learn the dynamics in simulation along with a residual model that can adapt online the estimated state of a rope. While sensing the material deformation, either partially or completely, might be feasible for deformable objects such as ropes and cloths, this is more challenging for other types of objects, such as granular materials [100].

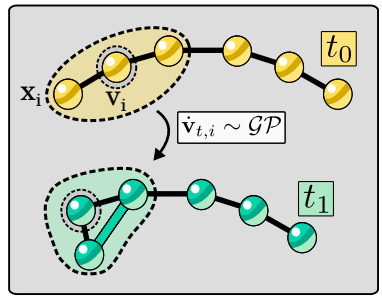


Figure 2.4. The evolving-Graph Gaussian Process proposed in Publication I models the transition of nodes in a dynamic graph using a Gaussian Process, $\mathcal{GP}$. Here, the connectivity of the graph changes from the initial time step t_0 to the next time step t_1. The model also accounts for the effect of the 1-hop neighbours of the node $\mathbf{v}_i$. (Originally from Publication II).

2.4 The Author's Contribution

In this dissertation, we address two limitations of existing works on learning the dynamics of deformable objects. First, the aforementioned works using a graph representation do not account for model uncertainty. This is beneficial in robotic manipulation when models need to adapt for different dynamics [101], or in model-based control approaches that account for the safety of the actions [102,103]. Second, when granular materials are manipulated, the related literature often neglects the interactions between the granular particles and the objects that manipulate the particles. These interactions are important for tasks such as pouring coffee grains from a recipient, where the movement of the recipient affects the position of all the grains that it contains.

2.4.1 Dynamic Graphs for Dynamics Modelling using Gaussian Processes

To address the first limitation, we introduce e-GGPs in Publication I, a non-probabilistic model to quantify the uncertainty for learning dynamics models using graphs. Gaussian Processes (GPs) are well-suited for modeling dynamics and quantifying the associated uncertainties [104]. Although GPs have been widely used to model the dynamics of systems in the Euclidean domain [105], few works have extended this to graph-structured domains [106–109]. In contrast to previous works, Publication I introduced a GP that models the transition of nodes in a dynamic graph (see Figure 2.4). GPs are defined by a mean function and kernel similarity function. In Publication I, we used a graph kernel [110] to measure the similarity of the graph node attributes as well as the 1-hop neighbouring nodes. This combines the advantages of GPs and dynamic graphs by capturing the dynamic interactions between particles of a deformable object using a graph, while quantifying the uncertainty of the model's predictions. The results not only showed that e-GGP was able to predict the dynamics of a rope

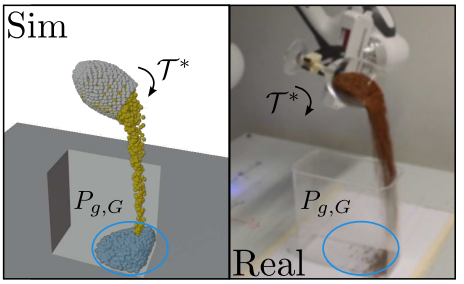


Figure 2.5. Overview of the granular pouring task of Publication II. A GNN is used to predict the dynamics of the granular particles and its interactions with the rigid object used for pouring the material. The optimal trajectory $\mathcal{T}^*$ of the cup is planned using the GNN to pour the material in the desired distribution $P_{g,G}$. The trajectory is then executed in the real world. (Originally from Publication II). ©IEEE 2022.

represented as particles in simulated experiments, but also demonstrated better accuracy than other GPs working either in the Euclidean or graph domain. However, the computational cost of the method limits its scalability for real-world manipulation.

2.4.2 Learning Material Interactions for Pouring Granular Materials

Following the success of GNNs for learning the dynamics of various deformable objects, the GNs framework was proposed in Publication II for learning the pouring of granular materials (see Figure 2.5). In that work, a forward dynamics model was proposed to learn the dynamics of a granular material. The proposed model takes into account the interaction between the particles and a rigid object for planning the pouring of the grains (see Figure 2.3). In contrast, prior works manipulating granular materials have used images to learn a dynamics model [38] or a controller [39, 111] for tasks such as shaping or scooping the material. Publication II showed that a GNN model can be used both to learn the dynamics of the manipulated material using data from the Taichi high-fidelity simulator [75], as well as to leverage this model for trajectory planning. The graph representation distinguished between two materials: the granular media and the rigid object that is used for pouring the material. Here, the rigid object is controllable, and the exact pose of its particles are known. Thus, the dynamics model was used to predict only the acceleration of the granular particles. Then, the semi-implicit Euler integration method was used to compute the particles position from the predicted accelerations.

Additionally, Publication II was the first work to treat the problem of pouring a granular material as an optimal transport problem. A key advantage of this work is that by utilising a particle-based representation, the particles of the material can be regarded as a distribution. On the other hand, height or density maps are restricted to planar surfaces and can offer less detail for fine-grained materials. Hence, a trajectory was planned by minimising the Wasserstein

distance between the distribution of the granular material and a target particle distribution. Then, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) optimiser was used for minimising the Wasserstein distance.

Our results showed that the method was able to match the desired particle distribution in three of the four experiments when using the GNN dynamics model. When transferred to the real-world, the method was able to closely match the target configurations in two of the four cases, where one of the remaining cases slightly differed from the goal, and the last case showed a large distance from the expected target distribution. One of the main reasons for this performance was the failure of the dynamics model to extrapolate, which could be circumvented by training the model using data that covers a wider spectrum of cases, thereby improving the generalisation capabilities of the GNN model. In addition, the gap between the This could be solved by closing the sim-to-real gap with techniques such as system identification for identifying the simulation parameters using data from the real world or real-to-sim [112–114] for improving the fidelity of the simulator.

2.5 Conclusions

This chapter has discussed previous works that use data-driven dynamic models for the manipulation of deformable objects, exploring image-based and graph-based representations. An unexplored issue in the discussed approaches for learning dynamics models is that of accounting for model uncertainty, which could help to explore control approaches when the sensed deformation is not accurate. For the case of graph-based representations, one possible direction would be to use models that are less computationally demanding than GPs, such as Bayesian GNNs [115, 116] or combining GPs with ODEs [117]. Moreover, the accuracy of these models could be further improved. Thus, future work could explore incorporating physics knowledge into the learning of the dynamics model. This is an active research topic referred to as physics-guided deep learning [118]. This has been recently explored in learning of deformable object dynamics models by Chen *et al.* [99].

Furthermore, a common limitation of data-driven approaches falls within the challenges of transferring a behaviour learnt in simulation to the real world, as further discussed in Chapter 4 and Publication V. A solution to this problem includes system identification [113, 114, 119]. However, this remains a challenge for deformable objects where the deformation of the material cannot be measured due to hardware limitations, such as the particles of granular media. One potential solution would involve merging visual information available in the real world, i.e. depth data, with physics simulation to develop a latent representation of the deformation. Another effective approach to bypass the sim-to-real gap, which has been less explored by other works, is the online adaptation of these models, similar to the work proposed by Wang *et al.* [85]

discussed earlier. Future research could also explore combining model-based approaches with real-time visual feedback, where a trajectory is planned using the surrogate model and adapted in real-time. In the next chapter, we will discuss a model-free method to adapt in real-time the manipulation actions, thus reducing the sim-to-real gap and addressing the adaptation to multiple materials, as presented in Publication IV.

3. Learning Adaptive Policies for Deformable Object Manipulation

Having discussed tools to build a representation and a dynamics model of our pottery mug, we can say that the mug is finished, and we want to use it for the first time. Let us consider a task where a robot needs to autonomously hand it over to a human. To successfully perform this task, the robot must take into account the content of the mug. If the mug is full of water; the robot should move slowly to avoid spilling its content. By contrast, if the contents of the mug is more dense, such as honey or liquid soap, the robot's motion can be more abrupt. Based on this observation, we can conclude that the robot will require an adaptive behaviour that can adapt to the content's material properties. Material properties are a critical factor in the deformation of deformable objects and, consequently, in their manipulation. Hence, how can a robot *adapt* its manipulation actions to handle a variety of materials?

This chapter focuses on learning policies that can adapt to different deformable object properties, such as shape, size, or material properties. The chapter begins by outlining the manipulation techniques described in related works based on the task definition. Afterwards, the chapter moves to discussing the state-of-the-art in the manipulation of deformable objects. Finally, the chapter summarises two of the thesis contributions: Publication III for adapting the set of manipulation primitives performed by a robot and Publication IV for adapting robot manipulation actions in real-time.

3.1 Manipulation Techniques

First, this section establishes the notion of manipulation, which encompasses the related state-of-the-art discussed in subsequent sections of this chapter. The manipulation techniques that are applied to perform tasks can be analysed based on the task's requirements, following the taxonomy of Mason and Lynch [30]. This taxonomy characterises the type of manipulation based on four properties: kinematics, static forces, quasi-static forces, and acceleration forces. For the case of manipulating deformable objects, the techniques fall mainly into two categories: *quasi-static manipulation* and *dynamic manipulation*. This section

(a) High velocity.

(b) Medium velocity.

(c) Low velocity.

Figure 3.1. The final result of a trajectory performing a quasi-static manipulation task, diagonal cloth folding, executed at three different velocities. The end configuration of the cloth is the same regardless of the velocity at which the manipulation actions are performed.

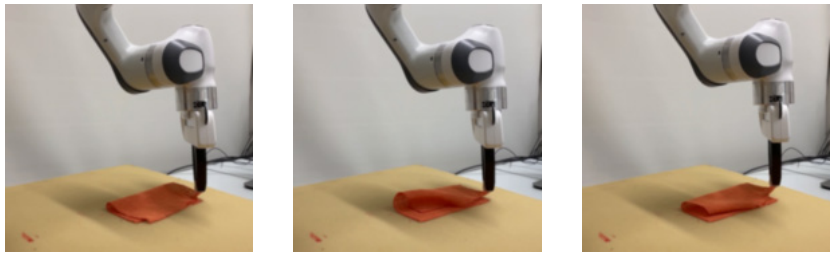
continues by defining each manipulation technique, while providing use cases in the context of deformable object manipulation.

3.1.1 Quasi-static manipulation

As defined by Mason and Lynch [30], quasi-static manipulation tasks are those where frictional and impact forces are predominant, neglecting inertial forces, such as acceleration forces. The task of diagonally folding a piece of cloth [10,120] is one of the tasks meeting this criterion (see Figure 3.1). In this task, the speed of the manipulation action does not affect the result of the task, as dynamic forces do not come into play. Tasks that can be solved by performing quasi-static manipulation include folding, unfolding, and placing of cloths [121, 122] as well as shaping granular materials [123, 124]. Quasi-static manipulation primitives, such as Pick-and-Place (P-n-P) primitives, are explored in Publication III for the task of cloth unfolding.

3.1.2 Dynamic manipulation

When inertial forces are pivotal for the success of the task, the manipulation is referred to as dynamic manipulation [30]. Some examples of dynamic manipulation are non-prehensile dynamic manipulation actions, such as throwing, dynamic catching, or sliding [125]. An example of dynamic deformable object manipulation is sideways folding using a single grasp point [120]. Here, a square piece of cloth needs to be folded by matching its top and bottom corners, where a single corner of the cloth is grasped by a manipulator. In this case, the velocity of the manipulation actions has an effect on the final shape of the cloth (see Figure 3.2). Additional examples of dynamic manipulation tasks include flinging cloths for unfolding [32], dynamically placing a cloth on a flat surface [120, 126, 127], sliding and tossing pizza dough [128], or dynamically manipulating ropes and cables to reach a specific target [127, 129]. Within the set of dynamic manipulation tasks we can differentiate between highly dynamic



(a) High velocity. **(b)** Medium velocity. **(c)** Low velocity.

Figure 3.2. The final result of a trajectory performing a dynamic manipulation task, sideways cloth folding, executed at three different velocities. The velocity of the manipulation actions affects the final configuration of the fabric.

tasks, such as throwing a basket ball, where the velocities when releasing the ball con go up to 6 m/s [130], and less dynamic tasks, such as swinging a cloth, where linear velocities of the robot swinging the cloth can go up to 1m/s (see Publication V).

As depicted by Lynch and Mason [131], dynamic manipulation offers some advantages:

- **Increased workspace**: when using dynamic actions, the workspace can be defined as the set of states reachable by the manipulated object. For example, by swinging a cable, a robot can reach points outside of its kinematic workspace.
- **Increased reachable configuration space**: dynamic actions enable reaching parts of the object's configuration space that cannot be reached by using quasi-static manipulation.

Thanks to these advantages, dynamic manipulation actions can be more *task-efficient* than quasi-static ones [32, 132]. However, the advantages presented above come at a cost. Planning dynamic manipulation actions adds complexity to the task due to the need to account for the object's inertia. This is particularly important in deformable objects, where the properties of the material have a larger effect on the object dynamics. Moreover, if by performing dynamic manipulation the deformable object switches from a contact state to contact-free state during the manipulation, the dynamics change, resulting in non-smooth dynamics. This can be particularly challenging for model-based approaches to learn, or might even present a significant gap when simulating the system.

Dynamic manipulation is applied in four of the five publications: in Publication II for pouring granular media, Publication III for the task of cloth unfolding, Publication IV for sideways cloth folding, and Publication V for placing a cloth in a flattened configuration on a flat surface.

a) Before manipulation. b) During trajectory. c) After trajectory.

Figure 3.3. Dynamic manipulation of a deformable object showing the deformation a) before the robot executes the trajectory, b) while the robot trajectory is performed, and c) after the trajectory has finished. (Adapted from Publication V).

3.2 Manipulation of Deformable Objects

Multiple studies have examined the control approaches applied to the manipulation of deformable objects [5, 19, 72, 133]. In this section, we categorise the methods into model-based and model-free approaches. In Section 3.2.1, the model-based methods comprise planning and control algorithms that utilise a dynamics model [134]. Section 3.2.2 considers both model-free approaches that learn a controller by interacting with an environment (e.g., RL techniques [135]), as well as methods that learn to mimic behaviour from expert demonstrations, such as Imitation Learning (IL) methods [136].

Furthermore, this section considers two control strategies for performing deformable object manipulation. The first strategy performs low-level control (LLC), where the controller acts based on the object deformation at every time instant of the manipulation, see Figure 3.3 b). The second strategy performs high-level planning (HLP), where the controller receives as information only the initial and final deformation state; see Figure 3.3 a) and Figure 3.3 c).

3.2.1 Model-Based Control

Early works in model-based control used analytical models to approximate the dynamics of the deformable object [137,138]. In these works, the dynamics model is used for planning the manipulator trajectories of various tasks, including garment folding [122], where a trajectory is optimised to match a desired shape, or shaping an elastic rod using optimal control [139]. Although these models are coarse approximations of the dynamics of the system, which may result in a sub-optimal trajectory, they enable closed-loop feedback over the whole deformation of the object, as shown recently by Luque *et al.* [66]. By contrast, some works have used high-fidelity simulators based on FEM [126, 140, 141] for planning a low-level control trajectory while taking into account the deformation throughout the manipulation. However, these methods run the control in an open-loop fashion due to the computational resources, which might result in undesired behaviour if the simulator suffers from a large reality gap. Moreover, these methods require finding the optimal parameters of the dynamics model

that match the object deformation, offering little adaptation for manipulating a variety of materials.

An alternative approach is to learn the system dynamics, as described in Section 2.2.3. The proposed methods include learning forward dynamics models [6,8,78,91,142], that use techniques such as MPC, random shooting or latent policies to plan the trajectory (i.e., HLP), as well as learning inverse dynamics models [92,143], where the trajectory is planned in a latent space. One of the limitations of these approaches is that they plan high-level actions, such as deciding the pick and place position of P-n-P primitives. Hence, the deformation occurring while performing the manipulation actions is neglected. Further works have discussed refining a learnt model of the manipulation trajectory, rather than the dynamics model, from observed trajectories [127]. This process enables low-level control at the expense of using multiple iterations to fine-tune the trajectory. Similar to the works that learn the deformable object dynamics, the trajectory is executed in open-loop, presenting the same disadvantages as the FEM control approaches discussed earlier.

The aforementioned methods suffer from some common disadvantages of model-based methods. First, the optimality of the solutions is subject to the accuracy of the model, as some assumptions are made on the deformation model. Second, these methods suffer from poor adaptability and generalisation to the materials of different objects. Alternatives to bypass this problem include adapting the model online [85] or using models that can adapt to different material properties [90]. In contrast, model-free approaches do not need to explicitly model these phenomena, resulting in better generalisation and adaptation across a wider range of material properties.

3.2.2 Model-Free Control

Model-free approaches learn a policy by leveraging data gathered in simulation or real-world experiments, rather than building or learning a model of the environment for planning as done by model-based methods [135]. Within the set of deep learning techniques, the related literature has explored both reinforcement learning methods [10,120,144–147] as well as self-supervised methods [129,132,148,149] to learn a policy by interacting with an environment. In these methods, a reward function is often defined as a function of the object's deformation, which serves as the objective for training the policy. Since the vast majority of these works use visual feedback to learn the policies, either for LLC or HLP, these will be discussed more thoroughly in the next section of this chapter.

Unlike DL approaches, IL does not require a reward function, thus eliminating the effort of hand-designing reward functions. Early works explored learning to reproduce human demonstrations collected using techniques such as teleoperation [150] or kinesthetic teaching [151]. In order to adapt the demonstrations to unseen situations, Lee *et al.* [152] proposed adjusting the trajectories of

the demonstrations by measuring the object's deformation using point clouds, thereby enabling better generalisation for ropes of different lengths. However, the adaptation capabilities are still limited to the demonstrations. Rather than adapting the demonstration, further works have explored learning a policy that matches the behaviour of the expert policy given an image as observation for tasks such as bed-making [153, 154] or cloth manipulation [155, 156]. One limitation of these approaches is that the performance of the demonstrations can be sub-optimal, which will inevitably hinder task performance. Moreover, in order for the policies to have an adaptive behaviour, the set of demonstrations needs to be a distribution that covers a wide range of object properties. This would highly increase the human-effort required for collecting the data set, thus being less efficient due to required interaction with the real environment. Nevertheless, these demonstrations can be used to bootstrap the process of learning a policy, as shown in [10, 157, 158] and Publication IV.

Apart from model-free approaches, another potential strategy has been to combine model-based and model-free approaches [159]. However, this has been less actively explored in the literature concerning manipulation of deformable objects [160]. Due to the aforementioned limitations of model-based approaches and the limited amount of literature on combining model-based and model-free methods, the following section will concentrate on model-free techniques.

3.3 Learning Policies from Experience using Visual Feedback

In deformable object manipulation, employing images as input to policies is a widespread practice, as they are readily available in the real world and provide an information source for decision-making. The training methods vary from self-supervised methods to RL methods, where the data is collected either in the real-world from robot interactions [148] or human-annotated data [132], or in simulated environments [32, 161]. Rather than exploring the differences in the methodologies used to learn the policy, such as the network structure or learning strategy, this section discusses how these works deal with the manipulation of different materials, thereby dividing the methods into high-level planning and low-level control.

3.3.1 High-Level Planning

Learning visual feedback policies for HLP includes applications such as manipulating ropes [149], cloths [32, 132, 162, 163] and bags [149, 164, 165]. A common methodology in these works is to learn a policy that proposes the pick or place position of the manipulator and then to apply a quasi-static manipulation primitive [148, 149, 161]. To achieve generalisation across a variety of materials, many studies employ simulation-based learning approaches along with domain randomisation. Although this improves the generalisation capabilities of the

policy, the performance when transferring the policy to the real-world is highly tied to the fidelity of the simulator. On the other hand, approaches that learn directly from real-world interactions [148] require changing the real set-up to provide a distribution with the properties of various objects, thus presenting the same limitations as IL approaches when collecting human demonstrations.

Recently, Ha *et al.* [32] proposed a dynamic manipulation primitive referred to as *fling*. The proposed policy decides the pick positions and executes the fling motion in an open-loop fashion. This primitive has provided great results on different cloths and t-shirts, showcasing great efficiency in terms of the amount of iterations required to succeed with the task. Due to the effectiveness of dynamic manipulation primitives in cloth manipulation, this motion has been later used in combination with quasi-static primitives for garment folding and unfolding [132]. However, the parameters of these primitives, such as their velocity, are kept constant, which plays a crucial role, as shown in Figure 3.2.

A noteworthy limitation of these methods is that they run in an open-loop fashion, presenting the same limitation as model-based methods that run LLC trajectories optimised using a dynamics model, which may result in discrepancies between the anticipated and real manipulation outcome.

3.3.2 Online Adaptation

While extensive research has explored learning HLP visual policies, the development of LLC policies has been less actively explored. Zhang *et al.* [39] proposed learning a policy for gathering, spreading and flipping visco-plastic and granular materials in simulation. The policy receives the state of the material via density and height maps, and then proposes the tool pose. The work evaluated the performance of the policy when applied to different materials by changing the resolution of the granular particles. Since some of the tasks require dynamic manipulation, the fidelity of the simulator is crucial when transferring the policy to the real world.

Other works have explored learning policies in simulation and transferring them to the real world for manipulating cloth [10] and tissue [147]. The work by Matas *et al.* [10] proposed transferring a visual feedback policy learnt entirely in simulation for quasi-static cloth manipulation tasks. Although the transfer to the real world showed great performance when the grasp was successful, only one type of cloth material was used. Hence, the effectiveness of the policy when applied to different types of fabric was not evaluated.

3.4 The Author's Contribution

A major challenge in deformable object manipulation, which few works have explored, is that of learning policies that can adapt and manipulate a variety of deformable object materials and shapes. This thesis explores two strategies to

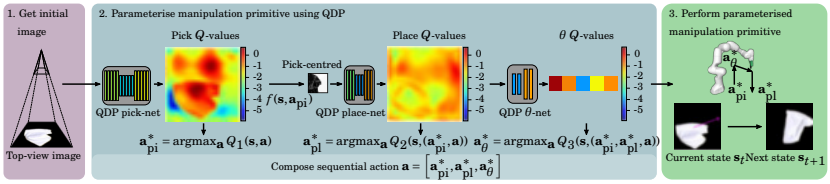


Figure 3.4. Overview of the proposed Quasi-Dynamic Parameterisable (QDP) approach proposed in Publication III. First, the method gathers a top-view image of the cloth (left). Then, QDP proposes the manipulation primitive parameters through the sequential action $\mathbf{a}$ (centre). This action is composed of three sub-actions: the optimal pick position $\mathbf{a}^*_{\text{pi}}$, the place location $\mathbf{a}^*_{\text{pl}}$, and the additional primitive parameters $\mathbf{a}^*_{\theta}$. Finally, the manipulation primitive is executed, placing the cloth in a new state (right). (Originally from Publication III). ©IEEE 2023.

approach this problem: 1) closed-loop feedback of low-level control policies, and 2) adapting manipulation primitives using high-level planning policies.

3.4.1 Learning to Sequentially Adapt Manipulation Primitives

In Publication III, we proposed the Quasi-Dynamic Parameterisable (QDP) method, a RL approach that learns a policy to modify the parameter values of both quasi-static and dynamic manipulation primitives for the task of cloth unfolding (see Figure 3.4). Unlike earlier works that perform HLP, given an input image, the proposed policy decides the primitive parameters, such as the height or velocity of a manipulation primitive, along with the pick and place locations. To determine these parameters, we extended the Sequential DQN (SDQN) approach [166, 167] and formulated the problem to sequentially decide the pick, the place, and the additional parameters of the primitive. One of the benefits of this sequential process is the reduction in computational complexity. Furthermore, it enhances the adaptability capabilities of high-level planning approaches. The performance of both quasi-static and dynamic manipulation primitives was evaluated in the real-world by comparing their effectiveness for different cloth shapes, sizes, and materials in both simulation and real-world experiments. However, despite utilising domain randomisation to enhance the policy generalisation capabilities, the sim-to-real gap remained significant, a discrepancy that could be mitigated by retraining, as in [32], or by identifying cloth properties, as demonstrated in Publication IV.

3.4.2 Adapting the Manipulation Actions in Real-Time via Visual Feedback

In Publication IV, the method proposed in [10] was expanded to perform dynamic cloth manipulation of multiple fabric materials (see Figure 3.5). For this purpose, Soft-Actor Critic (SAC) was used to learn a closed-loop visual feedback policy, which was bootstrapped by human demonstrations. Unlike the approach in [10], Publication IV also randomised the physical properties of the cloth, where the

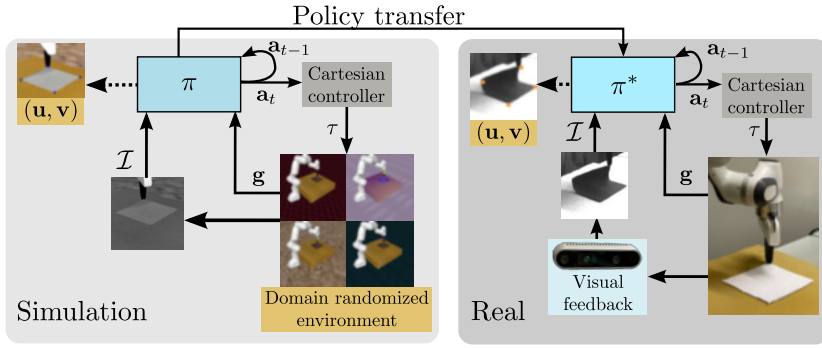


Figure 3.5. Overview of the proposed method in Publication IV. The policy is trained in simulation (left), where the simulated environment randomises the dynamics of the cloth as well as the environment's visual properties. The trained policies π^* are transferred directly to the real world (right). The policy receives an image $\mathcal{I}$, a goal $\mathbf{g}$ and the previous action $\mathbf{a}_{t-1}$ as input. The policy outputs the next action $\mathbf{a}_t$ as well as a prediction of the cloth corners $(\mathbf{u},\mathbf{v})$. Both simulation and real setup use the same Cartesian controller, which generates the torque commands τ to be executed by the manipulator. (Originally from Publication IV). ©IEEE 2022.

distribution of the cloth's parameters was identified using the collected human demonstrations.

Publication IV addresses two critical aspects discussed throughout this thesis: adaptability and efficiency. First, domain randomisation helps to find a policy that can generalise and adapt to the deformations of different fabrics, which are accentuated during dynamic manipulation. Second, the method demonstrates efficiency in two aspects: 1) the dynamic manipulation requires a single interaction with the environment to succeed with the task, and 2) the policy is learnt in simulation, with limited human-effort for gathering the demonstration. One of the assumptions of this work is that the task of grasping the cloth is solved, which is one of the major challenges in deformable object manipulation [5]. Another challenge faced was providing the policy with visual input at a high frame rate to enable the robot to act rapidly and adapt its actions based on the cloth deformation. This was solved by using a low-resolution greyscale image, which increased the frequency of the visual feedback given by the sensor. However, this does not provide as detailed information about the deformation as depth sensors. This is primarily a hardware limitation, which could be solved by employing depth sensors capable of delivering feedback at a high frame rate.

3.5 Conclusions

This chapter has both provided an overview of methods for learning to manipulate deformable objects, as well as analysed the capabilities of these methods in adapting to diverse material properties. Although model-based approaches can find optimal low-level control trajectories, the optimality of the trajectory is

conditional on the accuracy of the dynamics model, which is often restricted to model the dynamics of a single material. Hence, for enabling the manipulation of multiple materials with these methods, further work should explore dynamics models that adapt to various materials. On the other hand, model-free methods offer more adaptability when trained with sufficient and varied data. However, current research has mainly explored high-level planning policies. One strength of low-level control policies is that they can plan actions throughout the manipulation, measuring the object's deformation and adapting these actions, as demonstrated in Publication IV. However, adapting the robot trajectory in real time for more complex tasks, such as assisted dressing [163], might require a combination of adaptive high-level planning policies with low-level control. Future research could examine combining the methods proposed in Publication III and Publication IV or combining model-based and model-free approaches, where the model-based method would propose an optimal trajectory, and this trajectory would then be adapted to the current object deformation by the model-free approach. Furthermore, future research could investigate extending the hybrid control methods proposed by Pinosky *et al.* [168] or Jenelten *et al.* [169] to deformable object manipulation.

Although closed-loop visual feedback policies present clear benefits for deformable object manipulation, sensing the deformation is more challenging for granular materials. A promising avenue for research could be to investigate methods that can infer the granular material deformation in real-time, such as implicit neural representations [34], and to provide this input to a low-level control policy.

In addition to being more task-efficient than quasi-static manipulation approaches, methods that use dynamic manipulation benefit from an extended workspace, enabling tasks that are not feasible with quasi-static actions. For this reason, further research should explore learning policies that benefit from this manipulation technique.

Simulation engines are extensively used for planning, learning object dynamics, or learning policies. By using physical engines, researchers can conduct a large number of trials safely without real-world interaction. Furthermore, there is full access to the object deformation state, which is especially important for deformable object manipulation. Given these benefits, as well as the limitations mentioned in this chapter and Chapter 2, the next chapter delves into the sim-to-real gap and the impact it has on manipulation, as seen in Publication II, Publication III and Publication IV of this thesis.

4. Simulation to Real Gap in Deformable Objects

Our journey towards autonomously handing over a mug to a human started from the representation and dynamics of the mug, followed by learning behaviours that enabled a robot to adapt its actions according to the mug's content. Over this journey, we have briefly looked into the simulation domain and the possibilities that it offers for learning this behaviour. Now, our narrative takes us to the domain where both simulation and the real world converge. As soon as we deploy our robot into the real world, we experience that the learnt behaviour differs from what it exhibited in simulation. This raises several questions: How significant is the discrepancy between simulation and reality? What can be done to minimise this gap?

This chapter explores the gap between simulation and reality, also known as the sim-to-real gap, in the context of deformable objects. The chapter opens with the relevant state-of-the-art techniques for quantifying the reality gap. Thereafter, it examines the approaches used to bridge the sim-to-real gap. The chapter concludes with a discussion of the methodologies used in Publication V for quantifying the reality gap, as well as the methods used to deal with this gap in Publication III and Publication IV.

4.1 Measuring the Reality Gap in Manipulation

This section explores the state-of-the-art techniques used to measure and evaluate the reality gap in deformable object manipulation. The discussed literature is divided into two types of work: 1) those that evaluate the gap in task performance and 2) those that measure the gap by comparing real-world dynamics measurements with simulation.

4.1.1 Benchmarking the Manipulation Performance

Most of the work in deformable object manipulation has primarily focused on evaluating task performance in the real-world. The most common approach has been to perform zero-shot transfer of the policies and evaluate their per-

formance, both in simulation and the real-world [11, 158, 161, 162, 170]. In this direction, Wang *et al.* [85] compared the performance of a dynamics model learnt in simulation for planning a trajectory using zero-shot transfer, dynamics randomisation, re-training with real data, and using a separate dynamics model for planning online. This work demonstrated that the online model outperformed the aforementioned approaches. However, the evaluation methods used in the related literature present some limitations. While some of the literature focuses on solving the same task, such as cloth unfolding, the simulators used for training, as well as the set of real-world objects upon which they are evaluated are different. This results in an inconsistent evaluation of the algorithms' performance. For this reason, recent work has focused on establishing data sets for benchmarking cloth manipulation [171], and defining standard tasks to assess control strategies [172] in order to provide a common framework for evaluating manipulation performance.

4.1.2 Quantifying the Reality Gap

This section overviews approaches that quantify the reality gap using techniques that quantitatively measure the disparity between simulation and data collected in the real-world for applications such as fabrics and granular manipulation. This gap between simulation and real world can be caused by discrepancies in interactions between different objects, such as collisions between deformable and rigid objects, unmodelled physics, or disparity in the physical properties of objects [29, 173]. The techniques used to quantify the reality gap extend from system identification by measuring the sim-to-real discrepancy after the simulation parameters have been identified. Aside from the specific methods that optimise the simulation parameters, there are two key aspects for quantifying the gap: measuring deformation and the representation of the deformable object.

The approaches found in the literature to measure the object's deformation range from using markers placed on the object [114], to more generic methods such as using images or point clouds [113, 119]. To identify the parameters of cables, Cheng *et al.* [114] proposed placing fiducials on a cable and representing it as a rigid manipulator. To evaluate the reality gap, they measure the discrepancy between the cable's joint positions in sim and real. However, this approach demands significant human labour for setting the fiducials and collecting the data set. Furthermore, most simulation engines represent deformable objects as collection of particles [40, 41, 65, 174], thus restricting the applicability of this approach. On the other hand, Lim *et al.* [113] extended the approach in [114] by extracting the points of a manipulated cable from images and collecting the data set in an automatic manner. Here, the reality gap is assessed in terms of the final cable configuration, as well as the position of the cable throughout the manipulation trajectory. Similarly, Sundaresan *et al.* [119] extract the deformation of cloths using a point cloud and evaluating the distance between the real-world point clouds and meshes from a differentiable simulator.

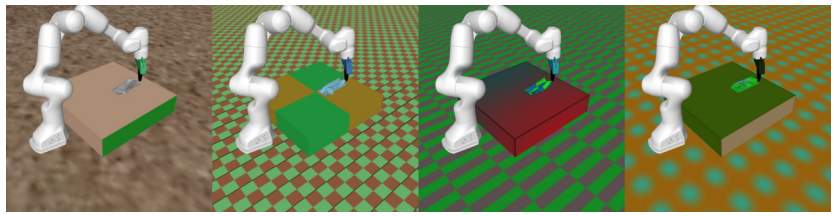


Figure 4.1. Domain randomisation of physical appearance and texture of the objects in a simulated scene of a robot manipulating a cloth using the MuJoCo simulation engine. (Adapted from Publication III).

Although the aforementioned approaches are valid for measuring the gap in simulated cables and cloths, this is more challenging for other objects, such as granular materials. The task of measuring the reality gap in granular media has been addressed by Matl *et al.* [112]. Here, the reality gap is evaluated using two methods. First, they evaluate the error in simulated and real depth images of the granular material. Secondly, they manipulate the fine-tuned material in both simulation and the real world, allowing them to identify the amount of grains that bounce off in a pouring task. However, the system identification and reality gap measured are restricted to the task of pouring couscous and barley. The reality gap in other tasks such as scooping [175, 176] or granular pile manipulation [177], as well as other granular materials such as rice, beans, or ground coffee, remains open.

4.2 Closing the Sim-to-Real Gap

This section covers techniques used for closing the gap in sim-to-real manipulation. These methods include domain randomisation, domain identification, knowledge distillation, and meta-learning [178]. The section is restricted to the two most widespread methods used in deformable object manipulation: domain randomisation and domain identification.

4.2.1 Domain Randomisation

So far, the previous chapters have explored approaches that learn policies or surrogate models in simulation, as well as their subsequent transfer to the real-world using domain randomisation [97, 98, 179] for transferring the system to the real world. The objective of domain randomisation is to provide policies or dynamics models with diverse data during training time in order to generalise better to environmental conditions or cover a wider spectrum of dynamics. To achieve this, a distribution is defined across various properties of the simulation domain. Then, the values for these properties are sampled from the distribution and modify the simulation scene during the training of the system. The simulation attributes that can be randomised, which cover some of the causes that

originate the reality gap discussed in Section 4.1.2, include the following sets:

- Object properties. This refers to the physical properties of the objects, such as the stiffness [173] or the Young's modulus of soft objects [180], as well as their size, shape, and location.
- Visual properties. These attributes involve the visual aspects of the simulation scene (see Figure 4.1). They encompass characteristics such as camera position, lighting conditions, as well as the physical appearance and texture of the objects [97, 181].
- Physics engine properties. These properties are related to the simulator's engine. The attributes range from more general simulation parameters, such as friction [173, 182] or air density [183], to more specific attributes of the simulator, such as the simulation frequency, which impacts the accuracy of the simulation.

Within the scope of sim-to-real transfer for manipulating deformable objects, domain randomisation has been extensively applied for learning policies that generalise to different object locations [129], as well as learning visual policies that are agnostic to the textures and colours of the object [10, 32, 91, 158, 161–163, 170]. However, domain randomisation presents some limitations. Firstly, a substantial sim-to-real gap can significantly reduce the performance of the policy in the real-world. This issue can be alleviated by retraining the learnt policy with experience collected from the real-world [32, 163]. A different solution is to use closed-loop feedback policies, as previously discussed in Chapter 3. Another limitation of domain randomisation is that training policies with a naive randomisation can lead to high variance policies [184], which might be detrimental when transferring the learnt policies to the real-world [10]. One of the solutions for this problem is learning the distributions of parameters to randomise [173], thus obtaining a suitable randomisation for transferring policies to the real-world in a zero-shot manner.

4.2.2 System and Domain Identification

An alternative method for reducing the sim-to-real gap is the traditional approach of system identification [185–187]. This approach aims to identify the simulation parameters that better align the simulation behaviour with data collected from the real world. As emphasised throughout the dissertation, measuring the object's deformation in the real world is not straightforward. One solution is to use visual feedback, such as images or point clouds, to optimise the simulation parameters that better match the shape of simulated and real data [26, 113, 114, 119, 188]. Although the forces applied to the deformable objects cannot be directly measured from visual input, these can be estimated from, for example, the torques applied using a robot manipulator [189]. Another solution is to quantify the deformation using vision systems along with tactile or force

sensors [190–192], which can provide a more accurate measurement of the forces exerted on the deformable object.

Building upon system identification, domain identification tries to identify a distribution of parameters, rather than specific values for a certain material. In our context, Matl *et al.* [112] proposed BayesSim, a Bayesian framework for identifying the distribution of granular materials' parameters from depth images. By learning a policy that generalises across a material distribution, it can be less prone to overfitting to a specific material, while adapting to the dynamics of different materials.

4.3 The Author's Contribution

This dissertation takes inspiration from prior research attempting to bridge the sim-to-real gap for zero-shot transfer, as well as to quantify the reality gap in cloth manipulation tasks.

4.3.1 Addressing the Reality Gap in Policies Learnt from Simulation

In Publication III, a policy was trained using Flex [174] by randomising both the visual and physical properties of the cloth. The policy is then transferred in a zero-shot manner without further tuning. However, in real-world experiments, the results showed a noticeable decrease in performance. This was attributed to not randomising other parameters, such as the simulation friction coefficients, as well as the absence of closed-loop feedback, which aligns with the issues discussed in this chapter. This could have been circumvented, to some extent, by retraining the policy with real data. However, this would not provide a sensible comparison of the gap in terms of task performance. In Publication IV, a closed-loop visual feedback policy was trained using MuJoCo [65]. Similar to related work, the training process includes the randomisation of visual and cloth properties. In addition, our work also randomises simulation parameters such as the contact friction. In the real world, the visual input is provided by a camera that has a frame rate with unstable latency. Thus, to accommodate for these observation delays [193], the policy was trained with random temporal delays. In this work, the real-world experiments presented similar performance to that exhibited in simulation. In both of these works, the reality gap was only evaluated in terms of the performance of the policy in simulation and real-world experiments.

4.3.2 Quantifying the Reality Gap in Dynamic Cloth Manipulation

Additionally, to quantify the sim-to-real gap in simulation engines, Publication V proposed a data set to measure the gap of deformable object simulators (see Figure 4.2). The data set was collected by performing a dynamic and a quasi-

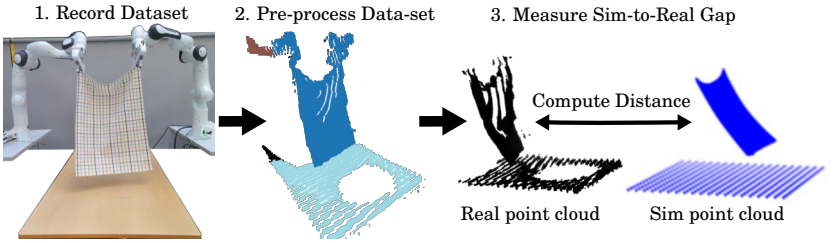


Figure 4.2. Overview of the process used to measure the reality gap in Publication V. First, a data set was recorded in the real-world and pre-processed to optimise the simulation parameters. After the simulation parameters were fine-tuned, the reality gap was measured by computing the distance between the simulation and real point clouds of the deformable object. (Adapted from Publication V).

static cloth manipulation task on cloths with different elasticity and weight values. In both tasks, the cloth starts free of contact, grasped by two manipulators, and is placed on a flat surface. In the dynamic task, the trajectory consists of a dynamic *fling* motion of the fabric. In the quasi-static task, the trajectory brings the garment in contact with the rigid surface and then drags it through the planar surface. The design of these tasks aimed at measuring the reality gap by addressing two properties challenging to simulate (see Chapter 3): the switching between a contact state and a contact-free state, as well as the large deformation resulting from dynamic manipulation. The deformation of the cloth was extracted from RGB-D data, which is first pre-processed to filter points that are not part of the garment, filter outliers, and then aligned with the coordinate system of the simulator. The gap is evaluated by measuring the distance between the simulator meshes and the data set point cloud. Following the pipeline discussed in the literature of Section 4.1.2, we first optimise the parameters of the simulator by using the average Chamfer Distance (CD) as a minimisation objective. This optimisation was performed using Bayesian Optimisation (BO) [194] and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [195], selecting the parameters that achieve the minimum distance from the two methods. Then, the gap was quantified by computing the Chamfer and Hausdorff distances at each instant of the fling motion. In this work, four simulators were evaluated: MuJoCo, Bullet, Flex, and SOFA. Along with the quantitative results, qualitative results were provided showing the simulated and real cloths at different time steps of the motion. The results showed similar performance in all simulation engines for the quasi-static task, where the qualitative errors were low and the simulators were able to closely match the real data. In the dynamic task MuJoCo and SOFA presented relatively low errors and exhibited the real cloth's dynamic motion. By contrast, Flex presented larger errors and was not able to match the motion of the real cloths, while Bullet presented an unstable simulation despite the system identification process. Overall, these results indicated that the largest reality gap is on dynamic manipulation, where MuJoCo and SOFA performed. However, the reality gap

remains in all the evaluated simulators despite the system identification process.

4.4 Conclusions

This chapter explored the reality gap in deformable objects, focusing on techniques used to quantify and bridge this gap. Within the methods used to close the gap, the chapter highlighted the widespread use of domain randomisation. Although related works have included randomisation of the visual and object properties, they often overlook parameters such as friction. One possible reason for this is that the simulation engines used have limits on the parameters that can be modified. Therefore, a critical initial step in bridging the sim-to-real gap is the selection of an appropriate simulator. This decision should be based on the physics phenomena of the manipulation task to be tackled and the capabilities of the chosen simulator.

Learning low-level control visual feedback policies presents several benefits for adapting the manipulation actions to different materials, as discussed in Chapter 3. However, transferring these policies to the real-world is challenging if the reality gap is large. In order to reduce the effect of this gap, similar to Publication IV, future work should take into account the temporal delay [196] to facilitate the transfer of these policies.

Another issue discussed in this chapter is the lack of consistency among the existing studies concerning the same manipulation tasks. By using different simulation engines, data sets, and performance metrics, the results over different state-of-the-art methods are not comparable. This could be circumvented by including a measurement of the reality gap along with task performance metrics. A promising direction for future work would be to evaluate simulation engines with a broad range of deformable objects and materials. This could serve as a reference for learning-based approaches for selecting the simulation engine and performing system identification. In addition, the open sourcing of these data sets could benefit simulation-based approaches for closing the gap, further facilitating a smooth zero-shot transfer, similar to those available in rigid object manipulation tasks [29, 197, 198].

5. Conclusion

Over the last decade, considerable progress has been made in robotic manipulation of deformable objects. Playing a leading role in this development, data-driven approaches along with simulation engines have bypassed some challenges in learning to manipulate these objects in the real-world, such as sensing object deformation. However, the current approaches are still far from enabling robotic systems to manipulate a broad range of objects and materials, such as the variety of clothes and fabrics found in retail shops. This thesis has addressed the problem of deformable object manipulation by focusing on two key aspects: adaptability (enabling manipulation of diverse shapes, colours, and materials) and efficiency (both in terms of the interactions with the real-world required to solve manipulation tasks and learning robot policies). In this dissertation, deformable object manipulation has been explored using data-driven approaches and dividing this problem into three separate sub-problems: representation and dynamics modelling, manipulation, and the sim-to-real gap.

To model the dynamics of deformable objects for manipulation, the thesis has discussed two types of data-driven approaches in terms of their representation: image-based and graph-based methods. These approaches involve learning either forward or inverse dynamics models, which are then leveraged for planning the appropriate manipulation actions. The image representation enables straightforward deployment in the real world due to the robot sensing systems. By contrast, dynamics models learnt using graph representations–which can represent deformable objects as particles-–provide higher accuracy, as they better resemble the physics engines in which these models are trained. Furthermore, the graph representation is more intuitive for various deformable objects, such granular media as beans or ground coffee. For this reason, Publication I and Publication II used a graph representation for learning the dynamics. These publications revealed two gaps in the related literature. First, the learnt models do not quantify the uncertainty of their predictions, which was addressed in Publication I by proposing a data-efficient method based on graph Gaussian Processes. Second, existing studies on granular media manipulation are unable to capture the underlying physical interactions between the grains. Hence, Publication II proposed learning a surrogate Graph Neural Network dynamics model

of granular materials for manipulation planning. The benefit of this approach was demonstrated for learning the grain interactions in the task of pouring the material to form different shapes. Both Publication I and Publication II demonstrated that using a graph representation enables *efficient* learning of deformable object dynamics in simulation, thus facilitating their manipulation in the real world and addressing **RQ 1**.

Furthermore, the thesis has provided an overview of techniques for manipulating deformable objects while analysing their capabilities to adapt to diverse material properties and shapes. Here, the benefit of dynamic manipulation was highlighted for solving tasks involving deformable manipulation with just a few trials, despite the challenges posed when handling diverse types of materials. While methods that utilise dynamics models for action planning can find optimal solutions–given that the surrogate model accurately reflects the real behaviour of the object, they exhibit limited generalisation to unseen objects and materials. Therefore, the thesis has focused on approaches that train policies from data gathered in simulation. Within this context, two open problems were examined. First, the methods that perform high-level planning use primitives with fixed parameters, such as the primitive's velocity, thereby neglecting their impact on task performance. To overcome this limitation, Publication III proposed optimising the parameters of quasi-static and dynamic manipulation primitives following a sequential decision process. The results show that optimising these parameters could facilitate handling a diverse range of cloth materials and shapes. The second issue identified in the existing literature concerns the use of open-loop control policies, which cannot adapt to deformation changes during the manipulation process. To this end, Publication IV proposed learning a closed-loop visual feedback policy for dynamic cloth manipulation. Our results demonstrate that the use of closed-loop feedback can enable the robot to adapt its actions to the deformation of various cloth materials in a dynamic task. Publication III and Publication IV resulted in the development of methods that can *adapt* the manipulation actions, as well as utilise dynamic manipulation actions to solve the manipulation tasks within several trials, thus improving *efficiency* and addressing **RQ 2**. A potential direction for future research would be to integrate closed-loop feedback with high-level planning using graph-based dynamics models. This approach could bring out the advantages of model-based planning, which can find solutions that are near to optimal and thus bypass the unexpected behaviour when these models fail to model real phenomena with the closed-loop control system.

Finally, analysis of the advantages and disadvantages of data-driven approaches learnt in simulation led to the third issue, which has been evident throughout this work, the reality gap. This crucial task of bridging the gap between simulation and the real world has been explored from two perspectives. First, the thesis examined the technique of domain randomisation, which was used in both Publication III and Publication IV for learning policies in simulation. The results show that although domain randomisation helps to generalise the

policies across different shapes and materials, the gap is considerably smaller when transferring closed-loop policies to the real world. Second, the thesis assessed the existing research on measuring and evaluating the disparity between the actual and expected outcomes of manipulating deformable objects. Given the difficulty in simulating deformable objects under dynamic manipulation, Publication V introduced a benchmark data set for dynamic cloth manipulation and then evaluated the gap in four open-source simulators. Together, Publications III, IV and V demonstrate that system identification and domain randomisation alone are insufficient to close the gap, thus highlighting the necessity of closed-loop control to mitigate the impact in the real world, as well as addressing **RQ 3**. These findings strongly suggest that a beneficial direction for robotic assistance research would be to explore the reality gap in tasks that also involve the forces applied by the robot when, for example, interfacing with a human. This could enable learning tasks that involve human-robot contact, where the learning occurs safely in simulation. Additionally, open-source data sets would be highly valuable for fine-tuning simulators, along with the simulation environments, when performing system identification.

All in all, this dissertation has investigated data-driven methods to efficiently manipulate deformable objects by leveraging simulation engines. In a world where robots have the potential to assist humans in caregiving tasks, advances in robot learning offer great potential for adapting to the variability of dynamic human environments. Hopefully, this work will encourage roboticists to consider simulation-based learning approaches to improve the adaptability of the manipulation systems, bringing us one step closer to bridging the human-robot manipulation gap.

References

[1] Maria Robinson. *Child Development From Birth To Eight: A Journey Through The Early Years.* McGraw-Hill Education (UK), 2007.

[2] J.Gavin Bremner. Developmental relationships between perception and action in infancy. *Infant Behavior and Development*, 23(3):567–582, 2000.

[3] David M. Huberdeau, John W. Krakauer, and Adrian M. Haith. Practice induces a qualitative change in the memory representation for visuomotor learning. *Journal of Neurophysiology*, 122(3):1050–1059, 2019. PMID: 31389741.

[4] Adrian M Haith and John W Krakauer. The multiple effects of practice: skill, habit and reduced cognitive load. *Current Opinion in Behavioral Sciences*, 20:196–201, 2018. Habits and Skills.

[5] Jihong Zhu, Andrea Cherubini, Claire Dune, David Navarro-Alarcon, Farshid Alambeigi, Dmitry Berenson, Fanny Ficuciello, Kensuke Harada, Jens Kober, Xiang Li, Jia Pan, Wenzhen Yuan, and Michael Gienger. Challenges and outlook in robotic manipulation of deformable objects. *IEEE Robotics & Automation Magazine*, 29(3):67–77, 2022.

[6] Zixuan Huang, Xingyu Lin, and David Held. Mesh-based Dynamics with Occlusion Reasoning for Cloth Manipulation. In *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.

[7] Zixuan Huang, Xingyu Lin, and David Held. Self-supervised cloth reconstruction via action-conditioned cloth tracking. *arXiv preprint arXiv:2302.09502*, 2023.

[8] Youngsun Wi, Andy Zeng, Pete Florence, and Nima Fazeli. Virdo++: Real-world, visuo-tactile dynamics and perception of deformable objects. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1806–1816. PMLR, 14–18 Dec 2023.

[9] Pol Monsó, Guillem Alenyà, and Carme Torras. Pomdp approach to robotized clothes separation. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1324–1329, 2012.

[10] Jan Matas, Stephen James, and Andrew J Davison. Sim-to-real reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 734–743. PMLR, 2018.

[11] Xingyu Lin, Yufei Wang, Zixuan Huang, and David Held. Learning visible connectivity dynamics for cloth smoothing. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 256–266. PMLR, 08–11 Nov 2022.

[12] Xiao Ma, David Hsu, and Wee Sun Lee. Learning latent graph dynamics for visual manipulation of deformable objects. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8266–8273, 2022.

[13] Halid Abdulrahim Kadi and Kasim Terzić. Data-driven robotic manipulation of cloth-like deformable objects: The present, challenges and future prospects. *Sensors*, 23(5), 2023.

[14] Jose Sanchez, Juan-Antonio Corrales, Belhassen-Chedli Bouzgarrou, and Youcef Mezouar. Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey. *The International Journal of Robotics Research*, 37(7):688–716, 2018.

[15] Isabella Huang, Yashraj Narang, Ruzena Bajcsy, Fabio Ramos, Tucker Hermans, and Dieter Fox. Defgraspnets: Grasp planning on 3d fields with graph neural nets. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5894–5901, 2023.

[16] Liman Wang and Jihong Zhu. Deformable object manipulation in caregiving scenarios: A review. *Machines*, 11(11), 2023.

[17] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 540–562. PMLR, 06–09 Nov 2023.

[18] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[19] Veronica E. Arriola-Rios, Puren Guler, Fanny Ficuciello, Danica Kragic, Bruno Siciliano, and Jeremy L. Wyatt. Modeling of deformable objects for robotic manipulation: A tutorial and review. *Frontiers in Robotics and AI*, 7, 2020.

[20] Y.F. Zheng, R. Pei, and C. Chen. Strategies for automatic assembly of deformable objects. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, pages 2598–2603 vol.3, 1991.

[21] T. Wada, S. Hirai, and S. Kawamura. Indirect simultaneous positioning operations of extensionally deformable objects. In *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190)*, volume 2, pages 1333–1338 vol.2, 1998.

[22] S. Tokumoto, Y. Fujita, and S. Hirai. Deformation modeling of viscoelastic objects for their shape control. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 1, pages 767–772 vol.1, 1999.

[23] Simon Duenser, James M. Bern, Roi Poranne, and Stelian Coros. Interactive robotic manipulation of elastic objects. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3476–3481, 2018.

[24] A. Koessler, N. Roca Filella, B.C. Bouzgarrou, L. Lequièvre, and J.-A. Corrales Ramon. An efficient approach to closed-loop shape control of deformable objects using finite element models. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1637–1643, 2021.

[25] Fei Liu, Zihan Li, Yunhai Han, Jingpei Lu, Florian Richter, and Michael C. Yip. Real-to-sim registration of deformable soft tissue with position-based dynamics for surgical robot autonomy. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12328–12334, 2021.

[26] Fei Liu, Entong Su, Jingpei Lu, Mingen Li, and Michael C. Yip. Robotic manipulation of deformable rope-like objects using differentiable compliant position-based dynamics. *IEEE Robotics and Automation Letters*, pages 1–8, 2023.

[27] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8459–8468. PMLR, 13–18 Jul 2020.

[28] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

[29] Jack Collins, David Howard, and Jurgen Leitner. Quantifying the reality gap in robotic manipulation tasks. In *2019 International Conference on Robotics and Automation*, pages 6706–6712, 2019.

[30] Matthew T. Mason. *Mechanics of Robotic Manipulation*. MIT Press, Cambridge, MA, USA, 2001.

[31] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4):1307–1319, 2020.

[32] Huy Ha and Shuran Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. In *Conference on Robotic Learning (CoRL)*, 2021.

[33] Zhenjia Xu, Cheng Chi, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. DextAIRity: Deformable Manipulation Can be a Breeze. In *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.

[34] Youngsun Wi, Pete Florence, Andy Zeng, and Nima Fazeli. Virdo: Visio-tactile implicit representations of deformable objects. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3583–3590, 2022.

[35] Gang Yao, Ryan Saltus, and Ashwin P Dani. Shape estimation for elongated deformable object using b-spline chained multiple random matrices model. *International journal of intelligent robotics and applications*, 4:429–440, 2020.

[36] Bohan Yang, Bo Lu, Wei Chen, Fangxun Zhong, and Yun-Hui Liu. Model-free 3-d shape control of deformable objects using novel features based on modal analysis. *IEEE Transactions on Robotics*, 39(4):3134–3153, 2023.

[37] Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. Learning predictive representations for deformable objects using contrastive estimation. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 564–574. PMLR, 16–18 Nov 2021.

[38] Connor Schenck, Jonathan Tompson, Sergey Levine, and Dieter Fox. Learning robotic manipulation of granular media. In *Conference on Robot Learning*, pages 239–248. PMLR, 2017.

[39] Yunbo Zhang, Wenhao Yu, C. Karen Liu, Charlie Kemp, and Greg Turk. Learning to manipulate amorphous materials. *ACM Transactions on Graphics*, 39, 2020.

[40] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

[41] François Faure, Christian Duriez, Hervé Delingette, Jérémie Allard, Benjamin Gilles, Stéphanie Marchesseau, Hugo Talbot, Hadrien Courtecuisse, Guillaume Bousquet, Igor Peterlik, et al. Sofa: A multi-model framework for interactive physical simulation. In *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, pages 283–321. Springer, 2012.

[42] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4470–4479, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[43] Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li Fei-Fei, Joshua B. Tenenbaum, and Daniel L. K. Yamins. Flexible neural representation for physics prediction. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 8813–8824, Red Hook, NY, USA, 2018. Curran Associates Inc.

[44] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, page 141–151, Goslar, DEU, 2004. Eurographics Association.

[45] Bart Adams, Mark Pauly, Richard Keiser, and Leonidas J. Guibas. Adaptively sampled particle fluids. *ACM Trans. Graph.*, 26(3):48–es, jul 2007.

[46] Barbara Solenthaler, Jürg Schläfli, and Renato Pajarola. A unified particle model for fluid–solid interactions. *Computer Animation and Virtual Worlds*, 18(1):69–82, 2007.

[47] Jung-Tae Kim, Fabio Ruggiero, Vincenzo Lippiello, and Bruno Siciliano. *Smoothed Particle Hydrodynamics-Based Viscous Deformable Object Modelling*, pages 73–102. Springer International Publishing, Cham, 2022.

[48] Gilles Daviet and Florence Bertails-Descoubes. A semi-implicit material point method for the continuum simulation of granular materials. *ACM Trans. Graph.*, 35(4), July 2016.

[49] Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*, SIGGRAPH '16, New York, NY, USA, 2016. Association for Computing Machinery.

[50] Luca De Luigi, Ren Li, Benoît Guillard, Mathieu Salzmann, and Pascal Fua. Drapenet: Garment generation and self-supervised draping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1451–1460, June 2023.

[51] Han Xue, Wenqiang Xu, Jieyi Zhang, Tutian Tang, Yutong Li, Wenxin Du, Ruolin Ye, and Cewu Lu. Garmenttracking: Category-level garment pose tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21233–21242, June 2023.

[52] Fangzhou Hong, Liang Pan, Zhongang Cai, and Ziwei Liu. Garment4d: Garment reconstruction from point cloud sequences. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 27940–27951. Curran Associates, Inc., 2021.

[53] Tran Nguyen Le, Jens Lundell, Fares J. Abu-Dakka, and Ville Kyrki. A novel simulation-based quality metric for evaluating grasps on 3d deformable objects. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3123–3129, 2022.

[54] Tran Nguyen Le, Fares J. Abu-Dakka, and Ville Kyrki. Sponge: Sequence planning with deformable-on-rigid contact prediction from geometric features. *arXiv:2303.14012*, 2023. arXiv: 2303.14012.

[55] Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.

[56] Patricia Moore and Derek Molloy. A survey of computer-based deformable models. In *International Machine Vision and Image Processing Conference (IMVIP 2007)*, pages 55–66, 2007.

[57] Jianping Cai, Feng Lin, and Hock Soon Seah. *Graphical Simulation of Deformable Models*. Springer International Publishing, 2016.

[58] Xavier Provot. Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In *Proceedings of Graphics Interface '95*, GI '95, pages 147–154, Toronto, Ontario, Canada, 1995. Canadian Human-Computer Communications Society.

[59] Dongliang Tan, Jiashi Zhao, Weili Shi, Xingzhi Li, Huamin Yang, and Zhengang Jiang. An improved soft tissue deformation simulation model based on mass spring. In *2020 International Conference on Virtual Reality and Visualization (ICVRV)*, pages 121–127, 2020.

[60] Hongly Va, Min-Hyung Choi, and Min Hong. Real-time surface-based volume constraints on mass-spring model in unity3d. *IEEE Access*, 11:17857–17869, 2023.

[61] Satoshi Ishikawa, Katsunori Tanaka, Daiki Yano, and Shinya Kijimoto. Design of a disc-shaped viscoelastic damping material attached to a cylindrical pipe as a dynamic absorber or houde damper. *Journal of Sound and Vibration*, 475:115272, 2020.

[62] J. Pinho da Cruz and F. Teixeira-Dias. On the optimisation of viscoplastic constitutive modelling using a numerical feedback damping algorithm. *Computer Methods in Applied Mechanics and Engineering*, 194(18):2191–2210, 2005.

[63] John C. Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, Hoboken, New Jersey, third edition, 2016.

[64] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer series in operations research and financial engineering. Springer, New York, NY, 2. ed. edition, 2006.

[65] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[66] Adrià Luque, David Parent, Adrià Colomé, Carlos Ocampo-Martinez, and Carme Torras. Model predictive control for dynamic cloth manipulation: Parameter learning and experimental validation. *arXiv preprint arXiv:2209.05798*, 2022.

[67] Veronica E. Arriola-Rios and Jeremy Wyatt. 2d mass-spring-like model for prediction of a sponge's behaviour upon robotic interaction. In Max Bramer, Miltos Petridis, and Lars Nolle, editors, *Research and Development in Intelligent Systems XXVIII*, pages 195–208, London, 2011. Springer London.

[68] Domen Mongus, Blaz Repnik, Marjan Mernik, and B $\mathbf{v}$Zalik. A hybrid evolutionary algorithm for tuning a cloth-simulation model. *Applied Soft Computing*, 12(1):266–273, 2012.

[69] M.B. Rubin and I. Einav. A large deformation breakage model of granular materials including porosity and inelastic distortional deformation rate. *International Journal of Engineering Science*, 49(10):1151–1169, 2011.

[70] Ruofeng Feng, Georgios Fourtakas, Benedict D. Rogers, and Domenico Lombardi. Large deformation analysis of granular materials with stabilized and noise-free stress treatment in smoothed particle hydrodynamics (sph). *Computers and Geotechnics*, 138:104356, 2021.

[71] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.

[72] Hang Yin, Anastasia Varava, and Danica Kragic. Modeling, learning, perception, and control methods for deformable object manipulation. *Science Robotics*, 6(54):eabd8803, 2021.

[73] Junheng Fang, Lihua You, Ehtzaz Chaudhry, and Jian Zhang. State-of-the-art improvements and applications of position based dynamics. *Computer Animation and Virtual Worlds*, 34(5):e2143, 2023.

[74] Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*, SIGGRAPH '16, New York, NY, USA, 2016. Association for Computing Machinery.

[75] Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)*, 37(4):150, 2018.

[76] Niels Saabye Ottosen and Matti Ristinmaa. *The mechanics of constitutive modeling*. Elsevier, 2005.

[77] Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jerome Parent-Levesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler. gradsim: Differentiable simulation for system identification and visuomotor control. *International Conference on Learning Representations (ICLR)*, 2021.

[78] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B. Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[79] Kelsey R Allen, Tatiana Lopez Guevara, Yulia Rubanova, Kim Stachenfeld, Alvaro Sanchez-Gonzalez, Peter Battaglia, and Tobias Pfaff. Graph network simulators can learn discontinuous, rigid contact dynamics. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1157–1167. PMLR, 14–18 Dec 2023.

[80] David Blanco-Mulero, Markus Heinonen, and Ville Kyrki. Evolving-graph Gaussian processes. In *Time Series Workshop at 34th International Conference on Machine Learning.*, 2021.

[81] Neea Tuomainen, David Blanco-Mulero, and Ville Kyrki. Manipulation of granular materials by learning particle interactions. *IEEE Robotics and Automation Letters*, 7(2):5663–5670, 2022.

[82] Yasmin Salehi and Dennis Giannacopoulos. Physgnn: A physics–driven graph neural network based model for predicting soft tissue deformation in image–guided neurosurgery. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 37282–37296. Curran Associates, Inc., 2022.

[83] Jiaqi Han, Wenbing Huang, Hengbo Ma, Jiachen Li, Josh Tenenbaum, and Chuang Gan. Learning physical dynamics with subequivariant graph neural networks. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 26256–26268. Curran Associates, Inc., 2022.

[84] Haochen Shi, Huazhe Xu, Zhiao Huang, Yunzhu Li, and Jiajun Wu. RoboCraft: Learning to See, Simulate, and Shape Elasto-Plastic Objects with Graph Networks. In *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.

[85] Changhao Wang, Yuyou Zhang, Xiang Zhang, Zheng Wu, Xinghao Zhu, Shiyu Jin, Te Tang, and Masayoshi Tomizuka. Offline-online learning of deformation model for cable manipulation with graph neural networks. *IEEE Robotics and Automation Letters*, 7(2):5544–5551, 2022.

[86] Xiao Ma, David Hsu, and Wee Sun Lee. Learning latent graph dynamics for visual manipulation of deformable objects. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8266–8273, 2022.

[87] Yufei Wang, David Held, and Zackory Erickson. Visual haptic reasoning: Estimating contact forces by observing deformable object interactions. *IEEE Robotics and Automation Letters*, 7(4):11426–11433, 2022.

[88] Yuhong Deng, Chongkun Xia, Xueqian Wang, and Lipeng Chen. Deep reinforcement learning based on local gnn for goal-conditioned deformable object rearranging. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1131–1138, 2022.

[89] Danny Driess, Zhiao Huang, Yunzhu Li, Russ Tedrake, and Marc Toussaint. Learning multi-object dynamics with compositional neural radiance fields. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1755–1768. PMLR, 14–18 Dec 2023.

[90] Alberta Longhini, Marco Moletta, Alfredo Reichlin, Michael C. Welle, David Held, Zackory Erickson, and Danica Kragic. Edo-net: Learning elastic properties of deformable objects from graph dynamics. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3875–3881, 2023.

[91] Ryan Hoque, Daniel Seita, Ashwin Balakrishna, Aditya Ganapathi, Ajay Tanwani, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[92] Daisuke Tanaka, Solvi Arnold, and Kimitoshi Yamazaki. Emd net: An encode–manipulate–decode network for cloth manipulation. *IEEE Robotics and Automation Letters*, 3(3):1771–1778, 2018.

[93] Halid Abdulrahim Kadi and Kasim Terzić. Data-driven robotic manipulation of cloth-like deformable objects: The present, challenges and future prospects. *Sensors*, 23(5), 2023.

[94] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2146–2153, 2017.

[95] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[96] Angelina Wang, Thanard Kurutach, Kara Liu, Pieter Abbeel, and Aviv Tamar. Learning robotic manipulation through visual planning and acting. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019.

[97] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

[98] Stephen James, Andrew J. Davison, and Edward Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 334–343. PMLR, 13–15 Nov 2017.

[99] Sirui Chen, Yunhao Liu, Shang Wen Yao, Jialong Li, Tingxiang Fan, and Jia Pan. Diffsrl: Learning dynamical state representation for deformable object manipulation with differentiable simulation. *IEEE Robotics and Automation Letters*, 7(4):9533–9540, 2022.

[100] Carolyn Matl, Yashraj Narang, Ruzena Bajcsy, Fabio Ramos, and Dieter Fox. Inferring the material properties of granular media for robotic tasks. 2020.

[101] Peter Mitrano, Alex LaGrassa, Oliver Kroemer, and Dmitry Berenson. Focused adaptation of dynamics models for deformable object manipulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5931–5937, 2023.

[102] P. Mitrano, D. McConachie, and D. Berenson. Learning where to trust unreliable models in an unstructured world for deformable object manipulation. *Science Robotics*, 6(54):eabd8170, 2021.

[103] Cheng-Yu Kuo, Andreas Schaarschmidt, Yunduan Cui, Tamim Asfour, and Takamitsu Matsubara. Uncertainty-aware contact-safe model-based reinforcement learning. *IEEE Robotics and Automation Letters*, 6(2):3918–3925, 2021.

[104] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[105] Juvs Kocijan. *Modelling and control of dynamic systems using Gaussian process models*. Springer, 2016.

[106] Viacheslav Borovitskiy, Iskander Azangulov, Alexander Terenin, Peter Mostowsky, Marc Deisenroth, and Nicolas Durrande. Matérn Gaussian processes on graphs. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2593–2601. PMLR, 13–15 Apr 2021.

[107] Felix L. Opolka and Pietro Liò. Graph Convolutional Gaussian Processes For Link Prediction. In *Workshop on Graph Representation Learning and Beyond (GRL+)*, 2020.

[108] A. Venkitaraman, S. Chatterjee, and P. Handel. Gaussian processes over graphs. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5640–5644, 2020.

[109] Yin-Cong Zhi, Yin Cheng Ng, and Xiaowen Dong. Gaussian Processes on Graphs via Spectral Kernel Learning. *arXiv:2006.07361 [cs, eess, stat]*, October 2020. arXiv: 2006.07361.

[110] Nils M Kriege, Fredrik D Johansson, and Christopher Morris. A survey on graph kernels. *Applied Network Science*, 5(1):1–42, 2020.

[111] Andrea Cherubini, Jürgen Leitner, Valerio Ortenzi, and Peter Corke. Towards vision-based manipulation of plastic materials. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 485–490, 2018.

[112] Carolyn Matl, Yashraj Narang, Ruzena Bajcsy, Fabio Ramos, and Dieter Fox. Inferring the material properties of granular media for robotic tasks. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2770–2777, 2020.

[113] Vincent Lim, Huang Huang, Lawrence Yunliang Chen, Jonathan Wang, Jeffrey Ichnowski, Daniel Seita, Michael Laskey, and Ken Goldberg. Real2sim2real: Self-supervised learning of physical single-step dynamic actions for planar robot casting. In *2022 International Conference on Robotics and Automation*, pages 8282–8289, 2022.

[114] Peng Chang and Taşkin Padif. Sim2real2sim: Bridging the gap between simulation and real-world in flexible object manipulation. In *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*, pages 56–62, 2020.

[115] Arman Hasanzadeh, Ehsan Hajiramezanali, Shahin Boluki, Mingyuan Zhou, Nick Duffield, Krishna Narayanan, and Xiaoning Qian. Bayesian graph neural networks with adaptive connection sampling. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4094–4104. PMLR, 13–18 Jul 2020.

[116] Maximilian Stadler, Bertrand Charpentier, Simon Geisler, Daniel Zügner, and Stephan Günnemann. Graph posterior network: Bayesian predictive uncertainty for node classification. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18033–18048. Curran Associates, Inc., 2021.

[117] Çagatay Yıldız, Melih Kandemir, and Barbara Rakitsch. Learning interacting dynamical systems with latent Gaussian process odes. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 9188–9200. Curran Associates, Inc., 2022.

[118] Rui Wang and Rose Yu. Physics-guided deep learning for dynamical systems: A survey. *arXiv preprint arXiv:2107.01272*, 2021.

[119] Priya Sundaresan, Rika Antonova, and Jeannette Bohg. Diffcloud: Real-to-sim from point clouds with differentiable simulation and rendering of deformable objects. In *arXiv preprint arXiv:2204.03139*, 2022.

[120] Rishabh Jangir, Guillem Alenyà, and Carme Torras. Dynamic cloth manipulation with deep reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4630–4636, 2020.

[121] N. Fahantidis, K. Paraschidis, V. Petridis, Z. Doulgeri, L. Petrou, and G. Hasapis. Robot handling of flat textile materials. *IEEE Robotics & Automation Magazine*, 4(1):34–41, 1997.

[122] Yinxiao Li, Yonghao Yue, Danfei Xu, Eitan Grinspun, and Peter K. Allen. Folding deformable objects using predictive simulation and trajectory optimization. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6000–6006, 2015.

[123] Andrea Cherubini, Valerio Ortenzi, Akansel Cosgun, Robert Lee, and Peter Corke. Model-free vision-based shaping of deformable plastic materials. *International Journal of Robotics Research*, 39, 2020.

[124] Yixuan Wang, Yunzhu Li, Katherine Driggs-Campbell, Li Fei-Fei, and Jiajun Wu. Dynamic-Resolution Model Learning for Object Pile Manipulation. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.

[125] Fabio Ruggiero, Vincenzo Lippiello, and Bruno Siciliano. Nonprehensile dynamic manipulation: A survey. *IEEE Robotics and Automation Letters*, 3(3):1711–1718, 2018.

[126] Simon Zimmermann, Roi Poranne, and Stelian Coros. Dynamic manipulation of deformable objects with implicit integration. *IEEE Robotics and Automation Letters*, 6(2):4209–4216, 2021.

[127] Cheng Chi, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Iterative Residual Policy for Goal-Conditioned Dynamic Manipulation of Deformable Objects. In *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.

[128] Fabio Ruggiero, Jung-Tae Kim, Alejandro Gutierrez-Giles, Aykut C. Satici, Alejandro Donaire, Jonathan Cacace, Luca Rosario Buonocore, Giuseppe Andrea Fontanelli, Vincenzo Lippiello, and Bruno Siciliano. Nonprehensile manipulation control and task planning for deformable object manipulation: Results from the rodyman project. In Oleg Gusikhin and Kurosh Madani, editors, *Informatics in Control, Automation and Robotics*, pages 76–100, Cham, 2020. Springer International Publishing.

[129] Harry Zhang, Jeffrey Ichnowski, Daniel Seita, Jonathan Wang, Huang Huang, and Ken Goldberg. Robots of the lost arc: Self-supervised learning to dynamically manipulate fixed-endpoint cables. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4560–4567, 2021.

[130] Stuart Miller and Roger Bartlett. The relationship between basketball shooting kinematics, distance and playing position. *Journal of Sports Sciences*, 14(3):243–253, 1996. PMID: 8809716.

[131] Kevin M. Lynch and Matthew T. Mason. Dynamic nonprehensile manipulation: Controllability, planning, and experiments. *The International Journal of Robotics Research*, 18(1):64–92, 1999.

[132] Yahav Avigal, Lars Berscheid, Tamim Asfour, Torsten Kröger, and Ken Goldberg. Speedfolding: Learning efficient bimanual folding of garments. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8, 2022.

[133] Olivia Nocentini, Jaeseok Kim, Zain Muhammad Bashir, and Filippo Cavallo. Learning-based control approaches for service robots on cloth manipulation and dressing assistance: a comprehensive review. *Journal of NeuroEngineering and Rehabilitation*, 19(1):1–25, 2022.

[134] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 4. Athena scientific, 2012.

[135] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[136] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

[137] Yuji Yamakawa, Akio Namiki, and Masatoshi Ishikawa. Dynamic manipulation of a cloth by high-speed robot system using high-speed visual feedback. *IFAC Proceedings Volumes*, 44(1):8076–8081, 2011. 18th IFAC World Congress.

[138] P. Jiménez. Survey on model-based manipulation planning of deformable objects. *Robotics and Computer-Integrated Manufacturing*, 28(2):154–163, 2012.

[139] Timothy Bretl and Zoe McCarthy. Quasi-static manipulation of a kirchhoff elastic rod based on a geometric analysis of equilibrium configurations. *The International Journal of Robotics Research*, 33(1):48–68, 2014.

[140] Eiichi Yoshida, Ko Ayusawa, Ixchel G. Ramirez-Alpizar, Kensuke Harada, Christian Duriez, and Abderrahmane Kheddar. Simulation-based optimal motion planning for deformable object. In *2015 IEEE International Workshop on Advanced Robotics and its Social Impacts (ARSO)*, pages 1–6, 2015.

[141] Huan Lin, Feng Guo, Feifei Wang, and Yan-Bin Jia. Picking up a soft 3d object by "feeling" the grip. *The International Journal of Robotics Research*, 34(11):1361–1384, 2015.

[142] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[143] Martina Lippi, Petra Poklukar, Michael C. Welle, Anastasiia Varava, Hang Yin, Alessandro Marino, and Danica Kragic. Latent space roadmap for visual action planning of deformable and rigid object manipulation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5619–5626, 2020.

[144] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 432–448. PMLR, 16–18 Nov 2021.

[145] Alexander Clegg, Zackory Erickson, Patrick Grady, Greg Turk, Charles C. Kemp, and C. Karen Liu. Learning to collaborate from simulation for robot-assisted dressing. *IEEE Robotics and Automation Letters*, 5(2):2746–2753, 2020.

[146] Rita Laezza, Robert Gieselmann, Florian T. Pokorny, and Yiannis Karayiannidis. Reform: A robot learning sandbox for deformable linear object manipulation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4717–4723, 2021.

[147] Yafei Ou and Mahdi Tavakoli. Sim-to-real surgical robot learning and autonomous planning for internal tissue points manipulation using reinforcement learning. *IEEE Robotics and Automation Letters*, 8(5):2502–2509, 2023.

[148] Robert Lee, Daniel Ward, Vibhavari Dasagi, Akansel Cosgun, Juxi Leitner, and Peter Corke. Learning arbitrary-goal fabric folding with one hour of real robot experience. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 2317–2327. PMLR, 16–18 Nov 2021.

[149] Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[150] Matthias Rambow, Thomas Schauß, Martin Buss, and Sandra Hirche. Autonomous manipulation of deformable objects based on teleoperated demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2809–2814, 2012.

[151] Alex X. Lee, Sandy H. Huang, Dylan Hadfield-Menell, Eric Tzeng, and Pieter Abbeel. Unifying scene registration and trajectory optimization for learning from demonstrations with application to manipulation of deformable objects. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4402–4407, 2014.

[152] Alex X. Lee, Henry Lu, Abhishek Gupta, Sergey Levine, and Pieter Abbeel. Learning force-based manipulation of deformable objects from multiple demonstrations. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 177–184, 2015.

[153] Michael Laskey, Chris Powers, Ruta Joshi, Arshan Poursohi, and Ken Goldberg. Learning robust bed making using deep imitation learning with dart. *arXiv preprint arXiv:1711.02525*, 2017.

[154] Daniel Seita, Nawid Jamali, Michael Laskey, Ajay Kumar Tanwani, Ron Berenstein, Prakash Baskaran, Soshi Iba, John Canny, and Ken Goldberg. Deep Transfer Learning of Pick Points on Fabric for Robot Bed-Making. In *International Symposium on Robotics Research (ISRR)*, 2019.

[155] Biao Jia, Zherong Pan, Zhe Hu, Jia Pan, and Dinesh Manocha. Cloth manipulation using random-forest-based imitation learning. *IEEE Robotics and Automation Letters*, 4(2):2086–2093, 2019.

[156] Daniel Seita, Aditya Ganapathi, Ryan Hoque, Minho Hwang, Edward Cen, Ajay Kumar Tanwani, Ashwin Balakrishna, Brijen Thananjeyan, Jeffrey Ichnowski, Nawid Jamali, Katsu Yamane, Soshi Iba, John Canny, and Ken Goldberg. Deep imitation learning of sequential fabric smoothing from an algorithmic supervisor. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9651–9658, 2020.

[157] Benjamin Balaguer and Stefano Carpin. Combining imitation and reinforcement learning to fold deformable planar objects. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1405–1412, 2011.

[158] Gautam Salhotra, I-Chun Arthur Liu, Marcus Dominguez-Kuhne, and Gaurav S. Sukhatme. Learning deformable object manipulation from expert demonstrations. *IEEE Robotics and Automation Letters*, 7(4):8775–8782, 2022.

[159] Yevgen Chebotar, Karol Hausman, Marvin Zhang, Gaurav Sukhatme, Stefan Schaal, and Sergey Levine. Combining model-based and model-free updates for trajectory-centric reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 703–711. PMLR, 06–11 Aug 2017.

[160] Jihong Zhu, Michael Gienger, Giovanni Franzese, and Jens Kober. Do you need a hand?–a bimanual robotic dressing assistance scheme. *arXiv e-prints*, pages arXiv–2301, 2023.

[161] Thomas Weng, Sujay Man Bajracharya, Yufei Wang, Khush Agrawal, and David Held. Fabricflownet: Bimanual cloth manipulation with a flow-based policy. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Conference on Robot Learning, 8-11 November 2021, London, UK*, volume 164 of *Proceedings of Machine Learning Research*, pages 192–202. PMLR, 2021.

[162] Alper Canberk, Cheng Chi, Huy Ha, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Cloth funnels: Canonicalized-alignment for multi-purpose garment manipulation. *arXiv preprint arXiv:2210.09347*, 2022.

[163] Fan Zhang and Yiannis Demiris. Visual-tactile learning of garment unfolding for robot-assisted dressing. *IEEE Robotics and Automation Letters*, 8(9):5512–5519, 2023.

[164] Arpit Bahety, Shreeya Jain, Huy Ha, Nathalie Hager, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Bag all you need: Learning a generalizable bagging strategy for heterogeneous objects, 2023.

[165] Lawrence Yunliang Chen, Baiyu Shi, Daniel Seita, Richard Cheng, Thomas Kollar, David Held, and Ken Goldberg. Autobag: Learning to open plastic bags and insert objects. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3918–3925, 2023.

[166] Luke Metz, Julian Ibarz, Navdeep Jaitly, and James Davidson. Discrete sequential prediction of continuous actions for deep rl. *arXiv preprint arXiv:1705.05035*, 2017.

[167] Dian Wang, Colin Kohler, and Robert Platt. Policy learning in se(3) action spaces. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 1481–1497. PMLR, 16–18 Nov 2021.

[168] Allison Pinosky, Ian Abraham, Alexander Broad, Brenna Argall, and Todd D Murphey. Hybrid control for combining model-based and model-free reinforcement learning. *The International Journal of Robotics Research*, 42(6):337–355, 2023.

[169] Fabian Jenelten, Junzhe He, Farbod Farshidian, and Marco Hutter. Dtc: Deep tracking control. *Science Robotics*, 9(86):eadh5401, 2024.

[170] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to Manipulate Deformable Objects without Demonstrations. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[171] Irene Garcia-Camacho, Júlia Borràs, Berk Calli, Adam Norton, and Guillem Alenyà. Household cloth object set: Fostering benchmarking in deformable object manipulation. *IEEE Robotics and Automation Letters*, 7(3):5866–5873, 2022.

[172] Irene Garcia-Camacho, Martina Lippi, Michael C. Welle, Hang Yin, Rika Antonova, Anastasiia Varava, Julia Borras, Carme Torras, Alessandro Marino, Guillem Alenyà, and Danica Kragic. Benchmarking bimanual cloth manipulation. *IEEE Robotics and Automation Letters*, 5(2):1111–1118, 2020.

[173] Gabriele Tiboni, Karol Arndt, and Ville Kyrki. Dropo: Sim-to-real transfer with offline domain randomization. *Robotics and Autonomous Systems*, 166:104432, 2023.

[174] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. Unified particle physics for real-time applications. *ACM Trans. Graph.*, 33(4), 2014.

[175] Yen-Ling Tai, Yu Chien Chiu, Yu-Wei Chao, and Yi-Ting Chen. Scone: A food scooping robot learning framework with active perception. In Jie Tan, Marc Toussaint, and Kourosh Darvish, editors, *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 849–865. PMLR, 06–09 Nov 2023.

[176] Daniel Seita, Yufei Wang, Sarthak J Shetty, Edward Yao Li, Zackory Erickson, and David Held. Toolflownet: Robotic manipulation with tools via predicting tool flow from point clouds. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1038–1049. PMLR, 14–18 Dec 2023.

[177] Yixuan Wang, Yunzhu Li, Katherine Driggs-Campbell, Li Fei-Fei, and Jiajun Wu. Dynamic-resolution model learning for object pile manipulation. *arXiv preprint arXiv:2306.16700*, 2023.

[178] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744, 2020.

[179] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. In *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts, July 2017.

[180] Gabriele Tiboni, Andrea Protopapa, Tatiana Tommasi, and Giuseppe Averta. Domain randomization for robust, affordable and effective closed-loop control of soft robots, 2023.

[181] Aleksi Hämäläinen, Karol Arndt, Ali Ghadirzadeh, and Ville Kyrki. Affordance learning for end-to-end visuomotor robot control. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1781–1788, 2019.

[182] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810, 2018.

[183] Murtaza Hazara and Ville Kyrki. Transferring generalizable motor primitives from simulation to real world. *IEEE Robotics and Automation Letters*, 4(2):2172–2179, 2019.

[184] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher Pal, and Liam Paull. Active domain randomization. 2019.

[185] K.J. Åström and P. Eykhoff. System identification—a survey. *Automatica*, 7(2):123–162, 1971.

[186] Huamin Wang, Ravi Ramamoorthi, and James F. O'Brien. Data-driven elastic models for cloth: Modeling and measurement. *ACM Transactions on Graphics*, 30(4):71:1–11, 2011. Proceedings of ACM SIGGRAPH 2011.

[187] Yuming Zhao, Jian Wang, Yi Zhang, and Chao Luo. A novel method of soil parameter identification and force prediction for automatic excavation. *IEEE Access*, 8:11197–11207, 2020.

[188] Pingchuan Ma, Tao Du, Joshua B Tenenbaum, Wojciech Matusik, and Chuang Gan. Risp: Rendering-invariant state predictor with differentiable simulation and rendering for cross-domain parameter estimation. In *International Conference on Learning Representations*, 2021.

[189] Alex X. Lee, Henry Lu, Abhishek Gupta, Sergey Levine, and Pieter Abbeel. Learning force-based manipulation of deformable objects from multiple demonstrations. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 177–184, 2015.

[190] Zhongkui Wang, Kazuki Namima, and Shinichi Hirai. Physical parameter identification of rheological object based on measurement of deformation and force. In *2009 IEEE International Conference on Robotics and Automation*, pages 1238–1243, 2009.

[191] Ixchel G. Ramirez-Alpizar, Mitsuru Higashimori, Makoto Kaneko, Chia-Hung Dylan Tsai, and Imin Kao. Dynamic nonprehensile manipulation for rotating a thin deformable object: An analogy to bipedal gaits. *IEEE Transactions on Robotics*, 28(3):607–618, 2012.

[192] Barbara Frank, Cyrill Stachniss, Rüdiger Schmedding, Matthias Teschner, and Wolfram Burgard. Learning object deformation models for robot motion planning. *Robotics and Autonomous Systems*, 62(8):1153–1174, 2014.

[193] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

[194] Roman Garnett. *Bayesian optimization*. Cambridge University Press, 2023.

[195] Nikolaus Hansen and Anne Auger. Cma-es: evolution strategies and covariance matrix adaptation. In *Proc. 13th annual conference companion on Genetic and evolutionary computation*, 2011.

[196] Yann Bouteiller, Simon Ramstedt, Giovanni Beltrame, Christopher J. Pal, and Jonathan Binas. Reinforcement learning with random delays. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[197] Kuan-Ting Yu, Maria Bauza, Nima Fazeli, and Alberto Rodriguez. More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 30–37, 2016.

[198] Kuan-Ting Yu and Alberto Rodriguez. Realtime state estimation with tactile and visual sensing for inserting a suction-held object. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1628–1635, 2018.

Publication I

David Blanco-Mulero, Markus Heinonen and Ville Kyrki. Evolving-Graph Gaussian Processes. In *International Conference on Machine Learning, Time Series Workshop*, Virtual Workshop., July 24 2021.

Evolving-Graph Gaussian Processes

David Blanco-Mulero [1] **Markus Heinonen** [2] **Ville Kyrki** [1]

Abstract

Graph Gaussian Processes (GGPs) provide a data-efficient solution on graph structured domains. Existing approaches have focused on static structures, whereas many real graph data represent a dynamic structure, limiting the applications of GGPs. To overcome this we propose evolving-Graph Gaussian Processes (e-GGPs). The proposed method is capable of learning the transition function of graph vertices over time with a neighbourhood kernel to model the connectivity and interaction changes between vertices. We assess the performance of our method on time-series regression problems where graphs evolve over time. We demonstrate the benefits of e-GGPs over static graph Gaussian Process approaches.

1. Introduction

Dynamic graphs provide rich representations for temporal structured data to model evolving relationships in the data. Structured data is often dynamic, such as human interactions (Casteigts et al., 2012), social networks (Trivedi et al., 2019) or brain interactions (Ofori-Boateng et al., 2020). These structures have been studied as dynamic graphs, time-varying structures (Le Bars et al., 2020), and *evolving graphs* (Bui-Xuan et al., 2002). However, the predominant methods for estimating dynamic graph evolution fail to account for model uncertainty (Sanchez-Gonzalez et al., 2018; Li et al., 2019; Zhu et al., 2019). This uncertainty can be crucial in dynamic systems where safety constraints are required (Hewing et al., 2020).

A natural choice to quantify uncertainty of a model are Gaussian processes (GPs) (Williams & Rasmussen, 2006). Pioneering attempts of applying Gaussian processes to model dynamics estimate autoregressive temporal transition functions with GPs in the Euclidean domain (Wang et al., 2005; Mattos et al., 2015). More recently, GPs have been applied to problems on graph-structured domains (graph GPs), such as signal processing over graphs (Walker & Glocker, 2019; Venkitaraman et al., 2020; Li et al., 2020) or link prediction (Opolka & Liò, 2020). Some of these works decompose the structured domain through graph convolutions (Walker & Glocker, 2019; Opolka & Liò, 2020) but their output is limited to the Euclidean domain. Others, provide an output in the graph-domain (Venkitaraman et al., 2020; Zhi et al., 2020) but are restricted to a vectorial input. To our knowledge, no Gaussian process models have been proposed to model temporal evolution of graph structures.

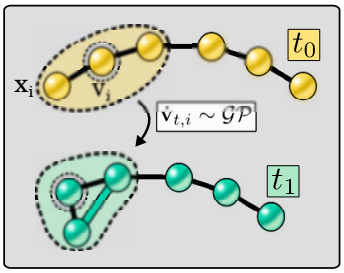


Figure 1: Evolving graph where the neighbourhood of the vertex $\mathbf{v}_i$ changes and the transition is modelled by an evolving-Graph Gaussian Process (e-GGP).

We present an autoregressive GP model that learns the transition function of vertices $f : \mathbf{v} \mapsto \dot{\mathbf{v}}$ on the graph domain G, which we call the evolving-Graph Gaussian Process (e-GGP), see Figure 1. We define a graph kernel that considers the neighbourhoods and attributes of the graph. We investigate the capability of our method to learn the graph evolution in simulated physical dynamical systems, where a measure of the simulation uncertainty can be required. Our results showcase the ability of e-GGP to learn accurate graph dynamics, and extrapolate the dynamics to graphs with unseen structures at test time.

2. Background

Graph-based Gaussian processes. There is a rich literature of different configurations of Gaussian processes

[1]School of Electrical Engineering, Aalto University, Espoo, Finland [2]Department of Computer Science, Aalto University, Espoo, Finland. Correspondence to: David Blanco-Mulero <david.blancomulero@aalto.fi>.

Time Series Workshop at 38^{th} International Conference on Machine Learning, 2021.

over graph domains. As previously mentioned, we can distinguish between two configurations. The first, maps from a vectorial input to an output in the graph domain $f : \mathbb{R}^D \to G$ (Venkitaraman et al., 2020; Zhi et al., 2020). The second, takes an input graph and maps it to the Euclidean domain $f : G \to \mathbb{R}$ (Ng et al., 2018; Walker & Glocker, 2019; Opolka & Liò, 2020; Borovitskiy et al., 2020). For a more thorough overview of graph GP methods see Supplementary Material Section A.1. These methods consider a fixed adjacency matrix of the graph input, or use convolution operations (van der Wilk et al., 2017), which narrows their input to a static graph structure.

Gaussian processes for time-series. The Gaussian process dynamical model (GPDM) models autoregressive Markov transitions of discrete-time random states with a Gaussian process (Wang et al., 2005). In latent modelling a Gaussian process interpolates the object trajectories in latent space (Damianou et al., 2011). More generally these models fall under recurrent Gaussian processes (Mattos et al., 2015). The GPAR extends multi-output Gaussian processes to time-series (Requeima et al., 2019). These models are limited to vector-valued inputs, and do not support non-Euclidean inputs.

The most relevant prior work to our aims is the deep Graph Gaussian process (DGPG) (Li et al., 2020), where the graph structure is auto-regressed before applying a conventional classification or regression step. However, their approach is limited to a fixed graph structure.

3. Evolving Graphs using Gaussian Processes

We consider a temporally evolving, or *dynamic*, undirected graph $G_t = \langle \mathcal{V}_t, \mathcal{E}_t \rangle$, where $\mathcal{V}_t$ is the set of vertices and $\mathcal{E}_t \subset \mathcal{V}_t \times \mathcal{V}_t$ is the set of edges at discrete timepoints $t \in N$ of the graph time-series $G = \{G_0, \dots, G_N\}$. For notation simplicity we consider the number of vertices in each graph to be the same $|\mathcal{V}_t| = M$.

The vertices $\mathbf{v}_t \in \mathcal{V}$ reside in a D-dimensional space. For our objective of learning physics simulations, a vertex at time t is represented as $\mathbf{v}_t := (\mathbf{x}_t, \Delta\mathbf{x}_{t-1}, \phi(\mathbf{v}_t))$ over the Cartesian coordinates $\mathbf{x}$, and static attributes $\boldsymbol{\phi}(\mathbf{v})$, which describe node properties or types. We assume that edges are a function of vertices and that the vertices-to-edges function $\mathbf{g} : 2^{\mathcal{V}} \to \mathcal{E}$ is defined by a distance function $d(\mathbf{v}_i, \mathbf{v}_j)$. We adopt the L2-norm distance between the vertices Cartesian coordinates. Therefore, an edge exists if the distance between two nodes is lower than a connectivity threshold R_{nn}, see Supplementary Material Section B.

We assume the graph dynamics are governed by

$$G_{t+1} = T(G_t) \tag{1}$$

driven by the graph transition function $T : G \to G$, where the graph transition is defined by a transition function of node attributes $\mathcal{V}_{t+1} = T(\mathcal{V}_t)$ and the edges are given by the nodes-to-edge function. Our goal is to learn the evolution function T. Hence, we opt to model and learn the node evolution in the discrete timestep Δt by:

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \underbrace{\mathbf{f}(N^{0,1}(\mathbf{v}_t))}_{\Delta\mathbf{v}_t}, \tag{2}$$

with a velocity function $\mathbf{f} : s(G) \to \mathbb{R}^F$ mapping a sub-tree $s(G)$ into coordinate velocities $\mathbb{R}^F$. The sub-tree $N^{0,1}(\mathbf{v}_t)$ consists of a node $\mathbf{v}_t$, or $N^0(\mathbf{v}_t)$, and its neighbours $N^1(\mathbf{v}_t)$. To simplify the notation in the following, we will use $\mathbf{f}(\mathbf{v}_t)$ to denote the mapping. This formulation allows us to handle node insertion and deletion.

3.1. Prior and likelihood

We model the velocity transitions with a vector-valued Gaussian process (GP) prior (Williams & Rasmussen, 2006; Alvarez et al., 2012)

$$\mathbf{f}(\mathbf{v}) \sim \mathcal{GP}(\mathbf{m}(\mathbf{v}), K(N^{0,1}(\mathbf{v}), N^{0,1}(\mathbf{v}'))), \tag{3}$$

which implies that the transition is a stochastic process, whose mean and covariance are parameterised by the mean function $\mathbf{m}(\cdot)$ and the kernel similarity $K(\cdot, \cdot)$ as

$$\mathbb{E}[\mathbf{f}(\mathbf{v})] = \mathbf{m}(\mathbf{v}), \tag{4}$$

$$\mathrm{cov}[\mathbf{f}(\mathbf{v}), \mathbf{f}(\mathbf{v}')] = K(N^{0,1}(\mathbf{v}), N^{0,1}(\mathbf{v}')). \tag{5}$$

For tractability, we assume different velocity predictions are independent, however sharing the same kernel, which simplifies the kernel into a scalar function. Furthermore, for any subset of nodes $\mathbf{v}_1, \dots, \mathbf{v}_M$ the corresponding transitions are jointly Gaussian

$$p(\Delta\mathbf{v}_1, \dots, \Delta\mathbf{v}_M) = \mathcal{N}(\Delta\vec{\mathbf{v}} | \vec{\mathbf{m}}, \mathbf{K}), \tag{6}$$

where $\Delta\vec{\mathbf{v}} \in \mathbb{R}^{MF}$ and $\vec{\mathbf{m}} \in \mathbb{R}^{MF}$ stack all vertices and means to column vectors, and $\mathbf{K} \in \mathbb{R}^{MF \times MF}$ is the block kernel matrix consisting of $M \times M$ blocks $K(\mathbf{v}_i, \mathbf{v}_j)$ of size $F \times F$. The key property of Gaussian processes is that similar nodes have similar transitions, as specified by the similarity function K.

We assume a dataset

$$\mathcal{D} = \left\{ (\mathbf{v}_{t,i}, \Delta\mathbf{v}_{t,i}) \right\}_{t=1,i=1}^{N,M} \tag{7}$$

of observed graph states G_t that are described in terms of the M vertices at N timepoints. Our observation model induces a likelihood

$$p(\mathcal{D}|\mathbf{f}) = \prod_{t=1}^{N} \mathcal{N}\left(\mathbf{v}_{t,i} | \mathbf{f}, \Sigma\right), \tag{8}$$

where $\mathbf{f}$ is the estimated node evolution following equation (2), and $\Sigma \in \mathbb{R}^{F \times F}$ represents the numerical variance required for numerical stability.

3.2. Posterior

The predictive posterior distribution of our Gaussian process model is conveniently tractable under the Gaussian likelihood model (8) as (Williams & Rasmussen, 2006)

$$p(\mathbf{f}_*|\mathcal{D}) = \mathcal{N}(\mathbf{f}_*|\boldsymbol{\mu}_*, \Sigma_*), \tag{9}$$

$$\boldsymbol{\mu}_* = \mathbf{K}_*(\mathbf{K}+\boldsymbol{\Sigma})^{-1}\Delta\vec{\mathbf{v}}, \tag{10}$$

$$\Sigma_* = \mathbf{K}_{**} - \mathbf{K}_*(\mathbf{K}+\boldsymbol{\Sigma})^{-1}\mathbf{K}_*^T, \tag{11}$$

where $\mathbf{K} \in \mathbb{R}^{NMF\times NMF}$, $\mathbf{K}_* \in \mathbb{R}^{N_*MF\times NMF}$ and $\mathbf{K}_{**} \in \mathbb{R}^{N_*MF\times N_*MF}$ for a test set consisting of N_* graphs with M nodes. The posterior effectively interpolates the velocity predictions from the observed velocities of the training graphs.

3.3. Kernel between attributed sub-trees

We now define a kernel that can measure the similarity between vertices and their transitions. We propose to use a kernel on the graph that measures the similarity between attributed sub-trees $s(G)$, $K : s(G)\times s(G) \to \mathbb{R}$. For simplicity, we refer to the kernel as $K(\mathbf{v}_i, \mathbf{v}_j)$. The kernel requires the attributed vertices to be in the same space. Therefore, we define a vertex mapping function $\varphi_\mathbf{v}$ which maps the attributed vertex to an Euclidean-space $\varphi_\mathbf{v} : \mathbb{R}^D \mapsto \mathbb{R}^E$. In our case, $\varphi_\mathbf{v}$ discards the static attributes of the vertices.

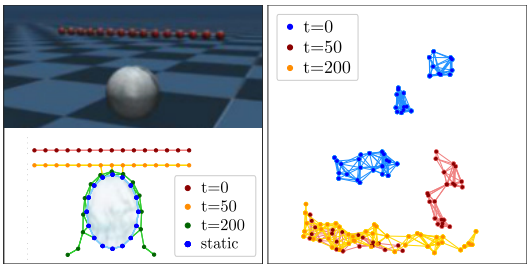
The kernel input sub-trees are rooted at the vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ with 1-hop neighbourhood. We measure the similarity between the attributed roots using the *root kernel* $k_r : \mathbb{R}^E \times \mathbb{R}^E \to \mathbb{R}$. In order to share information with the root and compare the structures, we compute a kernel between all the attributed leaves in the neighbourhood $N_i^1 = N^1(\mathbf{v}_i)$. The similarity between the leaves is measured by the *leaf kernel* $k_l : \mathbb{R}^E \times \mathbb{R}^E \to \mathbb{R}$. Both the *root* and the *leaf kernel* operations are performed in the mapped space $\mathbb{R}^E$. We express the neighbourhood kernel $k_{nn}(\mathbf{v}_i, \mathbf{v}_j)$ as

$$k_{nn}(\mathbf{v}_i, \mathbf{v}_j) = \frac{1}{L}\sum_{\mathbf{x}_i\in N_i^1}\sum_{\mathbf{x}_j\in N_j^1} k_l(\varphi_\mathbf{v}(\mathbf{x}_i), \varphi_\mathbf{v}(\mathbf{x}_j)), \tag{12}$$

where L is a normalisation constant that denotes the product between the number of leaves in N_i^1 and N_j^1. Then, the kernel between two attributed vertices is defined by

$$K(\mathbf{v}_i, \mathbf{v}_j) = k_r(\varphi_\mathbf{v}(\mathbf{v}_i), \varphi_\mathbf{v}(\mathbf{v}_j)) + k_{nn}(\mathbf{v}_i, \mathbf{v}_j). \tag{13}$$

We restrict the kernel to the 1-hop neighbourhood for scalability and plan to investigate the effect of the neighbourhood in future work.



(a) (b)

Figure 2: Physics simulations: (a) graph interaction environment in MuJoCo simulation and 2-D approximation, (b) evolving isolated sub-graphs environment, showing graphs connectivity for different timepoints t.

4. Experimental results

4.1. Experiments description

We evaluate the performance of e-GGPs to learn the node transition function $\Delta\mathbf{v}_t = \mathbf{f}(\mathbf{v}_t)$ in two physics simulations where graphs evolve over time: 1) graph interaction (GI) environment, and 2) evolving isolated sub-graphs (EIs) environment. The GI environment simulates a rope on free fall that gets in contact with a static ball using MuJoCo (Todorov et al., 2012). The EIs environment simulates a set of 2D fluid particles using Taichi (Hu et al., 2018). Both simulations are represented in 2-D, where the joints of the rope, the static ball and the particles define the set of nodes $\mathbf{p}_t = \mathbf{v}_t \in \mathcal{V}$, see Figure 2. Thus, the experiments consist of multiple sub-graphs where connectivity varies over time.

The purpose our experiments is twofold: (i) investigate the capability of the methods to address the varying connectivity and (ii) validate that e-GGP can learn the evolution of the graph by capturing the interaction changes between vertices.

4.2. Baselines and training details

We consider three Gaussian process methods as baselines: GP regression (GPR) using exact inference, GPAR and DGPG. The first two baselines are GP functions that model transition functions in the Euclidean domain. We use baselines in the Euclidean domain to demonstrate the potential of graph-structured information compared to approaches restricted to a vector-valued input. For these we take the vertices attributes vector $\mathcal{V}$ as input. Then, we learn a mapping $\mathbf{f} : \mathbb{R}^D \to \mathbb{R}^F$. The mapping of DGPG is defined as $\mathbf{f} : G \to \mathbb{R}^F$. Since DGPG does not handle node insertions we assume the adjacency matrix to be equal to the initial state graph adjacency for any given timepoint $A_t = A_0$.

Table 1: Average results of GPR, GPAR, DGPG and e-GGP on test sets (lower is better) for different training data sizes N.

		Graph interaction				Evolving isolated sub-graphs			
N	**Metric**	**GPR**	**GPAR**	**DGPG**	**e-GGP**	**GPR**	**GPAR**	**DGPG**	**e-GGP**
10	RMSE	0.121	0.143	0.413	**0.049**	0.294	0.283	0.337	**0.182**
	MAPE	0.418	0.537	2.456	**0.249**	0.873	**0.726**	2.696	0.777
	NLL	67.63	**-9.16**	9.51	10.44	43.37	-47.942	22.24	**-58.042**
15	RMSE	0.149	0.209	0.275	**0.031**	0.276	0.277	0.325	**0.183**
	MAPE	0.410	0.397	1.873	**0.230**	**0.671**	**0.671**	3.135	0.853
	NLL	102.49	-7.67	2.20	**-17.56**	103.82	-46.764	20.12	**-78.58**
20	RMSE	0.162	0.250	0.222	**0.036**	0.257	0.253	0.300	**0.160**
	MAPE	0.426	0.466	1.455	**0.148**	0.612	**0.594**	2.998	0.766
	NLL	126.97	-6.84	-2.18	**-29.69**	4.17E6	**-48.52**	9.731	8.77E5

The kernel hyper-parameters are trained using automatic relevance determination (ARD) and optimised by minimising the negative marginal log-likelihood (MLL) (Williams & Rasmussen, 2006). For all the methods we used as optimizer Adam (Kingma & Ba, 2015). We use a Radial Basis Function kernel for the *root* and *leaf* kernels k_r and k_l in e-GGP and GPR. In DGGP we use the Matérn32 kernel and $Z = 5$ inducing points. For more details about the training see Supplementary Material Section B.

4.3. Results

We investigate the error of the node evolution predictions. In order to evaluate the methods ability to capture the evolving graph information we limit the amount of data provided in training. We select the most informative training points as described in Supplementary Material Section B. To evaluate the prediction performance and confidence we measure the root mean squared error (RMSE), mean absolute percentage error (MAPE) and negative log-likelihood (NLL). The test datasets on the GI environment include an offset between the rope and the static ball from the training data to evaluate the generalisation capabilities of the methods. We provide additional results in Supplementary Material Section C.

The results for the GI environment on Table 1 show that e-GGP outperforms the baselines when data is scarce. The different orders of magnitude in the RMSE provide evidence that the method we propose benefits of the evolving graph-structured information captured by the kernel. These results also provide evidence that e-GGP can generalise better to unseen data under limited training samples. Furthermore, e-GGP presents low RMSE for the EIs environment, which shows that the model can generalise to unseen sub-graph structures, i.e. sub-graphs that have not been seen during training. When increasing the sample size to $N = 20$ the confidence drops while keeping low RMSE and MAPE. This suggests that the model is not able to provide a confident estimate under sparse data, which will be investigated in the future. The DGPG method is limited to a fixed graph size and cannot handle node insertions, which is shown by the higher error results.

In summary, the results provide evidence that: (i) e-GGP can predict the node evolution under varying connectivity, and (ii) learning the interactions between the evolving vertices is beneficial as e-GGP outperforms the baselines in the limited data setting.

5. Conclusion

In this paper we proposed evolving-Graph Gaussian Processes (e-GGPs), an autoregressive graph-GP model that learns the evolving structure of dynamic graphs via the node transitions. Our model is able to learn the node transitions as well as the interactions of the structure via an attributed subtree kernel that incorporates information from each node's neighbourhood. The experimental results demonstrate that our model is able to capture evolving graph connectivity, overcoming this limitation of current methods. We provide evidence of the importance of capturing the dynamic connectivity and investigated the performance of our method in scarce data. Our experiments showed that e-GGP outperforms other approaches by making use of the rich information in the evolving graph-domain.

Our model opens GPs to dynamic graphs which is of great interest in real graph problems. However, our model suffers from poor scalability, which we plan to investigate via variational inference. We also plan to study the application of e-GGP to real problems that require both data-efficiency and uncertainty, such as dynamic systems with safety constraints, where other approaches such as neural networks are limited.

Acknowledgements

This work was supported by the Academy of Finland, decision 317020.

References

Alvarez, M. A., Rosasco, L., and Lawrence, N. D. Kernels for vector-valued functions: a review, 2012.

Borovitskiy, V., Azangulov, I., Terenin, A., Mostowsky, P., Deisenroth, M. P., and Durrande, N. Matern Gaussian Processes on Graphs. *arXiv:2010.15538 [cs, stat]*, October 2020. arXiv: 2010.15538.

Bui-Xuan, B.-M., Ferreira, A., and Jarry, A. Computing shortest, fastest, and foremost journeys in dynamic networks. Technical Report RR-4589, INRIA, October 2002.

Casteigts, A., Flocchini, P., Quattrociocchi, W., and Santoro, N. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

Damianou, A., Titsias, M., and Lawrence, N. Variational gaussian process dynamical systems. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31, pp. 7576–7586. Curran Associates, Inc., 2018.

Hewing, L., Kabzan, J., and Zeilinger, M. N. Cautious model predictive control using gaussian process regression. *IEEE Transactions on Control Systems Technology*, 28(6):2736–2743, 2020.

Hu, Y., Fang, Y., Ge, Z., Qu, Z., Zhu, Y., Pradhana, A., and Jiang, C. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)*, 37(4): 150, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Le Bars, B., Humbert, P., Kalogeratos, A., and Vayatis, N. Learning the piece-wise constant graph structure of a varying ising model. In *International Conference on Machine Learning*, pp. 675–684. PMLR, 2020.

Li, N., Li, W., Sun, J., Gao, Y., Jiang, Y., and Xia, S.-T. Stochastic deep gaussian processes over graphs. *Advances in Neural Information Processing Systems*, 33, 2020.

Li, Y., Wu, J., Tedrake, R., Tenenbaum, J. B., and Torralba, A. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Mattos, C. L. C., Dai, Z., Damianou, A., Forth, J., Barreto, G. A., and Lawrence, N. D. Recurrent gaussian processes. *arXiv preprint arXiv:1511.06644*, 2015.

Ng, Y. C., Colombo, N., and Silva, R. Bayesian semi-supervised learning with graph gaussian processes. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31, pp. 1683–1694. Curran Associates, Inc., 2018.

Ofori-Boateng, D., R.Gel, Y., and Cribben, I. Nonparametric anomaly detection on time series of graphs. 2020. doi: 10.1101/2019.12.15.876730.

Opolka, F. L. and Liò, P. Graph Convolutional Gaussian Processes For Link Prediction. In *Workshop on Graph Representation Learning and Beyond (GRL+)*, 2020.

Requeima, J., Tebbutt, W., Bruinsma, W., and Turner, R. E. The gaussian process autoregressive regression model (gpar). In *AISTATS*, pp. 1860–1869, 2019.

Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M., Hadsell, R., and Battaglia, P. Graph networks as learnable physics engines for inference and control. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4470–4479, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Trivedi, R., Farajtabar, M., Biswal, P., and Zha, H. Dyrep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*, 2019.

van der Wilk, M., Rasmussen, C. E., and Hensman, J. Convolutional gaussian processes. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30, pp. 2849–2858. Curran Associates, Inc., 2017.

Venkitaraman, A., Chatterjee, S., and Handel, P. Gaussian processes over graphs. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5640–5644, 2020. doi: 10.1109/ICASSP40776.2020.9053859.

Walker, I. and Glocker, B. Graph convolutional Gaussian processes. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6495–6504. PMLR, 09–15 Jun 2019.

Wang, J. M., Fleet, D. J., and Hertzmann, A. Gaussian process dynamical models. In *NIPS*, volume 18, pp. 3. Citeseer, 2005.

Williams, C. K. and Rasmussen, C. E. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Zhi, Y.-C., Ng, Y. C., and Dong, X. Gaussian Processes on Graphs via Spectral Kernel Learning. *arXiv:2006.07361 [cs, eess, stat]*, October 2020. arXiv: 2006.07361.

Zhu, C., Storandt, S., Lam, K.-Y., Han, S., and Bi, J. Improved dynamic graph learning through fault-tolerant sparsification. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7624–7633. PMLR, 09–15 Jun 2019.

Supplementary Material: Evolving Graph Gaussian Processes

A. Analysis of e-GGP

A.1. Overview of graph Gaussian processes

There is a rich literature of different configurations of Gaussian processes over graph domains. Graph-output Gaussian processes consider functions $f : \mathbb{R}^D \to G$, where the $|\mathcal{V}|$ outputs have dependencies according to an underlying output graph (Venkitaraman et al., 2020; Zhi et al., 2020). In semi-supervised Gaussian processes a graph relationship between inputs is assumed to propagate label information within the graph from labelled to unlabelled inputs (Ng et al., 2018). Borovitskiy et al. (2020) adapts Matérn Gaussian processes for graph node labelling. The graph convolutional Gaussian processes consider functions $f : G \to \mathbb{R}$, where the label of an entire input graph G is predicted (Walker & Glocker, 2019; Opolka & Liò, 2020) with convolution operations (van der Wilk et al., 2017).

Our method learns the transition function of vertices $f : \mathbf{v} \mapsto \dot{\mathbf{v}}$ and infers the new edges using a nodes-to-edge function $\mathbf{g} : 2^{\mathcal{V}} \to \mathcal{E}$, learning the graph transition. We provide an overview of the different graph-GP methods as well as their mapping function to highlight the scope of our work on Table A.1.

Table A.1: Overview of different graph-GP methods, their mapping functions $\mathbf{f}$ and whether an static graph is assumed or not. We denote each model by the name addressed in the paper or by the formulation they propose.

	Input domain			Output domain				
Model	$\mathbb{R}^D$	$\mathcal{V}$	$\mathcal{G}$	$\mathbb{R}^D$	$\mathcal{Y}$	$\mathcal{G}$	Time-series	Reference
GPDM	✓	-	-	✓	-	-	✓	Wang et al. (2005)
GGP	-	✓	-	-	✓	-	-	Ng et al. (2018)
GCGP	-	-	✓	-	✓	-	-	Walker & Glocker (2019); Opolka & Liò (2020)
DGPG	-	-	✓	✓	-	-	-	Li et al. (2020)
Matérn Graph GP	-	-	✓	✓	-	-	-	Borovitskiy et al. (2020)
GPG	✓	-	-	-	-	✓	-	Venkitaraman et al. (2020)
Graph-output GP	✓	-	-	-	-	✓	-	Zhi et al. (2020)
Ours	-	-	✓	-	-	✓	✓	

B. Supplementary training details

B.1. Experiments data sets

The number of nodes in each of the graph data sets is provided in Table B.1. The connectivity hyper-parameter R_{nn} and the maximum number of neighbours K_{nn} are selected based on the environment definition.

Table B.1: Description of train and test data for the experiments described in Section 4. N denotes the horizon, $|\mathcal{V}| = M$ the number of nodes in the graph and D the attributes of the nodes.

	N	R_{nn}	K_{nn}	M_{TRAIN}	D_{TRAIN}	M_{TEST}	D_{TEST}
GRAPH INTERACTION	500	0.043	2	31	7	31	7
ISOLATED EVOLVING SUB-GRAPHS	200	0.08	20	44	8	44, 55, 66	8

Graph interaction connectivity. We define two objects: an elastic rope and a static ball. The distance between the nodes of each object was set to $d(\mathbf{v}_i, \mathbf{v}_j) = 0.03$. However, because of the elasticity of the material in MuJoCo (Todorov et al., 2012), we found out experimentally that using a connectivity lower than $R_{nn} = 0.043$ would lead to disconnected nodes in the graph that are physically connected in the simulation. Considering that each of the objects is internally connected, we use that information to build the graph connectivity. Therefore, we considered the maximum number of neighbours affecting a node to be $K_{nn} = 2$, as the internal connectivity was already taken into account.

The attributes of the nodes are set to $\mathbf{v}_t = \{x_t, y_t, z_t, \Delta x_{t-1}, \Delta y_{t-1}, \Delta z_{t-1}, \phi(\mathbf{v}_t)\}$, where $\phi(\mathbf{v}_t) = 0$ if the node belonged to the ball and $\phi(\mathbf{v}_t) = 1$ if it was part of the rope.

Isolated evolving sub-graphs connectivity. We randomise for each test the initial conditions of the particles (Hu et al., 2018). For each of the tests, we initialise a random number of blocks of particles, where the initial position and velocity are also randomised. This lead to a different evolution of the connectivity, as shown in Figure B.1.

The resolution used in our Taich-MPM environment was set to 16. We found out experimentally that particles that are separated by a distance of $R_{nn} = 0.08$ or lower had an effect on each other so as to be considered neighbours. We decided to use $K_{nn} = 20$ to provide the maximum information from the nearby particles.

In this environment, the nodes are defined as $\mathbf{v}_t = \{x_t, y_t, \Delta x_{t-1}, \Delta y_{t-1}, d(x_t, x_{\text{max}}), d(y_t, y_{\text{max}}), d(x_t, x_{\text{min}}), d(y_t, y_{\text{min}})\}$, where x_{max}, x_{min}, y_{max}, y_{min} refer to the boundaries of the water container.

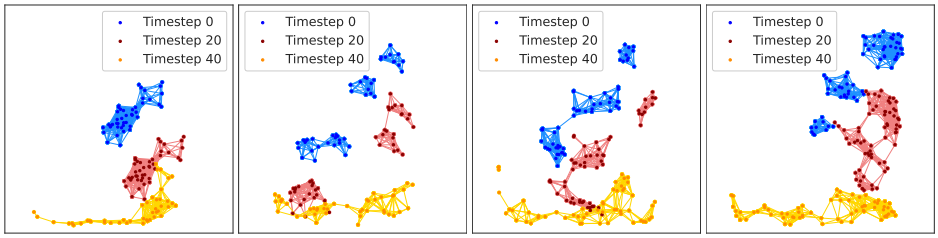


Figure B.1: Visualisation of the evolving isolated sub-graphs initial configuration, evolution and connectivity for (a) training data $N = 44$, (b) test data $N = 44$, (c) test data $N = 55$, (d) test data $N = 66$.

B.2. Kernel selection

The attributes of the nodes live in the coordinate space. Therefore, we decided to use the Radial Basis Function (RBF) stationary kernel as the *roof* and *leaf* kernels for e-GGP as well as for the baseline of GPR. We note to the reader that different combinations of kernels are possible when using e-GGP. As an example, one could choose to use an RBF *root* kernel and a Matérn 52 *leaf* kernel. The decision of the kernel to use for the root nodes and the neighbourhood should be based on the expert knowledge.

For the *graph interaction environment*, even though the environment is approximated in 2-D, the nodes are attributed with 3-dimensional coordinates and velocities. The models are not affected by the third dimension as the kernel hyper-parameters will reject the non-informative dimensions as training is done using automatic relevance determination (ARD) (Williams & Rasmussen, 2006).

B.3. Implementation details

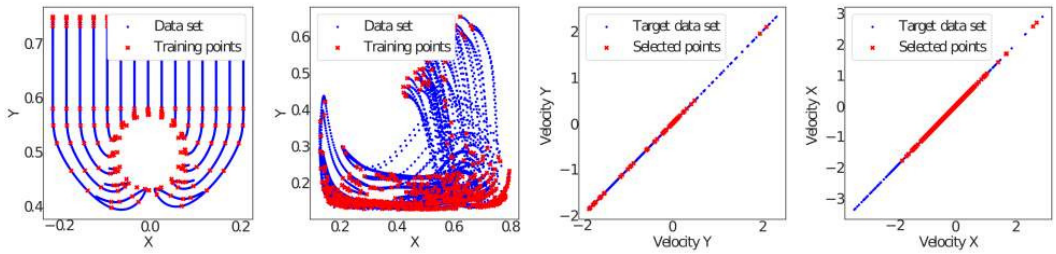
e-GGP implementation. We provide an open-source implementation of e-GGP[1]. Our implementation is based on GPyTorch (Gardner et al., 2018). The implementation of the *root* and *leaf* kernels was done so that the stationary kernels distance was scaled by a length-scale parameter θ_d per dimension $d \in D$, and a noise parameter σ^2.

[1] https://github.com/dblanm/evolving-ggp

Graph definition. The graph is built as follows. We take the attributed nodes at a timepoint $\mathcal{V}_t$ and build a k-d tree. We define the neighbourhood in the tree by the nearest K neighbours K_{nn}, with distance lower than the threshold R_{nn}, to the queried node. The K_{nn} parameter can also be seen as the maximum degree of each node. The neighbours denoted by the parameters K_{nn} and R_{nn} define the edges of the graph $\mathcal{E}_t$, thereby forming the graph $G_t = \langle \mathcal{V}_t, \mathcal{E}_t \rangle$ at the given timepoint t.

Training points selection. We select points from the training data set based on the rate of change of the target. Therefore, if the target is the velocity Δx_t, we select the points with highest acceleration $\Delta^2 x_t$. In order to avoid points that are close in the temporal dimension, we restrict the time between samples to be at least $\Delta t = 20$ timepoints far from each other. In case that there are no more data points fitting the temporal constraint, the time difference is constantly reduced until the requested number of training points is satisfied.

We show the coordinate space for each environment in Figure B.2. The blue markers show the full set of training points whereas the red markers show the limited selected training points, in this case $N = 20$. The Figure B.2 also shows a target for each of the environments. Note that each node covers a different part of the state-space.



(a) (b) (c) (d)

Figure B.2: Coordinate state space in (a) GI environment and (b) IEs environment showing full data set (blue) and selected $N = 20$ training points (red). Selected training targets (c) Δy_t and (d) Δx_t in each environment respectively.

Hardware used for training All the experiments were performed in CPU. For training e-GGP we used a shared server with a CPU Xeon E5-2680 v2. All the other models were trained in a personal laptop.

Training time We provide information about the approximate training time of e-GGP in Table B.2 for different number of training points. We trained e-GGP using Adam with a learning rate of $l_r = 0.1$ for 150 iterations. We note that the computational complexity highly depends on the number of nodes in the graph M and the maximum connectivity for a node. We would like to highlight that our implementation of the kernel used in e-GGP is not ideal and the time taken shown in Table B.2 can be highly reduced.

Table B.2: Training time of e-GGP for the experiments using a CPU Xeon E5-2680 v2.

	N	M	W	D	t
GRAPH INTERACTION	5	31	4	7	14MIN
	10	31	4	7	1H 20MIN
	15	31	4	7	2H 55MIN
	20	31	4	7	4H 24MIN
ISOLATED EVOLVING SUB-GRAPHS	5	44	20	8	33MIN
	10	44	20	8	2H 16MIN
	15	44	20	8	5H 42MIN
	20	44	20	8	14H 7MIN

C. Supplementary results

In Section 4, we provided the results for the Graph Interaction environment. In the Isolated Evolving Sub-graphs environment, because of the limitation of DGPG to predict test inputs with a different graph structure, we only provided results for a single test set. Below, we extend those results and detail the results for each test set in the experiments. We also provide an ablation study in the GI environment.

C.1. e-GGP ablation study

C.2. Graph connectivity ablation study

We evaluated two variants of our method. The first one assumes that the graph connectivity is fixed (F. e-GGP), using the same adjacency matrix during inference and predictions $A = A_0$. The second one, has knowledge of the evolving graph structure (E. e-GGP) and uses a different adjacency A_t as the graph evolves over time.

We first compare the results for each of the unseen test sets for Δx_t and Δy_t in Table C.1 and Table C.2 respectively. We can see that the F. e-GGP variant performs slightly better for the offsets close to the training data. However, the evolving version (E. e-GGP) clearly outperforms the fixed variant shown in Table C.2. This supports our hypothesis and highlights the relevance of accounting for the graph evolution.

Table C.1: Results of fixed (F. e-GGP) and evolving (E. e-GGP) variants in GI environment for Δx_t with $N = 15$.

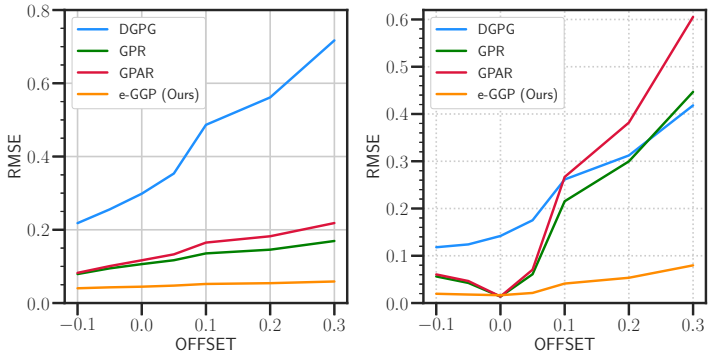
OFFSET	RMSE		MAPE		NLL	
	F. E-GGP $\times 10^{-2}$	E. E-GGP $\times 10^{-2}$	F. E-GGP $\times 10^{12}$	E. E-GGP $\times 10^{12}$	F. E-GGP	E. E-GGP
-0.1	**0.810**	0.879	**4.14**	4.52	**-57.8**	-54.0
-0.05	**0.890**	0.956	**4.28**	4.80	**-56.7**	-52.7
0	**0.980**	1.04	**3.89**	3.92	**-54.8**	-50.4
0.05	**1.10**	1.14	4.57	**4.17**	**-52.5**	-48.3
0.1	**1.40**	1.41	10.8	**9.80**	**-47.1**	-41.4
0.2	**1.56**	**1.56**	16.1	**15.4**	**-44.7**	-38.3
0.3	**1.87**	**1.87**	29.5	**29.4**	**-43.2**	-37.5

Table C.2: Results of fixed (F. e-GGP) and evolving (E. e-GGP) variants in GI environment for Δy_t with $N = 15$.

OFFSET	RMSE		MAPE		NLL	
	F. E-GGP $\times 10^{-2}$	E. E-GGP $\times 10^{-2}$	F. E-GGP $\times 10^{-1}$	E. E-GGP $\times 10^{-1}$	F. E-GGP	E. E-GGP
-0.1	2.52	**2.30**	3.98	**3.79**	-13.91	**-18.14**
-0.05	2.42	**2.36**	3.98	**3.79**	**-17.98**	-16.00
0	**2.17**	2.31	3.32	**2.86**	**-31.15**	-20.89
0.05	**2.25**	2.45	1.74	**1.73**	**-33.17**	-23.30
0.1	3.61	**3.31**	**1.04**	**1.04**	**-18.80**	-17.10
0.2	4.63	**3.98**	5.01	**4.69**	-11.09	**-12.10**
0.3	6.84	**5.46**	0.861	**0.732**	0.20	**-3.97**

C.3. Results on scarce data for the graph interaction environment

In this section, we provide additional results for the GI environment. We show in Figure C.1 the RMSE results for different offsets of the rope. The rope offset used for training was $Z = 0$. We can see that e-GGP outperforms the other methods for both $N = 10$ and $N = 20$ training points.



(a) (b)

Figure C.1: Comparison of one-step ahead prediction of RMSE results for GPR, GPAR, DGPG and e-GGP in the graph interaction experiment for (a) 10 training points and (b) 20 training points.

C.4. Results on scarce data for the isolated evolving sub-graphs environment

We provide the comparison of e-GGP and the baselines GPR, GPAR for the node velocity prediction for each target Δx_t and Δy_t in Table C.3 and Table C.4 respectively. We discussed in Section 4.3 the averaged results of e-GGP. In here, we can see that for both targets e-GGP has consistent low RMSE, MAPE and NLL. These results show that our model is capable of learning the transitions more accurately, regardless of the target.

Table C.3: Results for GPR, GPAR and e-GGP in evolving isolated sub-graphs test datasets for the target Δx_t.

		RMSE			MAPE			NLL		
N	M	GPR	GPAR	E-GGP	GPR	GPAR	E-GGP	GPR	GPAR	E-GGP
5	55	0.162	0.194	**0.107**	1.27	2.06	**1.09**	**-40.173**	242.025	3.542
	66	0.306	0.307	**0.148**	3.03	4.95	**2.23**	**-22.676**	435.562	89.792
10	55	0.148	0.172	**0.110**	0.97	1.32	**0.88**	**-43.425**	83.373	3.574
	66	0.397	0.277	**0.154**	**2.33**	3.31	2.37	**-18.255**	370.486	193.102
15	55	0.133	0.126	**0.096**	0.86	0.86	**0.79**	-49.657	146.950	**-90.342**
	66	0.325	0.295	**0.152**	2.41	2.56	**2.16**	-30.424	490.559	**-74.796**
20	55	0.117	0.111	**0.086**	0.78	**0.65**	0.70	-49.822	5.2×10^6	**-96.042**
	66	0.334	0.309	**0.194**	2.04	**1.76**	2.10	-30.945	1.07×10^7	1.98×10^6

Table C.4: Results for GPR, GPAR and e-GGP in evolving isolated sub-graphs test datasets for the target Δy_t.

		RMSE			MAPE			NLL		
N	M	**GPR**	**GPAR**	**E-GGP**	**GPR**	**GPAR**	**E-GGP**	**GPR**	**GPAR**	**E-GGP**
5	55	0.403	0.396	**0.264**	**2.52**	3.25	3.39	**-21.474**	92.63	<u>-0.256</u>
	66	0.643	0.751	**0.567**	4.48	**3.70**	6.07	**5.156**	167.1	200.3
10	55	0.414	0.429	**0.318**	1.38	1.16	**1.07**	-47.420	**-114.1**	<u>-110.4</u>
	66	**0.647**	0.654	0.713	3.54	8.73	**1.39**	-13.558	**-55.81**	-26.29
15	55	0.435	**0.391**	0.418	1.07	**0.93**	1.48	-42.35	**-111.38**	-50.755
	66	0.645	0.632	**0.596**	4.87	**1.45**	<u>1.83</u>	-9.78	**-44.65**	95.468
20	55	0.383	0.407	**0.253**	1.00	**0.78**	1.09	2.39×10^5	**-50.321**	1.17×10^6
	66	0.596	0.668	**0.435**	3.04	**0.96**	2.19	3.65×10^5	**-19.752**	2.47×10^6

Publication II

Neea Tuomainen*, **David Blanco-Mulero*** and Ville Kyrki. Manipulation of Granular Materials by Learning Particle Interactions. *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, issue 2, pp. 5663-5670, April 2022.

Manipulation of Granular Materials by Learning Particle Interactions

Neea Tuomainen, David Blanco-Mulero, and Ville Kyrki, *Senior Member, IEEE*

***Abstract*—Manipulation of granular materials such as sand or rice remains an unsolved problem due to challenges such as the difficulty of defining their configuration or modeling the materials and their particles interactions. Current approaches tend to simplify the material dynamics and omit the interactions between the particles. In this letter, we propose to use a graph-based representation to model the interaction dynamics of the material and rigid bodies manipulating it. This allows the planning of manipulation trajectories to reach a desired configuration of the material. We use a graph neural network (GNN) to model the particle interactions via message-passing. To plan manipulation trajectories, we propose to minimise the Wasserstein distance between a predicted distribution of granular particles and their desired configuration. We demonstrate that the proposed method is able to pour granular materials into the desired configuration both in simulated and real scenarios.**

***Index Terms*—Deep learning in grasping and manipulation, machine learning for robot control, manipulation planning.**

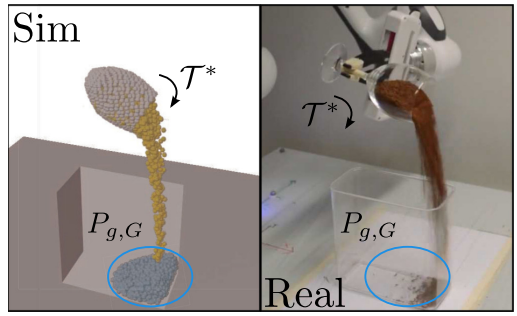


Fig. 1. Granular manipulation using the GNN (**left**) and physical system (**right**). The granular material is manipulated using a rigid-body (cup) following the optimal trajectory $\mathcal{T}^*$ to pour the material in the particles target distribution $P_{g,G}$.

I. Introduction

THE concept of granular materials encompasses materials such as ground coffee, uncooked rice, and sand to name a few. These materials are difficult to manipulate due to their particle nature [1], as the particles of the material interact with each other potentially grouping into individual shapes. Recently, several attempts have been made to solve various tasks involving granular materials ranging from transporting the material [2], shaping [3], [4], to common daily tasks such as gathering, spreading or flipping [5]. A common approach is to use visual feedback to learn to manipulate the material. However, there is a common caveat on these approaches, the interactions of the material particles are not captured. Thus, these methods are not able to plan manipulation of granular materials into a desired configuration when there are complex interactions including e.g. dropping the material and the material interacting with rigid bodies. This is because learning the interactions from visual-feedback is not feasible as the material is not fully observable.

Manuscript received October 22, 2021; accepted February 23, 2022. Date of publication March 10, 2022; date of current version March 25, 2022. This letter was recommended for publication by Associate Editor Pablo Gil and Editor Markus Vincze upon evaluation of the reviewers' comments. This work was supported by the Academy of Finland under Grants 317020 and 328399. *(Neea Tuomainen and David Blanco-Mulero are co-first authors.) (Corresponding author: David Blanco-Mulero.)*

The authors are with the Department of Electrical Engineering and Automation (EEA), Aalto University, 02150 Espoo, Finland (e-mail: neea.tuomainen@aalto.fi; david.blancomulero@aalto.fi; ville.kyrki@aalto.fi).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2022.3158382, provided by the authors.

Digital Object Identifier 10.1109/LRA.2022.3158382

Instead of using models learned from direct observations, simulations of the physical interaction can be used. However, high-fidelity simulators are considerably expensive computationally. Instead, we can learn a computational surrogate model from these simulators to speed up the manipulation planning. Recently, graph neural networks (GNNs) have been proposed for modeling interactions between particles [6]–[9]. Some of these works have explored modeling the dynamics of complex materials such as granular [10] or deformable ones [11], [12]. The GNN based approaches have shown good accuracy and generalisation capabilities, and thus present a promising surrogate model for granular manipulation planning.

In this work, we address the problem of manipulating granular materials into a desired configuration by interacting with a rigid-object (see Fig. 1). In order to capture the interactions of the granular material with the rigid-body, we take on a model-based approach and learn a surrogate model from simulated data based on the idea of Graph Networks-based Simulators (GNS) [10]. We propose to solve the problem of manipulating the material into a desired configuration as a trajectory planning problem, where we use the GNS to rollout the dynamics of the system. The trajectory of the rigid-body is planned by minimising the Wasserstein distance between the simulated final distribution of granular particles and its target.

We evaluate two different aspects of the method. First, we perform an ablation study of the graph attributes and GNN architecture, and provide evidence that the chosen attributes and architecture are able to capture the rigid-body and granular material interactions. Secondly, we evaluate the proposed trajectory

planning method over different target distributions of the granular particles, both using the GNN rollout and a real robot set-up (Fig. 1). As a test-case task we perform pouring of a granular material into a desired configuration. Our results demonstrate that the proposed method effectively pours the material into different desired distributions. Furthermore, we demonstrate that the optimal trajectories in the real robot approximate the expected result using the GNN dynamics model.

The main contributions of this letter are:

1) A GNN based approach for modeling the interaction of granular materials and rigid bodies, and evaluation of the model parameters required to effectively predict the interactions.
2) A solution for trajectory planning based on the Wasserstein distance between the distribution of the manipulated granular material and its target distribution.
3) A demonstration of the framework for pouring granular materials into a desired configuration both in simulated and real scenarios.

II. Related Work

A. Granular Manipulation

The manipulation of granular materials has been studied in different tasks such as robotic excavation [13], scooping [14] and pouring [15]. Recent works have focused on learning to manipulate granular materials using feedback-based methods where data is collected in the real-world [2]–[4], [15], [16]. In [2], a ConvNet model is learned to choose the required actions for transporting beans to a goal shape. Similarly, a learning-based approach was proposed by [16] to grasp granular foods. These methods learn a surrogate model from height or density maps, which does not capture the interaction of the granular particles. Furthermore, collecting data in the real-world is prohibitive and the feedback from the real data does not capture the inherent dynamics of the material. More recently, [5] proposed to learn a Reinforcement Learning policy to manipulate in simulation amorphous materials. However, the policy takes as input the density and height map, facing the same limitation of previous approaches. In contrast, [17] proposed a framework to infer the properties of granular materials using real-world data for fine tuning a simulator to perform robotic tasks.

In the present work, we propose to solve granular manipulation directly in simulation. This alleviates the burden of gathering data in the real-world of previous approaches. In addition, the proposed GNN approach enables learning the granular material interactions, as opposed to the aforementioned surrogate models. Furthermore, our method defines the desired configuration of granular particles as an experimental distribution. This enables planning for target particle configurations potentially more complex than those given by height or density maps.

B. Liquid Manipulation

Similar approaches to those in granular manipulation have been proposed for pouring liquids [18]–[20]. These either used visual feedback to detect the liquid [18], [19] or learned to manipulate from human demonstrations [20]. While dynamics may be learned on from real examples, using real-world data is limited by data collection, as with granular materials, which can be solved by using simulated dynamics. Some of the works in liquid manipulation have considered the dynamics of the material either using high-complexity simulations [21] or simplified dynamics models [22]–[24]. Considering the dynamics of the system allows trajectory planning in more complex manipulation tasks. However, simplified dynamics models might lead to inaccurate trajectories, whereas using high-quality simulations might be costly for complex materials.

C. Learning Particle Interactions Using GNNs

An alternative is to learn the dynamics of the system from high-fidelity simulators using data-driven models [9]–[11]. These can learn from simulation techniques and present the benefit of lower computational cost than the simulators they learn from. Particularly, some GNN approaches have shown outstanding results on learning to simulate complex materials [10], [11]. Sanchez-Gonzalez *et al.* [10] proposed the Graph Networks-based Simulators (GNS) framework, which demonstrated accurate forecasting for different materials such as sand or fluids. However, their work does not involve modeling the interactions when the materials are manipulated. Recently, the GNS framework was used to learn the dynamics of manipulated cloths [12], which suggests it is a good candidate for learning to manipulate granular materials. The closer work to our aim is DPI-Nets [9], as it combines a GNN model for learning material dynamics with model predictive control for controlling a fluid. However, they do not consider the interaction dynamics of pouring the material and the fluid does not have the mechanical properties of granular materials.

III. Method

In this letter, we address the problem of manipulation of granular materials that are shaped into a desired configuration by interacting with them using a rigid-object. We distinguish between two types of materials: the rigid-body that manipulates the granular material, and the granular material itself. We model both materials as particles, which has proven as a good representation for modelling the interaction between different materials [7], [9], [10]. Rigid bodies are modelled as a set of rigid-body particles $\mathbf{p}_r \in P_r$ controllable, where the true position is assumed known. Granular material particles $\mathbf{p}_g \in P_g$ are affected by gravity, their interaction with each other and with rigid-body particles, and the boundaries of the manipulation scene. The granular particles cannot be directly controlled, but we can affect their state through their interaction with controlled rigid-body particles.

A. Graph-Based Representation

Our method makes use of a graph representation where each particle $\mathbf{p}_i$ is represented by a node in a graph G. The nodes are connected via edges if the particles they represent interact with

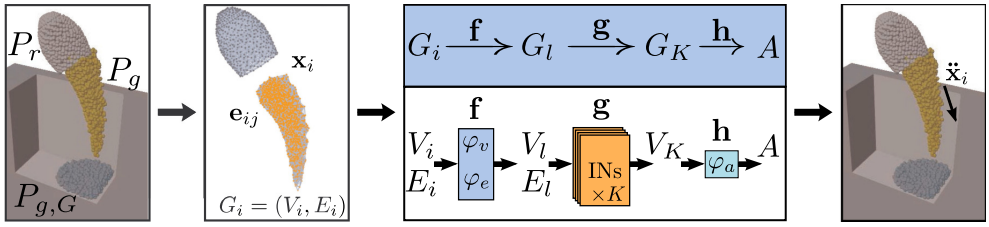


Fig. 2. The initial graph $G_i = (V_i, E_i)$ is created from the particles of the granular material P_g and the rigid-body P_r. The particles are manipulated to match the goal particle distribution $P_{g,G}$. The graph G_i is mapped through the GNN model encoder $\mathbf{f}$, processor $\mathbf{g}$ and decoder $\mathbf{h}$ to predict the acceleration $\ddot{\mathbf{x}}_i \in A$ of the granular material particles.

each other. We assume an interaction exists when two particles are in contact, that is, their distance is lower than a connectivity radius R (see Section IV-B).

We define a graph as $G = (V, E)$, where $\mathbf{v}_i \in V$ is a node attribute vector and $\mathbf{e}_{ij} \in E$ represents an edge attribute vector, where i is the index of the sender node and j is the index of the receiver node. Here, we represent the node set as a matrix $V \in \mathbb{R}^{N\times D}$, where the i-th column represents the node attribute for node i, N represents the number of nodes and D the node attributes. Similarly, we represent the edge set as a matrix $E \in \mathbb{R}^{M\times F}$, where M is the number of edges and F the number of edge attributes. We define the node attribute vector as $\mathbf{v}_i = [\dot{\mathbf{x}}_{t-C}, \ldots, \dot{\mathbf{x}}_{t-1}, \mathbf{b}_t, m, \mathbf{c}_t]$, where $\dot{\mathbf{x}}_{t-C}, \ldots, \dot{\mathbf{x}}_{t-1}$ are the C previous velocities of the particle, $\mathbf{b}_t$ represents the relative distance to the manipulation scene boundaries, m denotes whether the particle is a rigid-body or a granular material particle, and $\mathbf{c}_t$ represents the control input applied to the particle. The control input is defined as the current velocity of the particle $\dot{\mathbf{x}}_t$ for rigid-body particles, and as a zero vector for granular material particles. The edge attribute vector is $\mathbf{e}_{ij} = [\mathbf{s}_{ij}, d_{ij}]$, where $\mathbf{s}_{ij}$ represents the relative displacement between the particles $\mathbf{s}_{ij} = \mathbf{p}_i - \mathbf{p}_j$, and d_{ij} represents their relative distance.

B. Graph Neural Network Model

We learn the dynamics of the granular material particles and their interaction with the rigid-body using a GNN model. The GNN model is used to predict the acceleration of the granular material particles and their trajectories assuming their initial position as well as the position of the rigid-body particles are known. Our network architecture is based on the GNS framework [10]. The model is divided into three sequential parts: (a) an encoder $\mathbf{f} : G_i \mapsto G_l$, (b) a processor $\mathbf{g} : G_l \mapsto G_K$, and (c) a decoder $\mathbf{h} : G_K \mapsto A$. The encoder maps the input graph G_i into a latent graph G_l. The processor propagates the information through K latent graphs, outputting the final latent graph G_K. Finally, the decoder produces as output the acceleration of each particle represented by a node using the node attributes of the final latent graph G_K. Each sequential part is built as follows:

1) Encoder: The encoder takes an input graph $G_i = (V_i, E_i)$ and encodes the node attributes $V_i \in \mathbb{R}^{N\times D_i}$ and edge attributes $E_i \in \mathbb{R}^{M\times F_i}$ into a latent graph $G_l = (V_l, E_l)$, where $V_l \in \mathbb{R}^{N\times D_l}$ and $E_l \in \mathbb{R}^{M\times F_l}$. The latent node and edges are given by

$$\begin{aligned} V_l &= \varphi_v(V_i), \\ E_l &= \varphi_e(E_i), \end{aligned} \tag{1}$$

where φ_v and φ_e are multi-layer perceptrons (MLPs).

2) Processor: The processor consists of K interaction networks (IN) [6] such that the output attributes of one network are summed to its input attributes to produce a new latent graph. The new latent graph is given as input to the next network to propagate information through the graph. The output of the processor is the final latent graph $G_K = (V_K, E_K)$. Each IN is defined as

$$\begin{aligned} \mathbf{e}'_{ij} &= \phi_e(\mathbf{e}_{ij}, \mathbf{v}_i, \mathbf{v}_j), \qquad \bar{\mathbf{e}}'_j = \sum_{\mathbf{e}_{ij}\in E_j} \mathbf{e}_{ij}, \\ \mathbf{v}'_i &= \phi_v(\mathbf{v}_i, \bar{\mathbf{e}}'_i), \end{aligned} \tag{2}$$

where $E_j = \{\mathbf{e}_{ij} \mid i = 1, \ldots, N\}$ is the set of edges whose receiver node is the node with index j. The functions ϕ_e and ϕ_v are MLPs such that each IN has the same architecture but different weights.

3) Decoder: The decoder produces the output for each node using the nodes of the graph produced by the final IN of the processor by

$$\ddot{\mathbf{x}}_i = \varphi_a(\mathbf{v}_i), \quad \mathbf{v}_i \in V_K, \tag{3}$$

where $\ddot{\mathbf{x}}_i \in A$ is the acceleration of the particle and φ_a is an MLP.

The predicted accelerations are used to update the next state and then compute the trajectory. The next state is computed using the semi-implicit Euler integration via

$$\begin{aligned} \dot{\mathbf{x}}_{i,t} &= \dot{\mathbf{x}}_{i,t-1} + \Delta t\, \ddot{\mathbf{x}}_{i,t-1}, \\ \mathbf{x}_{i,t} &= \mathbf{x}_{i,t-1} + \Delta t\, \dot{\mathbf{x}}_{i,t}. \end{aligned} \tag{4}$$

C. Trajectory Planning for Granular Manipulation

Our objective is to control the rigid-body particles P_r by following a trajectory $\mathcal{T}$ that leads the granular particles to settle into a desired goal position $P_{g,G}$. We consider a planning horizon H, where the granular particles start from an initial position $P_{g,0}$, located inside the rigid-body. The rigid-body can be controlled through two set of actions: rotation and translation $\mathbf{u}_t = (\mathbf{R}_t, \mathbf{x}_t)$. The optimal trajectory is defined by Q via-points

$\mathcal{T}^* = (\mathbf{u}_0, \ldots, \mathbf{u}_Q)$. The trajectory is then interpolated to get the actions in the planning horizon H that minimise the distance between the end position of the particles and the desired goal position:

$$\mathbf{u}_0, \ldots, \mathbf{u}_Q = \arg\min d(P_{g,G}, P_{g,H}), \tag{5}$$

where $d(\cdot, \cdot)$ is the distance between the desired and predicted distributions of the particles. We choose this approach as it is unnecessary to transport each particle to the exact position of that same particle in the target distribution, as long as the distributions match. In contrast to existing approaches in particle manipulation that use the Chamfer distance, which measures the distance of every particle to its nearest neighbour [9], [25], we propose to use the Wasserstein distance, which can be understood as the minimum cost of moving the particles from one distribution to the other. We expect this to express better the semantics of material transport. Thus, the trajectory planning is formulated as a discrete optimal transport (OT) problem, where we minimise the quadratic Wasserstein distance:

$$d(P_{g,G}, P_{g,H}) = \left(\sum_{i=0}^{N} \| X_{i,G} - X_{i,H} \|^2 \right)^{1/2}, \tag{6}$$

where $X_{i,G} \in P_{g,G}$ and $X_{i,H} \in P_{g,H}$ are the samples of each empirical distribution. One of the issues when computing the Wasserstein distance for a large number of particles is the computational cost. In order to utilise the Wasserstein distance with large datasets we use the Sinkhorn loss $S_\varepsilon(P_{g,G}, , P_{g,H})$, which is an approximation of OT and reduces the complexity as presented by [26].

To solve the optimisation problem, we adopt a population-based black-box optimiser, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [27], and treat the trajectory planning as a constrained optimisation problem. As constraints, we use the manipulator end-effector limits and the velocity limits used for training the model, similar to [28]. The constraints guide the optimisation to consider trajectories within the training data of the model. The optimisation problem is then

$$\begin{aligned} \min_{\mathbf{u}_0, \ldots, \mathbf{u}_Q} \quad & J(P_{g,G}, P_{g,H}, \mathcal{T}) \\ \text{subject to} \quad & |\mathbf{u}_q - \mathbf{u}_{q-1}| \leq \Delta \mathbf{u}_{max}, \\ & \mathbf{u}_{min} \leq \mathbf{u}_q \leq \mathbf{u}_{max}, \end{aligned} \tag{7}$$

for all $q = 0, \ldots, Q$, where $J(\cdot)$ is the cost function and $\mathbf{u}_{min}$, $\mathbf{u}_{max}$ and $\Delta \mathbf{u}_{max}$ are the boundaries and velocity limits respectively. The cost function, in addition to minimising the distance between experimental distributions, includes a term that penalises the acceleration of the rigid-body to reduces abrupt motions, such that

$$\begin{aligned} J(P_{g,G}, P_{g,H}, \mathcal{T}) = \alpha\, & S_\varepsilon(P_{g,G}, P_{g,H}) \\ & + \beta \sum_{t=2}^{H} \| \mathbf{u}_t - 2\mathbf{u}_{t-1} + \mathbf{u}_{t-2} \|^2 \end{aligned} \tag{8}$$

where α and β are regularisation constants.

Algorithm 1: Trajectory Planning.

Input: $P_{g,0}$, $P_{g,G}$, $\mathcal{T}$
Output: $\mathcal{T}^*$
1: **for** $i = 0 \rightarrow T$ **do**
2: **for** $t = 0 \rightarrow H$ **do**
3: $\mathbf{u}_t^* \leftarrow \mathcal{T}_t$
4: $G_{i,t} \leftarrow$create_graph $(P_{g,t}, \mathbf{u}_t^*)$ ▷ Section III-A
5: $P_{g,t+1} \leftarrow$GNN$(G_{i,t})$
6: **end for**
7: $\mathcal{T} = \arg\min J(P_{g,G}, P_{g,H}, \mathcal{T})$ ▷Using (7)
8: **end for**
9: $\mathcal{T}^* \leftarrow \mathcal{T}$

The optimal trajectory is found by following Algorithm 1. The algorithm takes as input an initial trajectory $\mathcal{T}$ as well as the initial and target particle configurations $P_{g,0}$ and $P_{g,G}$. Then, the next position of the rigid-body particles $P_{r,t}$ is computed after applying the actions $\mathbf{u}_t$. From this, the control inputs $\mathbf{c}_t$ are computed as the difference of the rigid-body position $P_{r,t}$ and $P_{r,t-1}$. Next, the graph $G_{i,t}$ is created as explained in Section III-A. The next position of granular particles $P_{g,t+1}$ is updated on line 5 by following the semi-implicit Euler formulation (4). Once the rollout has finished, the cost $J(\cdot)$ is computed. We perform the optimisation T times, where CMA-ES generates a new population until the maximum number of iterations is reached.

IV. Experimental Results

In this section, we first provide information about the physics engine simulation set-up as well as the GNN training and the trajectory planning routine. Then, we perform an ablation study of the graph attributes and GNN parameters, where we evaluate the accuracy of the models predictions. After that, we assess the trajectory planning routine accuracy for pouring the granular material into different goal positions. The goals of the experiments are twofold: 1) to assess the required parameters for predicting accurately the interaction of granular materials with rigid-bodies, 2) to evaluate whether the proposed method can pour granular material into a desired shape, both in simulation and in a physical system.

A. Simulation Set-Up

We use Taichi-MPM [29] to generate training data for the GNN model. We create a simulated scene where a cup pours granular material into a container, which acts as boundaries of the scene. The container is a $10 \times 20 \times 20$ cm box with an open top allowing the cup and granular material to be located above the container. The cup is modelled as a 3D mesh of $\varnothing 7 \times 10$ cm, which has been populated and replaced by rigid-body particles P_r. The cup is controlled via translation and rotation actions. The granular material is modelled as sand particles which are initialised inside the cup. The simulation consists of $H = 300$ time-steps, where the number of particles is 1945, out of which

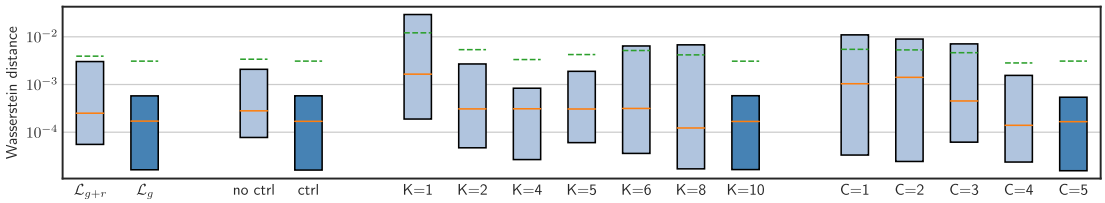


Fig. 3. Boxplot of ablation study using forecast Wasserstein distance. Boxes represent values between 1st and 3rd quartiles, orange horizontal lines represent medians and dotted green lines represent means. Models marked with darker blue represent the GNN model with the proposed graph attributes and architecture.

70% represent the granular material and the rest the rigid-body particles.

B. GNN Training Details

The training dataset consists of 20 simulations where the cup is translated along the Y-axis and rotated along the X-axis. The translation and rotation axes are constrained so that the granular material is always poured inside the container. We provide details about the different trajectories used for generating the training data in the Appendix. In addition, we have included a simulation where the cup only translates and rotates with small Gaussian noise and a simulation of the granular material falling without a cup in the scene.

The connectivity radius used to construct the graph is $R = 0.015$. Before constructing the graph we add a random-walk noise with standard deviation $\sigma = 3 \cdot 10^{-4}$ to the particle positions to improve the acceleration prediction as shown by [10]. The velocities and accelerations are normalised with mean and standard deviation of the dataset. The distance to boundaries $\mathbf{b}$ in node attributes and displacement $\mathbf{s}_{ij}$ in edge attributes are normalised using the connectivity radius R, and $\mathbf{b}$ is clipped to range $[-1, 1]$ as in [10]. The model is trained for 2000 epochs using the Adam optimiser [30]. For the first 500 epochs the learning rate is constant $lr = 10^{-4}$, after which we switch to an exponential learning rate with $\gamma = 0.997$. The training is done using minibatches of size 2. For each MLP in our model, we use two hidden layers where each hidden layer has a size of 128. Each MLP in the encoder and processor are followed by a Layer Normalization [31]. We have experimented using two different loss functions, the number of message-passing steps, including and excluding control inputs $\mathbf{c}_t$, as well as the length of velocity history $\{\dot{\mathbf{x}}_{t-C}, \ldots, \dot{\mathbf{x}}_{t-1}\}$ in the node attributes. The first loss function evaluated, $\mathcal{L}_g$, is the L1-loss that considers only accelerations of granular material particles. The second one, $\mathcal{L}_{g+r}$, is the L1-loss of both granular material and rigid-body particle acceleration. Both losses are averaged over all particles in the minibatch. All GNN models were trained until convergence, which takes 12 days for the model with parameters $K = 10$ and $C = 5$ on a computer with a NVIDIA V100 GPU.

C. Trajectory Planning Details

We use four target distributions $P_{g,G}$ of the granular material for evaluating the optimal trajectory determined using Algorithm 1. The target distributions are generated from simulations that are not included in the training data of the GNN model. The trajectory $\mathcal{T}$ has $Q = 6$ via-points and it is interpolated using a piecewise cubic hermite interpolating polynomial to the rollout horizon length used in the GNN training and test models. For the CMA-ES optimiser we used an initial variance of $\sigma_{\mathcal{T}} = 1.5$ and a population size of 20. In order to speed up the optimisation process, we scaled the trajectory rotation by π and the translation by 0.11 so that both variables are within the same range. The algorithm is run for 150 iterations with five different random seeds, where each random seed took 10 hours on a computer with a V100. The constraints were defined as $\mathbf{u}_{min,max} = \{\pm 2.8973, \pm 0.1\}$ and $\Delta\mathbf{u}_{max} = \{2.1973, 0.02\}$, which matches the maximum values of the training data distribution. We used $\alpha = 1000$ and $\beta = 0.001$ as values for the regularisation terms of the cost function (8). The α parameter is scaled with a high value due to the order of magnitude of the Sinkhorn loss, whereas β is reduced to keep it as a second minimisation objective. The values for these parameters were determined experimentally. We used one of the sinusoidal trajectories used for training (see Appendix) as initial trajectory $\mathcal{T}$, where the final material distribution does not match the test cases target.

D. GNN Ablation Study

We perform an ablation study of the aforementioned parameters. To evaluate the accuracy of the predictions, we use the Wasserstein distance between the predicted granular material particles and the ground-truth simulation. The initial Wasserstein distance for simulations when the particles are relatively still in the cup and the forecast error has not yet accumulated is in the order of 10^{-5}. In our experimental set-up we can consider that a Wasserstein distance of 10^{-2} for the entire rollout presents distant distributions, whereas a value in the order of 10^{-4} accurately matches the rollout particle distributions. The tests are performed by starting from the same initial state onward to create a rollout prediction, where the rollout length is the same as the simulation. The test-set includes 8 simulations that are in the model training domain. The results are summarised in Fig. 3, where the box plot excludes minimum and maximum values, since they are similar in each tested model. In Fig. 3 the best performing model is highlighted in darker blue, which was used to rollout the dynamics in the trajectory planning.

1) Training loss: We experimented on two loss functions $\mathcal{L}_g$ and $\mathcal{L}_{g+r}$. As shown in Fig. 3, the loss $\mathcal{L}_g$ presents a slightly

lower median and mean. The most noticeable difference between the two models is their quartiles, which are lower for the loss $\mathcal{L}_g$. This provides evidence that including the acceleration of rigid-body particles in the loss is unnecessary as it affects adversely when learning granular particle behaviour.

2) Control inputs: We investigated whether including control inputs $\mathbf{c}_t$ as a node attribute improves the model accuracy. Considering the quartiles of the models in Fig. 3 we can observe lower loss for the model that includes control inputs. This is to be expected since it gives additional information about the movement of the rigid-body, which improves the predictions of the granular material particles.

3) Message-passing steps: We also experimented on the number of INs in the processor. [10] suggests that using a higher number of message-passing steps provides better results. We selected $K = 10$ as maximum and evaluated how models with smaller values perform. The range of values selected was $K \in \{1, 2, 4, 5, 6, 8, 10\}$. We can notice in Fig. 3 that the highest K value indeed does perform best. The message-passing steps affect on how far the particle information reaches in the graph and, thus, how many interactions affect the particles. With $K = 4$ the performance is not significantly worse than $K = 10$. This suggests that $K = 4$ message-passing steps is enough to capture the system behaviour. However, the third quartile of $K = 6$ and $K = 8$ are higher than for $K = 4$, which suggests otherwise.

4) Node velocity history length: Finally, we assessed how the number of previous velocities included in the node attribute affects the performance. The values we tested were $C \in \{1, 2, 3, 4, 5\}$. One would hypothesise that $C = 3$ would be sufficient to capture the dynamics, as granular materials can be approximated via a second-order system. However, the performance of $C = 4$ and $C = 5$ is significantly better than the other models. This suggests that knowing the three previous velocities is not enough to accurately predict rollout dynamics in our system.

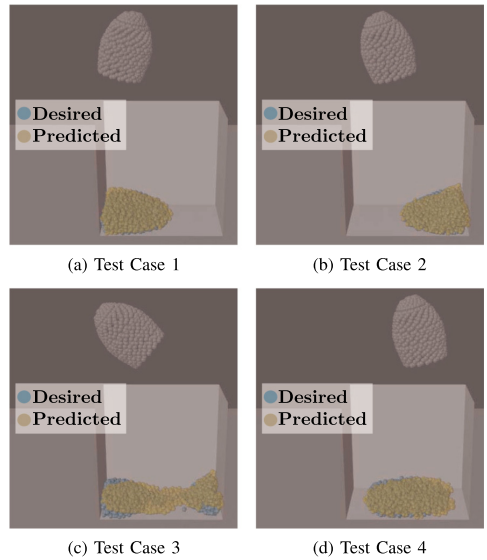
Overall, the results show that including the control input and the $\mathcal{L}_g$ loss function yield drastically lower values compared to the other variants, as the interquartile range improves by almost one order of magnitude. The results for the number of message-passing steps showed that with $K = 10$ the mean and interquartile range are lowest within the tested models. The node velocity history length $C = 5$ provided slightly higher mean and median than $C = 4$, but presented lowest interquartile range. For this reason we choose to use $C = 5$, $K = 10$, control inputs and $\mathcal{L}_g$ as loss function for the GNN model in the trajectory planning. This model speeds up the computation of the rollout dynamics by 20 times the high-fidelity simulator.

E. Trajectory Planning Results

The Sinkhorn loss for each of the four test cases is shown in Table I. The initial loss $S_\varepsilon(P_{g,G}, P_{g,0})$ measures the distance between the distribution of the particles in the initial configuration of the cup and its target. We also provide the loss for the end distribution of the particles following the initial trajectory $P_{g,H}$. This is used as a reference to highlight the improvement on matching the target distribution using the proposed algorithm. We can notice that the order of magnitude improves by an order of two following the optimal trajectory P_{g,H^*}. Our experiments showed that two particles distributions are well matched when the Wasserstein distance is in the order of 10^{-6}, whereas distances larger than 10^{-4} are far from the target distribution. The results indicate that our method is able to consistently reduce the distance between the end and target particle distributions by an order of magnitude. The particles distribution after following the optimal trajectory as well as the desired distribution for each test case are shown in Fig. 4. The final distribution for the test cases one, two and four nicely overlays the desired configuration of the material. On the other hand, the third test case Sinkhorn loss is reduced by an order of magnitude from the initial loss (see Table I). However, the final configuration does not accurately match the target shape. We hypothesise that with a larger population of CMA-ES the Algorithm 1 would find a better solution.

TABLE I
SINKHORN LOSS BETWEEN THE TARGET SHAPE $P_{g,G}$, THE INITIAL MATERIAL DISTRIBUTION $P_{g,0}$, THE FINAL DISTRIBUTION FOLLOWING THE INITIAL TRAJECTORY $P_{g,H}$, AND THE OPTIMAL TRAJECTORY P_{g,H^*}, FOR EACH TEST SET OVER FIVE DIFFERENT RANDOM SEEDS

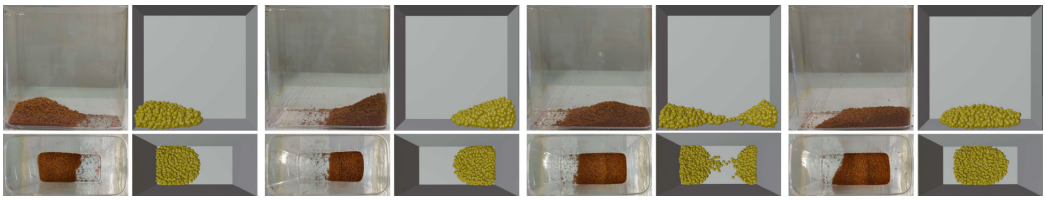
Test	$S_\varepsilon(P_{g,G}, P_{g,0})$	$S_\varepsilon(P_{g,G}, P_{g,H})$	$S_\varepsilon(P_{g,G}, P_{g,H^*})$
1	$3.90 \cdot 10^{-2}$	$1.43 \cdot 10^{-4}$	$(2.09 \pm 0.80) \cdot 10^{-6}$
2	$3.87 \cdot 10^{-2}$	$5.68 \cdot 10^{-3}$	$(1.16 \pm 0.36) \cdot 10^{-6}$
3	$3.93 \cdot 10^{-2}$	$2.29 \cdot 10^{-3}$	$(1.88 \pm 0.53) \cdot 10^{-4}$
4	$3.83 \cdot 10^{-2}$	$1.57 \cdot 10^{-3}$	$(2.69 \pm 0.78) \cdot 10^{-6}$



(a) Test Case 1 (b) Test Case 2

(c) Test Case 3 (d) Test Case 4

Fig. 4. **Qualitative results** for each test case showing the granular material final configuration (predicted) and goal configuration (desired) after following an optimal trajectory.



(a) Test Case 1 (b) Test Case 2 (c) Test Case 3 (d) Test Case 4

Fig. 5. **Qualitative results** of pouring ground coffee. The images show front and top views of the particles distribution after following the optimised trajectories on the real robot **(left)** and target distribution using the ground-truth simulation **(right)**.

We noticed that our GNN model was unable to generalise to rotation and angular velocities greater beyond those seen during training, where the model would predict the material leaking from the rigid-body (see supplementary material). This limited the range of actions the trajectory planning algorithm could apply. Extensions of this work could focus on finding a model that generalises to any rigid-body action or training the model with a wider range of actions.

F. Granular Manipulation in Real-World

We perform experiments on a real set-up where we use a Franka Emika Panda robot to manipulate the cup. We selected ground coffee as the granular material to pour into the container. We executed the best four trajectories optimised in simulation using the real robot and qualitatively evaluated the end result of the ground coffee particles. Prior to running the trajectories, we transform the reference frame of the rigid-body into the robot end-effector reference frame. The robot is controlled via a Cartesian controller [32], where each trajectory point is interpolated from the simulation frequency (100 Hz) to the robot frequency (1KHz).

We show the qualitative results in Fig. 5. We can observe that the first and second test cases, (Fig. 5(a), (b)), match closely the desired distribution. In contrast, the fourth test case, Fig. 5(d), slightly differs from the target distribution. Furthermore, the third test case, Fig. 5(c), which required pouring in both left and right side of the container was the distribution furthest to the target. In addition, in the supplementary material we provide the real-world performance of the predefined trajectories used for generating the target distributions. These show that the distributions generated in the simulator do not exactly match the real-world, with some spilling of the material for the first test case, and a slight mismatch for the third and fourth test cases. This provides further evidence of the sim-to-real gap between the simulated and the real material.

As a summary, our results provide evidence that the proposed method is able to optimise the trajectory of the cup and pour the material in the desired configuration using the GNN rollout. In addition, the results on the real-world highlight that the proposed method is able to effectively pour the material in some of the presented test cases without gathering data from the real-world to train the model. However, some of the test cases were far from the results using the GNN rollout. This is at least partially caused by imperfect simulation. Particles sometimes leak through the rigid body in the simulation, which is also illustrated in the supplementary material. We hypothesise that with a higher fidelity simulation that resembles better the granular material, such as the one proposed by [17], the proposed model would be able to match better the real material and pour precisely the granular material.

V. Conclusion

In this letter, we introduced a model-based approach for manipulating granular materials into a desired configuration by interacting with a rigid-object. The proposed method uses a GNN to learn the interactions of the granular material from simulations, as well as the interactions of the material with the rigid-body that manipulates it. We presented a trajectory planning routine that uses the GNN model to rollout the dynamics and optimises the trajectory by minimising the Wasserstein distance between the rollout result and the goal distribution of the poured material. We provided a study of the GNN model architecture as well as the graph attributes that can accurately predict the interactions between the material and the manipulated rigid-body. Our results show that the planning routine is able to find the required motions to pour the material in the desired configuration. We also demonstrated that the optimal trajectories used with the GNN rollouts are able to effectively pour the material in a real-world set-up.

One of the main limitations of using data from a simulator for planning is the simulation-to-real world gap. Naturally, using a high fidelity simulation is beneficial but the calibration of any simulation with a particular real world setting is difficult. Thus, it is an important avenue for future to study how simulation can be combined with limited data from the real world in order to achieve complex granular manipulation tasks across a variety of materials and tasks.

Acknowledgment

The authors would like to thank Fares J. Abu-Dakka and Gokhan Alcan of Aalto University for their help and support in this work. The authors would also like to acknowledge the computational resources provided by the Aalto Science-IT project.

Appendix
Additional Training Details

We defined four sets of trajectories equations to generate the data for training the Graph Neural Network:

1) The first trajectory rotates following a cosine function with zero-mean Gaussian noise. The rotation stops for several timesteps once the maximum rotation is reached and then rotates back to upright position. The translation is given by a zero-mean Gaussian noise which is added to the position at the previous timestep.
2) The second trajectory follows sinusoidal to translate and rotate the cup.
3) The third trajectory is given by a linear translation and rotation that tilt the cup a single time in both directions with constant velocity.
4) The last trajectory is given by a linear translation and rotation with Gaussian noise $\mathcal{N}(0, 10^{-3})$ and $\mathcal{N}(0, 1)$ respectively. The trajectory is defined as 300 time-steps where the first third is limited to translation, the second to rotation, and the last to going back to the origin position and orientation.

All the trajectories have randomised direction, maximum rotation angles and frequency for the sine and cosine trajectories, and maximum velocities for the linear trajectory. The specific details as well as the open source code can be found at.[1]

REFERENCES

[1] Z.-Y. Yin, P.-Y. Hicher, C. Dano, and Y.-F. Jin, "Modeling mechanical behavior of very coarse granular materials," *J. Eng. Mechanics*, vol. 143, no. 1, 2017, Art. no. C 4016006.

[2] C. Schenck, J. Tompson, S. Levine, and D. Fox, "Learning robotic manipulation of granular media," in *Proc. Conf. Robot Learn.*, 2017, pp. 239–248.

[3] A. Cherubini, J. Leitner, V. Ortenzi, and P. Corke, "Towards vision-based manipulation of plastic materials," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 485–490.

[4] A. Cherubini, V. Ortenzi, A. Cosgun, R. Lee, and P. Corke, "Model-free vision-based shaping of deformable plastic materials," *Int. J. Robot. Res.*, vol. 39, no. 14, pp. 1739–1759, 2020.

[5] Y. Zhang, W. Yu, C. K. Liu, C. Kemp, and G. Turk, "Learning to manipulate amorphous materials," *ACM Trans. Graph.*, vol. 39, no. 6, 2020, Art. no. 189.

[6] Y. Li, Y. Liang, and A. Risteski, "Recovery guarantee of non-negative matrix factorization via alternating updates," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Red Hook, NY, USA: Curran Associates, 2016.

[7] D. Mrowca *et al.*, "Flexible neural representation for physics prediction," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, Ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, pp. 8813–8824.

[8] P. W. Battaglia *et al.*, "Relational inductive biases, deep learning, and graph networks," 2018, *arXiv:1806.01261*.

[9] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba, "Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids," in *Proc. Int. Conf. Learn. Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=rJgbSn09Ym

[10] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, "Learning to simulate complex physics with graph networks," in *Proc. 37th Int. Conf. Mach. Learn.*, (Proceedings of Machine Learning Research Series) H. D. III and A. Singh, Eds., vol. 119, 2020, pp. 8459–8468.

[11] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia, "Learning mesh-based simulation with graph networks," in *Proc. Int. Conf. Learn. Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=roNqYL0_XP

[12] X. Lin, Y. Wang, Z. Huang, and D. Held, "Learning visible connectivity dynamics for cloth smoothing," in *Proc. 5th Conf. Robot Learn.*, (Proceedings of Machine Learning Research Series) A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164, 2022, pp. 256–266.

[13] S. Singh, "Learning to predict resistive forces during robotic excavation," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 2, 1995, pp. 2102–2107.

[14] S. Sarata, H. Osumi, Y. Kawai, and F. Tomita, "Trajectory arrangement based on resistance force and shape of pile at scooping motion," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 4, 2004, pp. 3488–3493.

[15] S. Clarke, T. Rhodes, C. G. Atkeson, and O. Kroemer, "Learning audio feedback for estimating amount and flow of granular material," in *Proc. 2nd Conf. Robot Learn.*, vol. 87, Oct. 29–31, 2018, pp. 529–550.

[16] K. Takahashi, W. Ko, A. Ummadisingu, and S.-I. Maeda, "Uncertainty-aware self-supervised target-mass grasping of granular foods," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 2620–2626.

[17] C. Matl, Y. Narang, R. Bajcsy, F. Ramos, and D. Fox, "Inferring the material properties of granular media for robotic tasks," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 2770–2777.

[18] C. Schenck and D. Fox, "Visual closed-loop control for pouring liquids," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 2629–2636.

[19] C. Do and W. Burgard, "Accurate pouring with an autonomous robot using an RGB-D camera," in *Proc. Int. Conf. Intell. Auton. Syst.*, Springer, 2018, pp. 210–221.

[20] Y. Huang, J. Wilches, and Y. Sun, "Robot gaining accurate pouring skills through self-supervised learning and generalization," *Robot. Auton. Syst.*, vol. 136, 2021, Art. no. 103692.

[21] Z. Pan, C. Park, and D. Manocha, "Robot motion planning for pouring liquids," *Proc. Int. Conf. Automated Plan. Scheduling*, vol. 26, no. 1, 2016.

[22] W. Aribowo, T. Yamashita, K. Terashima, and H. Kitagawa, "Input shaping control to suppress sloshing on liquid container transfer using multi-joint robot arm," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 3489–3494.

[23] M. Reyhanoglu and J. H. Rubio, "Nonlinear modeling and control of slosh in liquid container transfer via a PPR robot," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 18, no. 6, pp. 1481–1490, 2013.

[24] Z. Pan and D. Manocha, "Motion planning for fluid manipulation using simplified dynamics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 4224–4231.

[25] S. Chen, X. Ma, Y. Lu, and D. Hsu, "Ab initio particle-based object manipulation," in *Proc. Robotics, Sci. Syst.*, 2021. [Online]. Available: http://www.roboticsproceedings.org/rss17/p071.html

[26] J. Feydy, T. Séjourné, F.-X. Vialard, S.-I. Amari, A. Trouve, and G. Peyré, "Interpolating between optimal transport and MMD using Sinkhorn divergences," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 2681–2690.

[27] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2001.

[28] F. J. Abu-Dakka, F. J. Valero, J. L. Suñer, and V. Mata, "A direct approach to solving trajectory planning problems using genetic algorithms with dynamics considerations in complex environments," *Robotica*, vol. 33, no. 3, pp. 669–683, 2015.

[29] Y. Hu *et al.*, "A moving least squares material point method with displacement discontinuity and two-way rigid body coupling," *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018, Art. no. 150.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, Y. Bengio and Y. LeCun, Eds. San Diego, CA, USA, 2015. [Online]. Available: https://openreview.net/forum?id=8gmWwjFyLj

[31] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

[32] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE J. Robot. Autom.*, vol. 3, no. 1, pp. 43–53, Feb. 1987.

[1] https://sites.google.com/view/granular-gnn-manipulation

Publication III

David Blanco-Mulero, Gokhan Alcan, Fares J. Abu-Dakka and Ville Kyrki. QDP: Learning to Sequentially Optimise Quasi-Static and Dynamic Manipulation Primitives for Robotic Cloth Manipulation. Accepted for publication in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Detroit, US., October 2023.

QDP: Learning to Sequentially Optimise Quasi-Static and Dynamic Manipulation Primitives for Robotic Cloth Manipulation

David Blanco-Mulero[1], Gokhan Alcan[1], Fares J. Abu-Dakka[2], Ville Kyrki[1]

***Abstract*— Pre-defined manipulation primitives are widely used for cloth manipulation. However, cloth properties such as its stiffness or density can highly impact the performance of these primitives. Although existing solutions have tackled the parameterisation of pick and place locations, the effect of factors such as the velocity or trajectory of quasi-static and dynamic manipulation primitives has been neglected. Choosing appropriate values for these parameters is crucial to cope with the range of materials present in house-hold cloth objects. To address this challenge, we introduce the Quasi-Dynamic Parameterisable (QDP) method, which optimises parameters such as the motion velocity in addition to the pick and place positions of quasi-static and dynamic manipulation primitives. In this work, we leverage the framework of Sequential Reinforcement Learning to decouple sequentially the parameters that compose the primitives. To evaluate the effectiveness of the method we focus on the task of cloth unfolding with a robotic arm in simulation and real-world experiments. Our results in simulation show that by deciding the optimal parameters for the primitives the performance can improve by 20% compared to sub-optimal ones. Real-world results demonstrate the advantage of modifying the velocity and height of manipulation primitives for cloths with different mass, stiffness, shape and size. Supplementary material, videos, and code, can be found at `https://sites.google.com/view/qdp-srl`.**

I. Introduction

The broad variety of fabric materials that are handled by humans everyday presents several challenges when manipulated by robotic systems. As an example, the variability of cloth materials in house-hold objects in terms of shape, stiffness, elasticity, and mass [1], requires humans to perform a set of diverse manipulation actions such as those used for dressing assistance [2] or flattening, folding and twisting cloths [3]. However, it is not trivial to transfer such skills to robot manipulators, as manipulation primitives are often manually designed for a specific application [4], or a set of pre-tuned primitives is used [5]. Thus, to cope with a wide range of cloths, robotic systems should integrate learning algorithms that autonomously decide the parameter values of these manipulation primitives.

The manipulation primitives used in robotic cloth manipulation fall into two categories: *quasi-static*, such as the pick-and-place primitive [6]; and *dynamic*, which involve the forces of acceleration of the manipulator [7], such as the fling primitive [4]. Quasi-static primitives have been extensively used for cloth folding and unfolding [5], [6], [8], [9], [10], [11], where different algorithms have been proposed for deciding the end-effector pick-and-place positions. However, parameters such as the specific trajectory height or velocity of the primitive have been neglected. This is crucial in applications such as cloth manipulation in-contact with a surface, where the size of the cloth will drastically affect the manipulation result, *e.g.,* bigger cloths will have more contact if they follow a trajectory with lower height (see Fig 1). Hence, these parameters should be taken into account to cope with a diverse set of cloth materials and sizes.

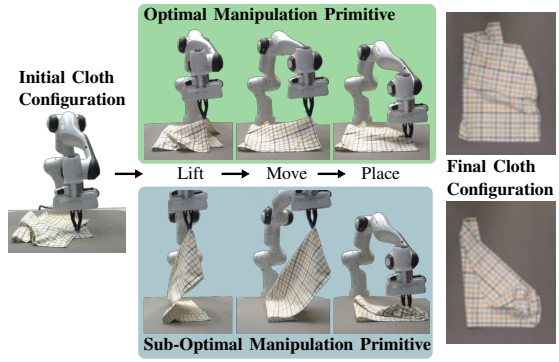


Fig. 1: QDP sequentially optimises the manipulation primitive parameter values to achieve better cloth configurations (*green*) compared to using sub-optimal parameter values (*blue*) for a manipulation primitive such as pick-and-place.

The main contribution of this work is a method to learn a visual policy that can determine the optimal parameters of manipulation primitives for cloth manipulation. We refer to our method as QDP, short for the Quasi-Dynamic Parameterisable manipulation method. Our key idea is to find the optimal parameters by following a sequential decision approach. We take inspiration from the Sequential RL (SRL) framework [12] to sequentially assign an action space to each primitive parameter. Here, each parameter of the primitive is informed by the previous proposed parameter. This enables learning the relationship between the primitive parameters and their impact on the manipulation performance. Our method works in a joint action space: the spatial action space of pick-and-place locations; and the parameter action space,

This work was financially supported by the Academy of Finland grant numbers 317020 and 328399. Abu-Dakka, F. is supported by the European project euROBIN under grant agreement No 101070596.

[1]David Blanco-Mulero, Gokhan Alcan and Ville Kyrki are with School of Electrical Engineering, Aalto University, Finland.) (e-mail: david.blancomulero@aalto.fi

[2]Munich Institute of Robotics and Machine Intelligence, Technische Universität München, 80992 München, Germany. Part of the research presented in this work has been conducted when Abu-Dakka, F. was at Intelligent Robotics Group, EEA, Aalto University, 02150 Espoo, Finland

Manuscript 1559 submitted to 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 1, 2023.

that defines parameters such as the velocity of the manipulation primitive. In order to evaluate the effectiveness of the sequential decision of parameter values, we evaluate our proposed method in simulation and real-world experiments in the task of cloth unfolding by using a single robotic arm. Our contributions include:

- Introducing QDP, a novel approach that can optimise the parameters of manipulation primitives, decoupling the pick-and-place decisions as well as additional primitive parameters, without supervision or hand-labeled data during training,
- An analysis of the performance of QDP on the cloth unfolding task, showing the superior performance of the proposed method compared to baselines, as well as its ability to find optimal parameters for both quasi-static and dynamic manipulation primitives,
- For the first time, a real-world evaluation of different manipulation primitives on a public cloth unfolding benchmark.

II. Related Works

Cloth Manipulation research has extensively focused on visual-based methods to plan the manipulation actions for a variety of tasks such as folding or unfolding [5], [6], [8], [9], [13]. Prior work has actively studied how to decide the pick-and-place locations of pre-defined manipulation primitives. Recent works have proposed to use spatial action maps [14] to decide the pick position, and rotating and scaling the input image to decide the place position [4], [8], [11]. However, the decision of the place position in these approaches is limited to a discrete combination of rotation angles and scales, thus restricting the primitive action space. In contrast, our work treats both pick and place as actions in a spatial action space, resulting in a more flexible primitive action space. Alternatively, other works do not suffer from a restricted place location by defining a conditional action space [6], or action spaces that contain both the pick and place locations [9], [10], [15]. However, all these works suffer from the same caveat, they focus on determining only the pick and place of pre-defined manipulation primitives, neglecting other parameters such as the velocity at which they are executed. The velocity at which the manipulation actions are executed has an impact in tasks such as cloth folding [16], where dynamic manipulation actions can be adapted according to the fabric material. In this work, we decide parameters such as the velocity for a dynamic manipulation primitive, and the height for a pick-and-place primitive, thus generating a more diverse set of manipulation actions to cope with cloths of varied sizes and materials.

Sequential RL was first introduced by Metz et. al [17] to enable learning in high dimensional spaces by decomposing the action space in simpler sub-action spaces, proposing the Sequential DQN (SDQN) method. This work was then extended to learn policies in SE(3) action spaces [12], where they introduced the idea of augmented state that we follow in the proposed method. The augmented state approach has shown success applied to learning grasping [18] and manipulation [19]. Although prior works have extended the action spaces to SE(2) and SE(3), they have not considered other action spaces such as the manipulation primitive parameter action space. In this work, we follow the SDQN method to compose an action space of pick and place location, as well as a parameter space that can modify the manipulation primitives behaviour; showing its effectiveness in cloth manipulation. In addition, prior work computed the pick and place actions at different time steps, whereas in this work we sequentially decide both, thus deciding the place location without interacting with the environment.

III. Background

A. Augmented State Representation

The SRL augmented state approach introduced by Wang. et al. [12] is formulated as follows. We define an MDP $\mathcal{M} = \langle S, A, R, T, \gamma \rangle$ with state space S, action space A, reward function R, transition dynamics T and discount factor $\gamma \in [0, 1]$. We denote this MDP as the *top-level* MDP. The top-level MDP action space is composed by a set of *sub-action* spaces $A = A_1 \times \cdots A_n$, where n is the number of sequential actions and $\mathbf{a}_i \in A_i$ is an action in the i-th sub-action space. Thus, the actions $\mathbf{a} = [\mathbf{a}_1, \cdots, \mathbf{a}_n] \in A$ are composed by a set of multiple sequential choices. A new MDP is defined by an augmented state representation $\bar{\mathcal{M}} = \langle \bar{S}, \bar{A}, R, T, \gamma \rangle$, with action space $\bar{A} = A_1 \cup \cdots \cup A_i$ and state space $\bar{S} = S_1 \cup \cdots \cup S_i$. The decision making process starts with the initial *sub-state* space $S_1 = S$ and sub-action space A_1. The consecutive sub-state spaces take into account the sequence of previous sub-action spaces $S_i = S \times A_1 \times \cdots A_{i-1}$ for $i \geq 2$. Following this formulation, the transition $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ can be rewritten as $p(\mathbf{s}'|\mathbf{s}, (\mathbf{a}_1, \cdots, \mathbf{a}_n))$. Furthermore, the n-th transition has the same reward as the MDP $\mathcal{M}$, whereas the previous transitions where $i < n$ have zero rewards. As an example, for a sequence of $n = 3$ actions, the partial transitions $(\mathbf{s}, \mathbf{a}_1)$ and $(\mathbf{s}, \mathbf{a}_1, \mathbf{a}_2)$ have zero rewards, whereas the last transition $(\mathbf{s}, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ has the same reward as the top-level MDP.

B. Sequential DQN

Sequential DQN [17] decomposes the action space A into N sequential sub-actions to decrease computational complexity.

The action-value functions are defined as $Q(\mathbf{s}, \mathbf{a})$ for the top-level MDP, and as $Q_i(\mathbf{s}, \mathbf{a}_i)$ for the low-level MDPs. In order to have consistent Q-values between the top-level and the low-level MDPs, [17] use a discount factor of $\gamma = 0$ for every $i < n$. Thus, for a sequence of states in the top-level MDP $\mathbf{s}, \mathbf{s}' \in S$, the action-value functions satisfy

$$\begin{aligned} Q(\mathbf{s}, \mathbf{a}) &= Q_1(\mathbf{s}, \mathbf{a}_1), \\ Q(\mathbf{s}', \mathbf{a}) &= Q_n(\mathbf{s}, \mathbf{a}_n) = Q_n(\mathbf{s}, (\mathbf{a}_1, \cdots, \mathbf{a}_n)). \end{aligned} \tag{1}$$

Here, the action-value functions are approximated by neural networks. The structure of the neural-networks proposed in this work is explained in detail in Section IV.

Manuscript 1559 submitted to 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 1, 2023.

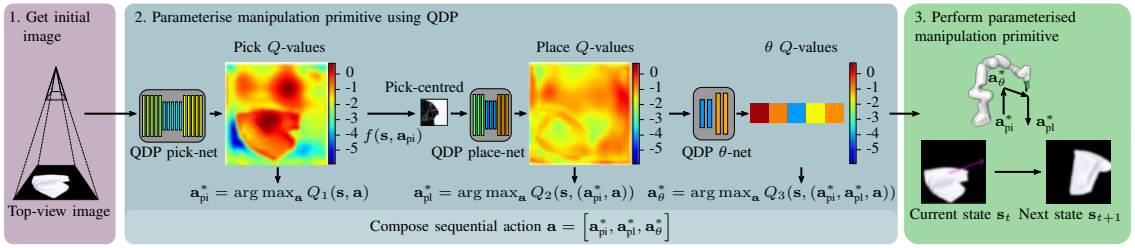


Fig. 2: The proposed Quasi-Dynamic Parameterisable approach starts by getting a top-view image of the cloth. Then, to find the optimal manipulation primitive parameters, a sequential action $\mathbf{a}$ is composed from the sub-action output of three different networks: QDP pick-net, predicts the optimal pick position $\mathbf{a}^*_{\text{pi}}$; QDP place-net, predicts the place location $\mathbf{a}^*_{\text{pl}}$; and QDP θ-net, predicts additional primitive parameters $\mathbf{a}^*_{\theta}$, such as the primitive velocity. Each sub-action takes into account the previous information via encodings, e.g. the place sub-action accounts for the pick location using a pick-centred image. Finally, the manipulation primitive is executed with the optimal parameters placing the cloth on a new state.

IV. Method

QDP learns a visual policy that can determine the optimal parameters of a manipulation primitive for cloth manipulation (see Figure 2). Given an initial top-view image, our proposed approach determines the optimal pick, $\mathbf{a}^*_{\text{pi}} \in A_1$; place, $\mathbf{a}^*_{\text{pl}} \in A_2$; and other parameters of the manipulation primitive, $\mathbf{a}^*_{\theta} \in A_3$. These sub-actions are determined sequentially for computational feasibility.

A. Sequential Parameter Choice via SDQN

The state is defined as a gray-scale image $\mathbf{s} \in \mathbb{R}^{D \times D}$, where D is the height and width dimension of the image. By decomposing the pick and place into multiple decisions we can decide the place position based on the pick action. Thus, the first decision in our sequential RL setting is the pick position which is determined as

$$\mathbf{a}^*_{\text{pi}} = \arg\max_{\mathbf{a}_{\text{pi}}} Q_1(\mathbf{s}, \mathbf{a}_{\text{pi}}), \tag{2}$$

where the optimal pick $\mathbf{a}^*_{\text{pi}} \in \mathbb{N}^2$ is the pixel position in the image space. The action-value function Q_1 is approximated by a neural network that also provides an encoding $g(\mathbf{s})$ of the state-space. Then, for deciding the place position we use the augmented state $S_2 = S \times A_1$. In order to encode the sub-action $\mathbf{a}_{\text{pi}}$ we use a mapping $f : (\mathbf{s}, \mathbf{a}_{\text{pi}}) \mapsto \mathcal{I}_{\text{pick}}$ that creates a pick-centred image $\mathcal{I}_{\text{pick}} \in \mathbb{R}^{E \times E}$, which is a cropped image centred in the pick position, similar to [12]. Instead of using the state $\mathbf{s}$ as input to the action-value function Q_2 we use an encoding $g(\mathbf{s})$, which is part of the output from the neural network that approximates Q_1. The place sub-action is then computed as

$$\mathbf{a}^*_{\text{pl}} = \arg\max_{\mathbf{a}_{\text{pl}}} Q_2(g(\mathbf{s}), f(\mathbf{s}, \mathbf{a}_{\text{pi}}), \mathbf{a}_{\text{pl}}), \tag{3}$$

where $\mathbf{a}^*_{\text{pl}} \in \mathbb{N}^2$, same as the pick sub-action.

Similarly, the last parameter sub-action augmented state $S_3 = S \times A_1 \times A_2$ re-uses information from both previous sub-actions. Here, the action-value function Q_2 is taken as input to provide information of the place sub-action. In order to re-use and share previous information, the third augmented state is given by the encoding $g(\mathbf{s})$ as well as an encoding of the pick-centred image. More details about the encodings are given in Section IV-B. The parameter sub-action is then computed as

$$\mathbf{a}^*_{\theta} = \arg\max_{\mathbf{a}_{\theta}} Q_3(g(\mathbf{s}), f(\mathbf{s}, \mathbf{a}_{\text{pi}}), Q_2(\mathbf{s}_2, \mathbf{a}_{\text{pl}}), \mathbf{a}_{\theta}), \tag{4}$$

where $\mathbf{a}^*_{\theta} \in \mathbb{N}^1$, and the parameter values are defined in a different range of values for each manipulation primitive as detailed in Section V-C.

Finally, the top-level MDP action is composed by the three sub-actions $\mathbf{a} = \left[\mathbf{a}^*_{\text{pi}}, \mathbf{a}^*_{\text{pl}}, \mathbf{a}^*_{\theta}\right]$ that have been sequentially computed re-using previous information throughout the augmented state space.

B. QDP-Net

We denote the first part of the network as QDP pick-net, which follows a U-net structure [20] similar to [12] with skip connections so as to not lose information throughout the encoder-decoder structure. The QDP pick-net performs the mapping $h_1 : \mathbf{s} \to (g(\mathbf{s}), Q_1(\mathbf{s}, \mathbf{a}_{\text{pi}}))$, providing the encoding $g(\mathbf{s})$ as well as the Q-values for the pick sub-action. This is followed by the QDP place-net mapping

$$h_2 : (g(\mathbf{s}), f(\mathbf{s}, \mathbf{a}_{\text{pi}})) \to (w(f(\mathbf{s}, \mathbf{a}_{\text{pi}})), Q_2(\mathbf{s}, \mathbf{a}_{\text{pl}})), \tag{5}$$

which combines the Q-pick encoding with the pick-centred image to output a second encoding $w(f(\mathbf{s}, \mathbf{a}_{\text{pi}}))$, as well as the Q-values for the place sub-action via a decoder structure.

Finally, the last part of the network QDP θ-net follows the mapping

$$h_3 : (g(\mathbf{s}), w(f(\mathbf{s}, \mathbf{a}_{\text{pi}})), Q_2(\mathbf{s}_2, \mathbf{a}_{\text{pl}})) \to Q_3(\mathbf{s}, \mathbf{a}_{\theta}), \tag{6}$$

which re-uses previous encodings information and, after a convolutional and fully connected network structure, outputs the Q-values for the third sub-action. Thus, the proposed network used in QDP is the combination of the three mappings

$$h : \mathbf{s} \xrightarrow{h_1, h_2, h_3} (Q_1(\mathbf{s}, \mathbf{a}_{\text{pi}}), Q_2(\mathbf{s}, \mathbf{a}_{\text{pl}}), Q_3(\mathbf{s}, \mathbf{a}_{\theta})),$$

which outputs the Q-values for the three sub-actions and sub-state spaces, used to determine the optimal sub-actions.

C. Training procedure

In this work we focus on the task of cloth unfolding. The objective is to maximise the area of the cloth to achieve a smooth or flattened state. Thus we define the reward for training the policy as

$$r(\mathbf{s}) = C(\mathbf{s})/C_{\text{max}}, \tag{7}$$

where $C(\mathbf{s})$ is a function that computes the current area of the cloth based on the image pixels, and C_{max} is the maximum area of the cloth. Each of the Q-networks is optimised by minimising the Huber loss [21]

$$\mathcal{L} = \mathbb{E}(y - Q(\mathbf{s}, \mathbf{a}; \phi)), \tag{8}$$

where y is the optimisation target, ϕ denotes the parameters of the network used for evaluation, and ψ the parameters of the SDQN target network. The target is defined as the 1-step TD return

$$y = r(\mathbf{s}) + \gamma Q_3(\mathbf{s}', (\mathbf{a}'_{\text{pi}}, \mathbf{a}'_{\text{pl}}, \mathbf{a}'_{\theta}); \psi). \tag{9}$$

Note that for all of the networks we use the same target, that is, the Q-values of the last Q-network, as it has been proven to be more sample efficient than using different targets per each network [12].

During training, in order to trade-off between exploration and exploitation we use ε-greedy. In addition, to improve the sim-to-real transfer we also perform Domain Randomisation [22] and randomise the internal parameters of the cloth, including the stiffness, mass, size, colour, and initial cloth configuration. Furthermore, the network parameters are initialised randomly and no prior is required. Finally, in order to reduce the number of network parameters and based on the results from prior work on cloth manipulation [16], we use a grayscale image $\mathbf{s} \in \mathbb{R}^{128\times128}$ as input to the network. The observations are rotated at each time-step to help generalise the network to different cloth states. However, we do not augment the data by increasing or decreasing the image scale, as this would lose the spatial information that has been used for training the manipulation primitives. Additional information is given in Section V-C.

V. Experiments

Our experiments focus on two main points: 1) evaluating the performance of QDP compared to state-of-the-art baselines, and 2) evaluating different manipulation primitives on the task of cloth unfolding using a single robotic arm. Thus, we analyse the following:

- What is the impact of learning to sequentially optimise parameters such as the height or velocity of pre-defined manipulation primitives?
- How does the proposed network architecture for QDP perform compared to other architectures?
- What is the performance of QDP when transferred to the real-world in a zero-shot manner?

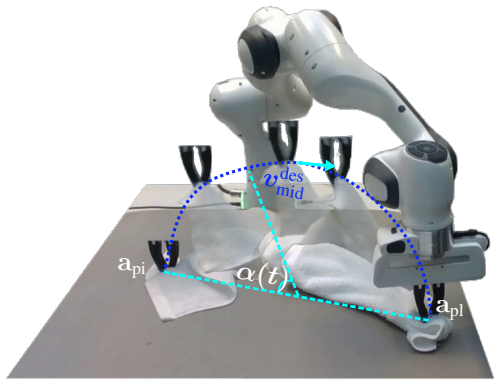


Fig. 3: Dynamic manipulation primitive that moves from the proposed $\mathbf{a}_{\text{pi}}$ to $\mathbf{a}_{\text{pl}}$ locations following a semi-circle shaped trajectory obtained from the quintic polynomial of $\alpha(t)$.

A. Manipulation primitives

We evaluate three manipulation primitives: pick-and-place, drag, and dynamic quintic polynomial; each defined at least by their pick and place locations. The primitives are defined as:

- **Dynamic Quintic Polynomial**: dynamic manipulation primitive which is defined by its velocity. This primitive follows a semi-circle shaped trajectory from $\mathbf{a}_{\text{pi}}$ to $\mathbf{a}_{\text{pl}}$, where $v^{des}_{\text{mid}} \in [0.08, 0.5]$ m/s is the desired velocity of the manipulator's end-effector at the middle point of the trajectory (see Fig. 3). The primitive is designed following a fifth-order polynomial of α angle [23], expressed as

$$\alpha(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5. \tag{10}$$

- **Pick-and-Place (P-n-P)**: quasi-static manipulation primitive with parameters $\{h_\theta, t_\theta\}$, where the height has the range of values $h_\theta \in [0.1, 0.2, 0.3, 0.4, 0.5]$ m. to lift the cloth; and the time to move from $\mathbf{a}_{\text{pi}}$ to $\mathbf{a}_{\text{pl}}$ is within the range of $t_\theta \in [10, 11, 12, 13, 14, 15]$ seconds.
- **Drag**: quasi-static primitive with t_θ as additional parameter. This primitive can be seen as a P-n-P primitive where h_θ is equal to zero, and the time has the range of values as the P-n-P primitive.

B. Baselines

In order to evaluate the performance of our method we compare against Max Value Map (MVP), proposed by [4], which suggests the pick position based on a spatial action map [14]. The network takes a combination of 5 different scales and 12 rotations of the input image, selecting the combination with highest Q-value to define the place position. It is important to note that the MVP method iterates through the spatial action maps until it finds a valid pick position. We use the same training procedure, where the input image is RGB $\mathbf{s} \in \mathbb{R}^{3\times64\times64}$, as in the original code provided by the authors. During training and evaluation of MVP, the

primitives height, time, and velocity, are set by using the median of the values $\mathbf{a}_\theta$ proposed by QDP after training.

In addition, we evaluate the performance of the sequential decision of pick-and-place parameters using different network architectures. We compare the U-net architecture against two manipulation architectures. The first one is the network used in the MVP method. The second network architecture is Form2Fit (F2F) [24], which showed great performance for rigid-body pick-and-place tasks. It is important to note that these networks do not share any parameters to decide the action parameters. Additional details about the implementation can be found on our website[1].

C. Experimental Set-Up

We employ Softgym [25] as the simulation environment with the modifications done by [4]. The simulation top-view images are originally in the dimension $\mathbf{s} \in \mathbb{R}^{3\times400\times400}$ and then reshaped onto the specific network input dimension. We train and evaluate on the data sets provided by [4] to benchmark against an existing common baseline.

The training for all the baselines is done using a V100 GPU, which takes approximately 3 days. All the methods are trained using three different random seeds until a total of $40{\cdot}10^3$ environment steps are collected into the replay buffer. We use the Adam optimiser [26] and scale the rewards by a factor of 10. By increasing the magnitude of the rewards their variation is more pronounced facilitating the convergence of the networks. The exploration coefficient starts at a value of $\varepsilon = 1.0$ and decays 0.001 per episode to finish at a value of $\varepsilon = 0.01$ after 20000 episode steps. During evaluation we rotate the top-view images in a range of $\{0, 180\}$ degrees, spaced by 15 degrees, and compare the pick Q-values of each input image. This enables to select the action with the highest expected improvement under the same cloth configuration.

Our experiments evaluate the maximum coverage and coverage improvement that can be achieved within 10 interactions with the cloth, both normalised by the estimated maximum area of the cloth. The coverage improvement allows to evaluate the performance of the methods when there is a high variance of the cloth initial configuration.

During evaluation, we use the depth image as prior data for correcting proposed pick points that are on the cloth low resolution image but are not in the high resolution image. Thus, pick points that are on a radius of 10 pixels of valid cloth points are used as pick location. The real-world experiments are performed using an Azure Kinect DK to obtain both the top-view and depth images. Here, the top-view images are $\mathbf{s} \in \mathbb{R}^{3\times1280\times720}$, and reshaped accordingly.

For the real-world experiments we evaluate three different cloths from the public household cloth benchmark [1]. We use a Franka Emika Panda to manipulate the cloths. Due to the limitations of the robot workspace we evaluate the objects that could be completely unfolded, which are the small towel (0.3×0.5 cm.), cotton napkin (0.5×0.5 cm.), and chequered rag (0.5×0.7 cm.). The initial cloth configuration is generated following the steps detailed by [27], where the cloth is grasped from one of the corners and placed in a crumpled configuration on a foam mat. Each of these cloths has a different pattern, size, elasticity, and weight; thus representing a wide variation of cloth types to evaluate the performance of each manipulation primitive. The real-world manipulation primitives are performed using a Cartesian impedance controller, where the stiffness and impedance parameters have been tuned empirically for the Franka Emika Panda robot.

[1] https://sites.google.com/view/qdp-srl

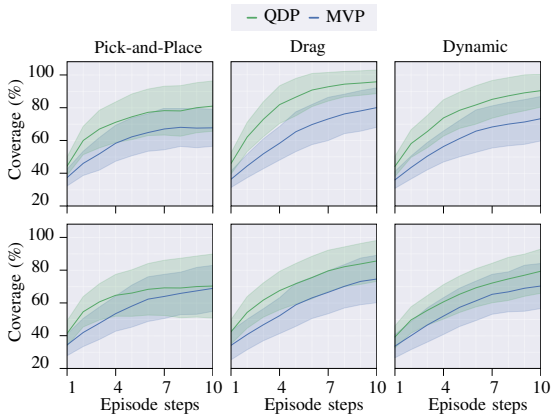


Fig. 4: Quantitative comparison of the coverage percentage for unfolding cloths in simulation. The results compare the proposed method QDP (*green*) against the baseline MVP (*blue*) for deciding the primitives parameter values. Three primitives are evaluated: Pick-and-Place, Drag, and Dynamic; where QDP selects the time t_θ and the velocity v_{mid} of the quasi-static and dynamic primitives. The results show an increase in coverage using QDP for both normal size (*Top*-row), and large size (*Bottom*-row) cloths.

VI. Simulation Experiments

A. Manipulation Primitives

We start by evaluating the performance of QDP against the MVP baseline for the aforementioned manipulation primitives. The coverage performance for both regular (*top*) and large (*bottom*) cloths test data sets is shown in Figure 4. Here, MVP uses, as fixed parameters, $t_\theta = 10$ for both P-n-P and drag, $h_\theta = 0.2$ for P-n-P, and $v_{\text{mid}} = 0.1$ for the dynamic primitive. For all the primitives, QDP presents higher mean coverage, where the variance for both drag and dynamic primitives falls within the 100% coverage. Although the improvement on the large data set is lower, the increase in performance of QDP compared to the MVP baseline is consistent. We attribute the reduced performance on the large data set to a low generalisation on the proposed method, as no large cloths are seen during training and the scale of the image is fixed as opposed to the baseline.

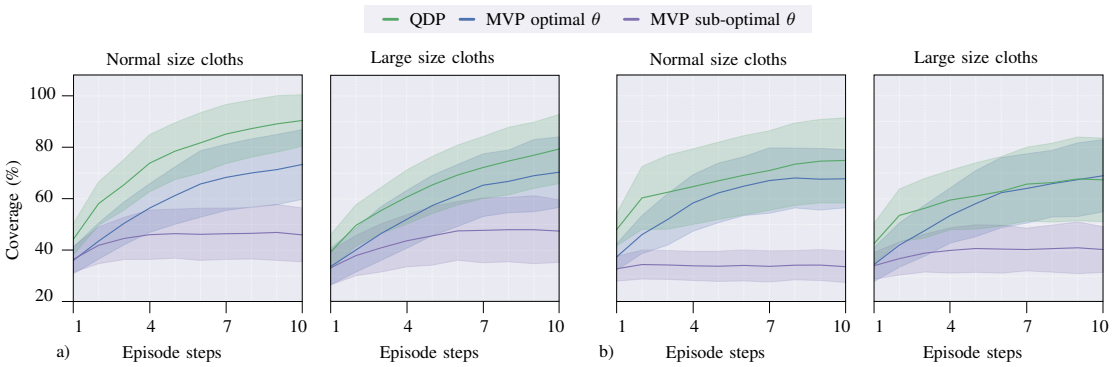


Fig. 5: Quantitative comparison of coverage percentage for unfolding cloth in simulation using the a) Pick-and-Place, and b) Dynamic manipulation primitive. The results compare the proposed QDP (*green*) against MVP using different velocity and height values. The additional parameter values are set as the median of the proposed values by QDP, denoted as MVP optimal θ (*blue*), and a sub-optimal value (*purple*); for normal rectangular cloths, and large rectangular cloths. This demonstrates that using sub-optimal primitive values is detrimental for the task of cloth unfolding.

B. Optimal and Sub-Optimal Primitive Parameter Values

We analyse as well the effect of using optimal and sub-optimal velocities and height values for the manipulation primitives. The results, in Figure 5, show the performance of QDP against MVP using as fixed parameter the median of the proposed values by QDP; as well as an experimentally selected sub-optimal value. In Figure 5 a), the dynamic manipulation primitive velocity has been set to an optimal value of $v_{\text{mid}} = 0.1$, and a sub-optimal value of $v_{\text{mid}} = 0.2$. The results show that using a sub-optimal value for the dynamic manipulation primitive is detrimental, as the performance drops more than 20% for both normal and large cloths. We attribute this poor performance to constantly using a high velocity, as opposed to adapting it, for cloth sizes and configurations that do not require such acceleration to improve their coverage.

The results in Figure 5 b) compare the P-n-P primitive using an optimal height value of $h_\theta = 0.2$, and a sub-optimal one of $h_\theta = 0.5$, keeping the primitive time fixed to $t_\theta = 10$. The constant large height results in a considerable performance drop for both normal and large size cloths. This is a consequence of lifting the cloth from a single point and losing contact with the surface, which results in a crumpled configuration when dropped onto its place location.

By using QDP to adapt the primitives the coverage performance improves at least 10% for most of the cases. We note that the P-n-P performance for large size cloths slightly drops compared to the optimal height, which can be a result of no large cloths in the training data set as mentioned before.

C. Network Architecture

Next, we continue by analysing whether the proposed network structure is the right choice for the sequential decision process. We evaluate the P-n-P primitive using different network architectures, shown in Figure 6. The U-Net architecture clearly outperforms the other two, showing

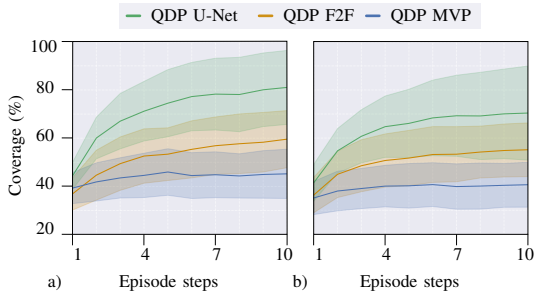


Fig. 6: Quantitative comparison of coverage percentage for unfolding cloth in simulation using the Pick-and-Place manipulation primitive. The results compare three different architectures, the proposed method QDP U-Net (*green*), Form2Fit or F2F (*orange*), and MVP (*blue*); for a) normal rectangular cloths, and b) large rectangular cloths.

an improvement of over 20% for the F2F network, and up to 40% for the MVP network. This demonstrates that the proposed U-Net architecture, sharing information over the sequential decision process, is one of the key ingredients for the performance of QDP.

VII. Real-World Experiments

Finally, we evaluate in the real-world three manipulation primitives, transferring the networks in a zero-shot manner. The performance is reported over 3 test episodes, with 10 episode steps each, resulting in at least 30 interactions per cloth, where we discard action steps in which the grasp was unsuccessful and the cloth configuration was not altered by the grasp attempt. Table I shows the results of the normalised coverage improvement after 10 interactions with the cloth. The results on the dynamic manipulation primitive show that QDP can increase the coverage improvement up to

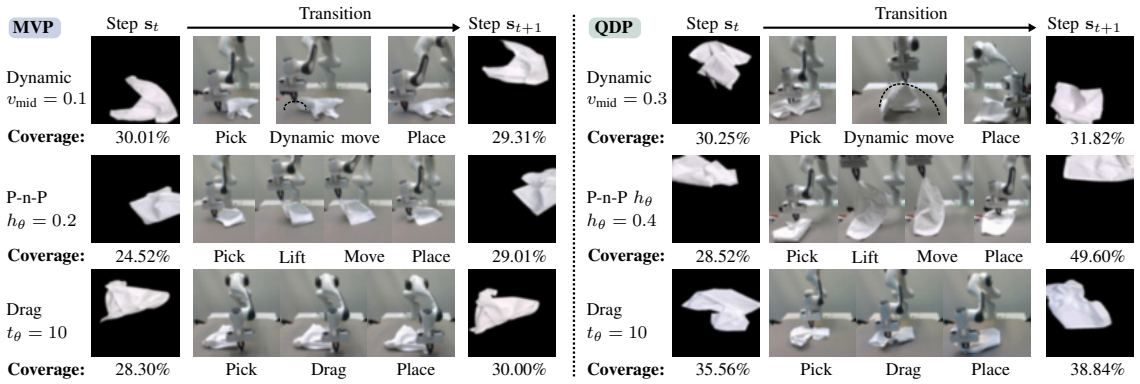


Fig. 7: Qualitative results for the real-world experiments for unfolding the napkin from the house-hold cloth data set. The results show the coverage at each time step $\mathbf{s}_t$ and $\mathbf{s}_{t+1}$, as well as the trajectories using the dynamic, pick-and-place (P-n-P), and drag manipulation primitives for both the baseline MVP (*left*) and the proposed method QDP (*right*). The results show that by modifying the velocity and height parameters QDP can alter the cloth configuration and lead to a better coverage.

TABLE I: Quantitative results of coverage improvement percentage after 10 interactions with the cloth in the real-world. The results show the performance of QDP and MVP for the the pick-and-place (P-n-P), drag and dynamic manipulation primitives. P-n-P t_θ and h_θ refer to QDP proposing either the time or height of the P-n-P primitive, respectively.

		Cloth		
		Towel	Napkin	Chequered Rag
MVP	P-n-P	24.93 ± 8.64	17.69 ± 11.57	-0.30 ± 4.70
	Drag	-1.21 ± 6.34	3.72 ± 4.02	-6.41 ± 2.96
	Dynamic	9.89 ± 1.16	1.57 ± 1.01	-6.04 ± 8.45
QDP (Ours)	P-n-P t_θ	0.56 ± 4.46	-2.75 ± 6.47	0.97 ± 7.71
	P-n-P h_θ	**29.39** ± 16.51	**20.43** ± 11.13	-4.97 ± 3.69
	Drag	-4.58 ± 4.89	3.69 ± 8.56	-1.35 ± 4.22
	Dynamic	9.88 ± 10.50	11.36 ± 4.23	**11.99** ± 11.96

11.99% compared to MVP. In addition, the P-n-P primitive where the optimal height is determined by QDP outperforms all the other manipulation primitives, for both the towel and napkin cloths, increasing the mean coverage more than 4.46% compared to MVP P-n-P, which is the second best primitive. These results show that modifying the velocity and height of the manipulation primitives based on the cloth state is beneficial, following the results achieved in simulation. Additionally, the results for each primitive in the chequered rag using QDP show superior performance compared to the performance of MVP. We hypothesise that the expected transitions of the rag when performing the manipulation primitives is closer to the training data, and thus is more in line with the simulation results.

Furthermore, Figure 7 provides qualitative results and shows the transitions for each manipulation primitive, both for QDP and MVP. During the transition from $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$, the QDP dynamic manipulation primitive is executed with a value of $v_{\text{mid}} = 0.3$, and QDP P-n-P uses a value of $h_\theta = 0.4$. These values that are away from the median of $v_{\text{mid}} = 0.1$ and $h_\theta = 0.2$ enable the displacement of a larger number of cloth points, not only increasing the coverage, but also moving the cloth to a configuration more suitable for a completely flattened state. In contrast, even though the baseline shows relative improvements in the coverage, it might get stuck in a sub-optimal cloth configuration where the cloth is too crumpled and it cannot be solved by using the fixed primitive parameter values.

Overall there is no improvement in determining the optimal time for the drag and P-n-P primitives for all the cloths. The decreased performance of QDP for these primitives compared to the simulation results shows the sim-to-real gap for finding the optimal pick and place locations. This could be solved by retraining the network using real data instead of transferring the networks directly from simulation. Another limitation when transferring to the real-world has been the amount of failed grasps, which could be improved by using a more accurate simulator. We will investigate if the performance can be improved by adding as well a depth image as information in the sequential process.

VIII. Conclusion

We presented QDP, a novel approach for sequentially choosing parameter values of manipulation primitives for cloth manipulation. While prior work has overlooked the effect of parameters such as the velocity or height of manipulation primitives, the proposed sequential decision process allows a greater variety and complexity of primitives to be used. This variety makes it possible to handle diverse fabric materials and sizes. Our experimental results show that, compared to previous work, QDP can improve up to a 20% coverage in simulation for the task of cloth unfolding. Furthermore, real-world experiments demonstrate the effectiveness of finding the optimal velocity and height for dynamic and quasi-static manipulation primitives.

Manuscript 1559 submitted to 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 1, 2023.

While we demonstrated the benefit of choosing the primitive parameter values for manipulating multiple cloths, generalising to a broader set of materials and shapes remains an open problem. Recent research by Hietala et al. [16] indicates that closed-loop feedback significantly improves the adaptation capabilities in cloth manipulation. Thus, combining parameterised primitives with real-time closed-loop feedback appears as a promising avenue towards more general and adaptive skills.

This work paves the way to a broader range of complex manipulation primitives, eliminating the human effort of fine-tuning or designing primitives, while reducing computational requirements due to the sequential decision process. This gives promise to exploring complex parameterised manipulation skills for shaping other deformable materials such as visco-elastic or elasto-plastic ones, which are present in many industrial and house-hold environments.

ACKNOWLEDGEMENTS

The authors would like to thank Kevin Sebastian Luck of Aalto University for their help and support in this work. The authors would also like to acknowledge the computational resources provided by the Aalto Science-IT project.

REFERENCES

[1] I. Garcia-Camacho, J. Borràs, B. Calli, A. Norton, and G. Alenyà, "Household cloth object set: Fostering benchmarking in deformable object manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 5866–5873, 2022.

[2] R. P. Joshi, N. Koganti, and T. Shibata, "Robotic cloth manipulation for clothing assistance task using dynamic movement primitives." New York, NY, USA: Association for Computing Machinery, 2017.

[3] B. Jia, Z. Pan, Z. Hu, J. Pan, and D. Manocha, "Cloth manipulation using random-forest-based imitation learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2086–2093, 2019.

[4] H. Ha and S. Song, "Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding," in *Conference on Robotic Learning (CoRL)*, 2021.

[5] Y. Avigal, L. Berscheid, T. Asfour, T. Kröger, and K. Goldberg, "Speedfolding: Learning efficient bimanual folding of garments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 1–8.

[6] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel, "Learning to Manipulate Deformable Objects without Demonstrations," in *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[7] M. T. Mason, *Mechanics of Robotic Manipulation*. Cambridge, MA, USA: MIT Press, 2001.

[8] R. Lee, D. Ward, V. Dasagi, A. Cosgun, J. Leitner, and P. Corke, "Learning arbitrary-goal fabric folding with one hour of real robot experience," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 2317–2327.

[9] X. Lin, Y. Wang, Z. Huang, and D. Held, "Learning visible connectivity dynamics for cloth smoothing," in *Conference on Robot Learning*, 2021.

[10] Z. Huang, X. Lin, and D. Held, "Mesh-based Dynamics with Occlusion Reasoning for Cloth Manipulation," in *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.

[11] A. Canberk, C. Chi, H. Ha, B. Burchfiel, E. Cousineau, S. Feng, and S. Song, "Cloth funnels: Canonicalized-alignment for multi-purpose garment manipulation," *arXiv preprint arXiv:2210.09347*, 2022.

[12] D. Wang, C. Kohler, and R. Platt, "Policy learning in se(3) action spaces," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 1481–1497.

[13] Y. Yamakawa, A. Namiki, and M. Ishikawa, "Dynamic manipulation of a cloth by high-speed robot system using high-speed visual feedback," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 8076–8081, 2011, 18th IFAC World Congress.

[14] J. Wu, X. Sun, A. Zeng, S. Song, J. Lee, S. Rusinkiewicz, and T. Funkhouser, "Spatial Action Maps for Mobile Manipulation," in *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[15] T. Weng, S. M. Bajracharya, Y. Wang, K. Agrawal, and D. Held, "Fabricflownet: Bimanual cloth manipulation with a flow-based policy," in *Conference on Robot Learning, 8-11 November 2021, London, UK*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 2021, pp. 192–202.

[16] J. Hietala, D. Blanco–Mulero, G. Alcan, and V. Kyrki, "Learning visual feedback control for dynamic cloth folding," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 1455–1462.

[17] L. Metz, J. Ibarz, N. Jaitly, and J. Davidson, "Discrete sequential prediction of continuous actions for deep rl," *arXiv preprint arXiv:1705.05035*, 2017.

[18] X. Zhu, D. Wang, O. Biza, G. Su, R. Walters, and R. Platt, "Sample Efficient Grasp Learning Using Equivariant Models," in *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.

[19] D. Wang, R. Walters, X. Zhu, and R. Platt, "Equivariant q learning in spatial action spaces," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 08–11 Nov 2022, pp. 1713–1723.

[20] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Springer International Publishing, 2015, pp. 234–241.

[21] P. J. Huber, *Robust Estimation of a Location Parameter*. New York, NY: Springer New York, 1992, pp. 492–518.

[22] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30.

[23] M. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Wiley, 2020.

[24] K. Zakka, A. Zeng, J. Lee, and S. Song, "Form2fit: Learning shape priors for generalizable assembly from disassembly," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 9404–9410.

[25] X. Lin, Y. Wang, J. Olkin, and D. Held, "Softgym: Benchmarking deep reinforcement learning for deformable object manipulation," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 432–448.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[27] I. Garcia-Camacho, M. Lippi, M. C. Welle, H. Yin, R. Antonova, A. Varava, J. Borras, C. Torras, A. Marino, G. Alenyà, and D. Kragic, "Benchmarking bimanual cloth manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1111–1118, 2020.

Publication IV

Julius Hietala, **David Blanco-Mulero**, Gokhan Alcan and Ville Kyrki. Learning Visual Feedback Control for Dynamic Cloth Folding. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Kyoto, Japan. pp. 1455-1462, October 2022.

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 23-27, 2022, Kyoto, Japan

Learning Visual Feedback Control for Dynamic Cloth Folding

Julius Hietala, David Blanco-Mulero, Gokhan Alcan, Ville Kyrki

Abstract— **Robotic manipulation of cloth is a challenging task due to the high dimensionality of the configuration space and the complexity of dynamics affected by various material properties. The effect of complex dynamics is even more pronounced in dynamic folding, for example, when a square piece of fabric is folded in two by a single manipulator. To account for the complexity and uncertainties, feedback of the cloth state using e.g. vision is typically needed. However, construction of visual feedback policies for dynamic cloth folding is an open problem. In this paper, we present a solution that learns policies in simulation using Reinforcement Learning (RL) and transfers the learned policies directly to the real world. In addition, to learn a single policy that manipulates multiple materials, we randomize the material properties in simulation. We evaluate the contributions of visual feedback and material randomization in real-world experiments. The experimental results demonstrate that the proposed solution can fold successfully different fabric types using dynamic manipulation in the real world. Code, data, and videos are available at https://sites.google.com/view/dynamic-cloth-folding.**

I. Introduction

Robotic manipulation of deformable objects has been applied to a different range of tasks from unfolding, folding, and shaping cloths [1]–[4] to rearranging objects such as cables, fabrics or bags into a goal configuration [5]. A common denominator of these solutions is that they involve static or quasi-static manipulation [6]. Instead, recent works have explored the effectiveness of dynamic manipulation when applied to deformable objects [7], [8].

Dynamic manipulation can be described as manipulation tasks where inertial forces are a crucial part of the process [6]. By contrast, quasi-static manipulation tasks neglect these forces. There are several benefits of dynamic manipulation over quasi-static manipulation when applied to cloth manipulation. First, dynamic manipulation enables reaching parts of the configuration space that cannot be reached with quasi-static manipulation. This can be useful for manipulating parts of the cloth that are not grasped by the manipulator by, for example, performing a dynamic primitive such as flinging the cloth [7]. Secondly, dynamic manipulation has been shown to be more efficient in tasks like unfolding cloth, where it requires fewer interactions with the environment than by performing quasi-static manipulation, such as pick and place primitives [7].

This work was financially supported by Academy of Finland grant numbers 328399 and 317020. (Corresponding author: David Blanco-Mulero)

The authors are with the Intelligent Robotics Group, Department of Electrical Engineering and Automation (EEA), Aalto University, 02150, Espoo, Finland. (e-mail: julius.hietala@alumni.aalto.fi; david.blancomulero@aalto.fi; gokhan.alcan@aalto.fi; ville.kyrki@aalto.fi)

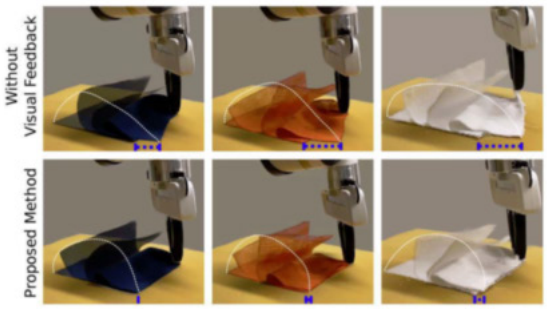


Fig. 1: The proposed solution succeeds in cloth folding via dynamic manipulation for different types of fabrics when compared to a fixed trajectory from a policy trained in simulation without visual feedback. The dashed blue line indicates the distance from the end position of the cloth non-controlled point to the target point.

In general, dynamic manipulation of cloth in the real world presents the following challenges:

- Observing the position and velocity of the cloth is not straightforward. The methods used in cloth manipulation either observe the cloth visually [4], [9] or use a motion capture system with markers attached to the cloth [10]. In dynamic manipulation, observing the velocity is especially important since the task's success depends on forces caused by acceleration.
- The behavior of the cloth depends on the material's physical properties. A fixed trajectory that successfully folds a certain cloth does not necessarily achieve the same end result on a different type of fabric (see Fig. 1).

Dynamic cloth folding has been studied only in simulation [8], where the true position, velocities, and forces of the cloth can be utilized. Thus, building visual feedback policies that can manipulate multiple materials in the real world remains an open problem.

In this work, we propose to solve the aforementioned problem by learning end-to-end a visual feedback policy using Reinforcement Learning (RL). The policy is learned in simulation using Soft-Actor Critic (SAC) [11] and then transferred to the real world. In order to adapt to different fabric types, our work makes use of domain randomization (DR) [12]–[14], where we randomize the physical properties of the cloth in simulation, to learn material-agnostic policies. In addition, we embed a Cartesian controller into the policy learning, where an identical controller is used in simulation and the real world. This helps the policy to account for the

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9981376

torque commands generated by the controller. We evaluate the trained policies both in simulation and in real-world experiments. Our results demonstrate that the visual feedback policies trained with dynamics DR succeed in dynamic cloth folding on a variety of fabrics. By contrast, visual feedback policies trained without dynamics DR as well as policies executed in an open-loop manner perform poorly across different fabrics. To the best of our knowledge, this is the first work to learning cloth folding using dynamic manipulation in the real world.

The specific contributions of this paper are then:

1) A method using domain randomization and visual feedback to learn policies for dynamic manipulation of different materials,
2) Experimental results showing that visual feedback with domain randomization is crucial for dynamic manipulation,
3) Demonstration of transfer of manipulation policies learned in simulation to the real world without any exploration in the real world.

II. Related Work

A. Quasi-static cloth manipulation

Quasi-static manipulation has been widely studied in cloth manipulation. Usually quasi-static cloth manipulation consists of finding the suitable pick-and-place position for tasks such as unfolding [4], [5] or folding [15]–[17]. There is quite an extensive number of methods that have been used to predict the pick and the place position ranging from value-based RL [15], model-free RL [16] and optical flow [17], to learning surrogate models for planning [4], [18]. These approaches only consider the start and end state of the cloth, thereby manipulating in an open-loop fashion, where they omit the cloth state while being manipulated.

An alternative is to use visual feedback policies, where the state of the cloth can be estimated at every time step of the manipulation and the policy can react accordingly. Matas *et al.* [9] proposed to transfer from simulation to the real world (sim-to-real) a visual feedback policy to perform quasi-static cloth manipulation tasks. In our work, we use a similar policy and training architecture. However, we evaluate the learned policies across different fabric materials in a dynamic cloth folding task, where the physical properties of the cloth play an essential role on the manipulation success. In addition, we use SAC instead of Deep Deterministic Policy Gradients (DDPG) [19].

B. Dynamic cloth manipulation

Prior works on dynamic cloth manipulation have used dynamic primitives such as a fling motion [7], parameterized trajectories [20], [21] or visual feedback [22]. Ha *et al.* [7] proposed a fling dynamic motion based on value-based RL to solve cloth unfolding. However, their approach only takes into account the start and end state of the cloth, thereby not being able to react to changes of the cloth configuration while manipulated. Furthermore, [7] trains policies in simulation and then collects additional real-world experience to continue the training. In our work, we transfer the policies from sim-to-real without any further training using real-world data. Previous works that have solved dynamic cloth manipulation as an optimization problem [23], or using parameterized trajectories [20], execute their trajectories in an open-loop manner. Similarly, these solutions cannot adapt their trajectories while the motion is executed or generalize across different cloths. The use of visual feedback for dynamic cloth manipulation has been previously studied [22]. However, it has been restricted to custom hardware with no learning involved. In this work, we propose to learn visual feedback policies that adapt to different fabric types using RL.

Recently, Jangir *et al.* [8] studied three dynamic cloth manipulation tasks in simulation, where they proposed to learn a policy using RL. Their work was limited to simulation, where the policy used the internal states of the cloth given by a simulator to predict the next action. This is infeasible in the real world as the true state of the cloth cannot be measured. In this work we study one of the dynamic manipulation tasks proposed by [8], that is, sideways folding. In contrast, we tackle the challenge of observing the cloth in the real world by learning a visual feedback policy.

III. Background

A. Problem statement

We consider a scenario, where a robotic manipulator interacts in finite episodes over discrete time steps with a cloth that needs to be manipulated to a goal configuration. We then address the problem of how to learn the control policy for the system. We assume that the full state of the cloth cannot be observed directly in the real world. Instead, we can only observe its state from visual feedback. Therefore, we formulate the dynamic cloth manipulation problem as a Partially Observable Markov Decision Process (POMDP). The goal of the RL agent is to find the optimal policy π^*, parametrized by θ^*, that maximizes the expected value of the cumulative reward

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}\left[\sum_{n=1}^{N} \gamma\, R(\mathbf{s}_t, \mathbf{a}_t, \mathbf{g})\right], \tag{1}$$

where $\mathbf{s}_t \in S$ set of states, $\mathbf{a}_t \in A$ set of actions, $\mathbf{o}_t \in O$ set of observations, $\mathbf{g} \in G$ set of goals and $\gamma \in (0, 1]$ is the discount factor. We include a goal $\mathbf{g}$ at each episode which is used to determine whether the configuration of the cloth results in success in the task or not.

B. Dynamic sideways cloth folding task

This work focuses on dynamic cloth folding. The task we consider is dynamic sideways cloth folding, where a single manipulator interacts with the cloth. This task requires manipulating flat cloths into a folded configuration, where only a single point of the cloth can be controlled. Thus, the policy needs to exert the necessary forces on the controllable point so that also the non-controllable points get to a folded state. The starting and folded states are shown on the left and right sides of Fig. 3 respectively. We consider $C = 8$

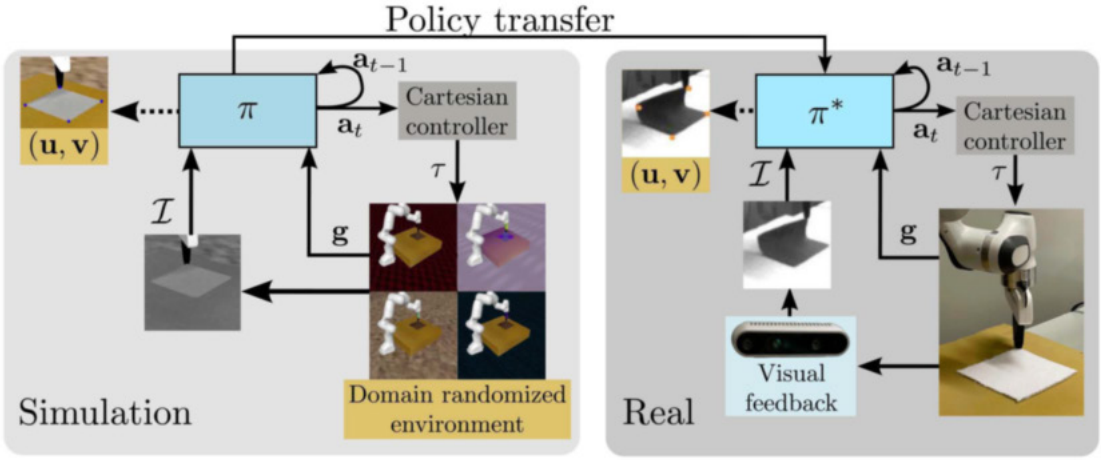


Fig. 2: Overview of the proposed solution. The policy is trained in simulation (**left**), where the simulated environment randomizes the dynamics of the cloth as well as the environment's visual properties. The trained policies π^* are transferred directly to the real world (**right**). The policy receives an image $\mathcal{I}$, a goal $\mathbf{g}$ and the previous action $\mathbf{a}_{t-1}$ as input. The policy outputs the next action $\mathbf{a}_t$ as well as a prediction of the cloth corners $(\mathbf{u}, \mathbf{v})$. Both simulation and real setup use the same Cartesian controller, which generates the torque commands τ to be executed by the manipulator.

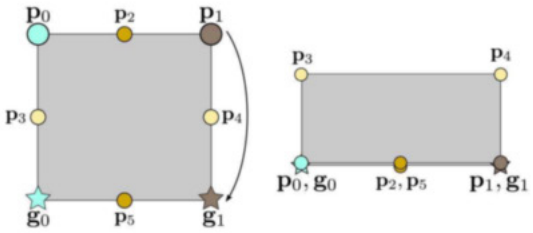


Fig. 3: Start (**left**) and desired (**right**) end configurations of the cloth for the dynamic sideways folding task, where $\mathbf{p}_1$ is the only controllable point, $\mathbf{p}_0$ is one of the non-controllable corners, and $\mathbf{g}_1, \mathbf{g}_0$ their respective goal locations.

cloth points $(\mathbf{p}_i, \mathbf{g}_i)$, where $\mathbf{p}_1$ is the only grasped corner. For a fold to be considered successful, the distance

$$d_{\text{sum}} = \sum_{i=0}^{1} \|\mathbf{p}_i - \mathbf{g}_i\|_2, \tag{2}$$

should be below a threshold δ at the end of an episode.

The reward is a combination of a sparse and a dense reward [24]. Thus, the agent will get a sparse reward $r_t = -1$ when the fold is unsuccessful, and a dense reward when it is successful according to $r_t = d_{\text{sum}}$, where the reward scales linearly from 0 to 1 based on the distances.

In order to focus on learning a visual feedback dynamic manipulation policy we assume that the cloth starts from a grasped state, which can be found using some of the methods discussed earlier [7], [15].

IV. Proposed Method

A general view of our proposed method for dynamic cloth folding is depicted in Fig. 2. The policies are trained in simulation using RL, where we use DR including the physical properties of the cloth, in order to achieve generalization across different fabric types. In both the sim (left Fig. 2) and real system (right Fig. 2) the policies take a visual input $\mathcal{I}$ and output the next actions to the same Cartesian controller, Operational Space Control (OSC) [25], which facilitates transferring the policy trained in simulation.

A. Learning in Simulation

In order to efficiently gather training data for policies that run in the real world, we use a physics engine simulator. The simulation environment includes a flat cloth and a robotic manipulator as shown in Fig. 2. The cloth is simulated as a 2D grid of bodies connected via a spring-damper system [26], where one of the corners is connected to the robot end-effector to assume the initial grasped state of the cloth.

The simulation provides the internal states S of the cloth, along with a 100×100 grayscale image $\mathcal{I} \in O$ of the cloth. The images serve as the partial observations of the RL problem and are used by the policy π to determine the next action $\mathbf{a}_t$. In addition, the policy takes as input the previous action $\mathbf{a}_{t-1}$ to provide temporal information, and a goal $\mathbf{g}$ that represents the desired cloth configuration.

We follow the same architecture and training scheme as [9], where the policy is composed of three mappings

$$\pi : \begin{cases} \mathbf{h}_1 : \mathcal{I} \rightarrow \mathcal{L}, \\ \mathbf{h}_2 : (\mathcal{L}, \mathbf{a}_{t-1}, \mathbf{g}) \rightarrow \mathbf{a}_t, \\ \mathbf{h}_3 : \mathcal{L} \rightarrow (\mathbf{u}, \mathbf{v}). \end{cases} \tag{3}$$

First, the input image is mapped to a latent space $\mathcal{L}$ via $\mathbf{h}_1$. The second mapping $\mathbf{h}_2$ outputs the continuous action $\mathbf{a}_t$, which is the desired change in the end-effector position. The last mapping $\mathbf{h}_3$ provides an auxiliary output with the estimated position of cloth points $(\mathbf{u}, \mathbf{v})$ in the image plane. The auxiliary outputs are only used during training for guiding the policy to detect relevant features [9]. The first mapping is implemented as a convolutional neural network, whereas the other two mappings are fully connected neural networks. The inference speed of the policy is crucial for the manipulator to execute the fast dynamic actions. We describe the network architecture, implementation, and the inference speed in Section V-B.

We use SAC for learning the visual feedback policy. Additionally, our method uses Hindsight Experience Replay [27] to speed up training. We also incorporate expert demonstrations in the replay buffer. The specific details about the policy training are given in Section V-B. In our system, the Q-functions of SAC observe the position and velocity of C cloth points $\{\mathbf{p}_i, \dot{\mathbf{p}}_i\}_{i=1}^{C} \in S$, along with the goal and current action. The reason behind the observation asymmetry between policy and Q-functions is that the cloth position and velocity information are not available in the real world, and the Q-functions are only needed during training. The fact that the policy observes only images rather than cloth points facilitates transferring the policy to the real world.

B. Sim-to-real dynamic manipulation

In order to effectively transfer a visual feedback policy from simulation we incorporate two methods into the policy learning process: 1) embedding the Cartesian controller used in the real hardware into the policy learning process, and 2) domain randomization, including the physical and visual properties of the cloth as well as a temporal delay in the observations [28].

The first element of our method that facilitates the transfer is employing the controller of the real manipulator in the policy learning process in simulation. For performing dynamic cloth manipulation in the real world we use a cascaded structure combining a low-frequency policy with a Cartesian controller operating at a high frequency required by the real hardware. The visual feedback policy is executed at a lower frequency due to the real-world constraints at which the images can be observed (see Section V). The Cartesian controller maps the desired positions of the end-effector given by the policy into the desired joint torques of the robotic manipulator. We propose to use the OSC as Cartesian controller, whereas other types of Cartesian controllers could be used as well. By embedding the controller in the learning process we let the policy implicitly learn the dynamics of the controller. Therefore, there is no need to tune the gains of the controller to a specific task prior to training. We use the policy output to compute the target end-effector position $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{a}_t$, where $\mathbf{x}_t$ is the previous desired position of the end-effector. Then, to provide a high-frequency reference signal to the OSC controller in between policy steps, the policy is interpolated K times from $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$ according to

$$\mathbf{x}_{d,k+1} = \lambda \cdot \mathbf{x}_{t+1} + (1 - \lambda) \cdot \mathbf{x}_{d,k}, \tag{4}$$

where λ is a filter value controlling the speed of the updates, $\mathbf{x}_{d,k}$ is the current desired position at sub-step k, and $\mathbf{x}_{d,0}$ is the previous end-effector position. The reason to use such filtering is to avoid drastic movements that may cause the robot to become unstable when new actions are performed.

The second element to assist in the sim-to-real transfer is DR. In order to randomize the physical properties of the cloth we first sample the cloth parameters ξ_c from a uniform distribution $\xi_c \sim \mathcal{U}$ across the possible ranges for each parameter (see Section V). The cloth parameters are simulator-dependent, in our case the spring-damper system is characterized by its impedance, stiffness and damping [26]. Then, we collect expert dynamic motion demonstrations from the real world that succeed on sideways folding on various real-world cloths (see Section V-B). The sampled parameters ξ_c are evaluated in simulation under the demonstration trajectories. Out of the sampled parameters, we pick the top M parameter sets that achieve the highest cumulative reward. Thus, we assume that the parameters that lead to a folded state and the highest cumulative reward are more realistic and can be used for randomizing the physical properties of the cloth. This approach is opposed to uniformly sampling the physical properties for each episode, which might lead to unrealistic behavior of the cloth that is not useful in the sim-to-real transfer. We denote the set of valid cloth properties as $F_{sim} = \{f_1, ..., f_M\}$, where we sample from F_{sim} at each episode during training. These properties do not necessarily match the real-world cloth dynamics, as we measure the success of the collected trajectories on a cloth type over different randomized parameters.

In addition to the physical properties of the cloth, we randomize the visual properties of the simulation environment including lighting conditions, image noise, camera position, and object textures. Additionally, we randomize the texture of the cloth, which is sampled from a set of textures gathered from a set of real cloths F_{real}. These parameters are sampled from a uniform distribution for each episode (see Section V for details).

Finally, considering that in the real world the frequency at which we obtain image observations is not stable, we incorporate Gaussian noise in the simulation image sample frequency. Therefore, the policy is trained with a random temporal delay between the taken action and the next observed state. These delays present a high impact on the agent performance [29]. Extensions of our work can incorporate a correction of the delay to improve further the agent performance [30].

V. Experiments

The goal of the experiments is to evaluate the proposed solution for learning visual feedback policies for cloth folding via dynamic manipulation. To that end, our experiments investigate: 1) how well does the performance of the policies in simulation match the real world, and 2) the performance of visual feedback policies across different fabric materials.

TABLE I: Real-world fabrics set F_{real} used for collecting expert demonstrations and real-world experiments.

Cloth	Fabric type	Weight (g/m^2)
Blue	Cotton	400
White	Cotton	345
Orange	Polyester	135

A. Baselines

In simulation, we use as a baseline the method proposed by Jangir *et al.* [8], where the policy directly observes the cloth positions and velocities $(\mathbf{p}_i, \dot{\mathbf{p}}_i)$ and there is no DR in the training process. We use SAC instead of DDPG so as to compare the policy mapping instead of the policy training method. The policies trained with this baseline are not directly transferable to the real world since the internal states of the cloth are not available. Therefore, we train multiple policies, each using different parameters $\xi_c \sim F_{\text{sim}}$, and select the ones that achieve the highest cumulative reward. Then, we execute the fixed trajectories from simulation on the real hardware, which we denote as π_{fixed} on the real experiments. This serves also to evaluate the performance of fixed trajectories across cloths with different dynamics.

The second baseline (π_{visual}) follows the same training and controller architecture as the proposed solution (π_{ours}). However, the policies are trained without randomizing the cloth's physical properties. Similarly to the first baseline, we train multiple policies on different parameters and transfer the ones that perform best in simulation. The policy π_{visual} serves as an ablation study of randomizing the cloth dynamics for learning visual feedback policies.

B. Simulation setup and training

The simulated environment is implemented in MuJoCo [26] where a model of the Franka Emika Panda is included along with the 2D grid of bodies representing the cloth.

In order to determine the set of cloth physical properties F_{sim} we used three different test cloths $F_{\text{real}} = \{f_{\text{orange}}, f_{\text{white}}, f_{\text{blue}}\}$. The cloth properties can be qualitatively differentiated as f_{blue} the most rigid and f_{orange} as the most shallow and most susceptible to deformation when manipulated (see Table I). Alternatively, other simulators that support different materials can be used [31]. The randomization set F_{sim} is composed of $M = 20$ cloth parameters which are determined using the demonstration-based parameter identification scheme defined in Section IV-B.

The environment episode has a length of 250 time steps, where the OSC controller acts at the simulation frequency of 100 Hz and the policy at 10 Hz. This was designed in order to mimic the real system, where the policy is interpolated following (4). An episode is terminated either at the end of the time steps or when the cloth corner $\mathbf{p}_1$ is within the threshold $\delta = 4$cm from its goal for more than 100 time steps. At each training episode the cloth goal corner $\mathbf{g}_i$ is sampled within 2cm of the adjacent corners to $\mathbf{p}_0$ and $\mathbf{p}_1$. The goal is specified in Cartesian coordinates as the relative position to the grasped corner given the cloth size.

The policy training is performed for 100 epochs, where each epoch consists of 10^4 environment time steps. At the end of each cycle, the collected observations are added to a replay buffer and policy optimization is performed for 1000 gradient steps using SAC. The replay buffer is a rolling buffer with 10^5 buffer size. We collected a single demonstration and added Gaussian noise $\mathcal{N}(0, 0.005)$ to the trajectory. The trajectory is then used to generate a tuple of expert state-actions added to the replay buffer. About 10% of the trajectories that are added consist of real-world demonstrations that succeed in cloth folding. This was done to further aid exploration when training the policy. Each epoch is concluded with 20 evaluation episodes where the success rate and corner distances to the goals are measured.

The policy network architecture as well as the specific range of randomization parameters can be found in the open source code[1]. The policy output actions are scaled to the range of [-0.03, 0.03] for each dimension.

C. Real-world setup

The real-world experiments are performed using the Franka Emika Panda robot and Robot Operating System [32].

The visual feedback input is provided by a Realsense D435 camera. In order to obtain observations at a valid frequency, we use the maximum available frequency setting of 300 frames per second, where the expected latency is between 25-39 ms [33]. The system is designed so that the policy acts at 10 Hz and the latest camera observation is given to the policy. The chosen frequency ensures that each action is determined based on the cloth behavior that took place after the previous action, which is a requirement for the Markov property to hold [34]. This procedure is followed for both π_{visual} and π_{ours}, whereas the fixed trajectory policy π_{fixed} sends directly the trajectory via-points at a frequency of 10 Hz to the OSC controller. The policy is implemented using PyTorch [35], which allows for an inference time below 20 ms.

In order to validate the performance over different cloth types, the real-world experiments are run across the set of F_{real}, where we run 10 evaluations per each fabric and policy type. The baseline policies π_{fixed} and π_{visual} transferred to the real world are selected as those that perform best under the cloth parameters ξ_c that match better the demonstration trajectories for each of the real test cloths.

Prior to running an evaluation, the cloth to be folded is attached to the robot's end-effector and the robot is reset to the same joint configuration for each episode. The grasping, camera configuration and lighting conditions are kept fixed across all evaluation types to make the evaluation as standardized as possible. The evaluation includes empirically measuring the distance of $\mathbf{p}_0$ and $\mathbf{p}_1$ to their goal locations $\mathbf{g}_0$ and $\mathbf{g}_1$ at the end of the trajectory. The goals are provided to the policies as in the simulated setup.

[1]https://github.com/hietalajulius/dynamic-cloth-folding

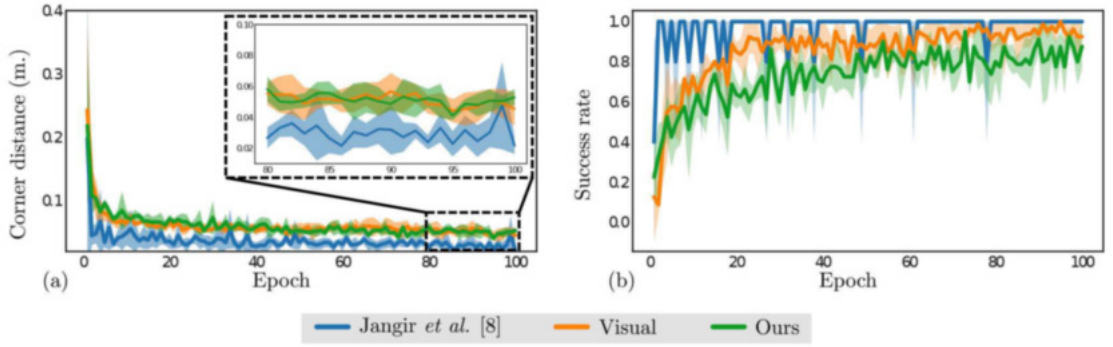


Fig. 4: Mean and variance for (a) fold distance error d_{sum} and (b) success rate for each evaluation epoch for each method.

VI. Results

We evaluate the folding performance of the different policies by measuring the sum of the distances (2) of the grasped and non-grasped corners to their goals once each trajectory has finished. It should be noted that d_{sum} was used in the reward function during training, so this measure serves also to assess the sim-to-real transfer quality. To determine whether any difference in the evaluation metrics between the methods is statistically significant we use the Mann-Whitney U test for the real-world experiments (Section VI-B). We choose a statistical significance level of 0.05 prior to the evaluations to assess whether the underlying distributions of the corner distance measurements across evaluations differ at least by the amount reported.

A. Simulation results

First, to set a baseline for the real-world folding task, we assess the policy evaluation during training over five different random seeds for each method via the metric d_{sum} and the success rate, Fig. 4a and Fig. 4b respectively.

The results show that the policies can converge and succeed on the task within 100 epochs. The policy based on the work by Jangir *et al.* [8] is able to learn the task only after a few epochs. We hypothesize that the success rate converges faster than in what is reported in [8] due to the fact that we omit the grasping problem. Additionally, the use of SAC is expected to aid in exploration compared to DDPG used in [8]. The corners error d_{sum} ranges from 2-4 cm on average at the end of the training, which is approximately 50% less than the visual feedback-based methods. This result suggests that observing the exact positions and velocities of selected points on the cloth can attain more accurate folds than the visual observation methods.

B. Real-world results

First, in order to compare against the simulation results, we evaluate the real-world error d_{sum} for each fabric and policy (see Fig. 5). The first thing to note is that the proposed solution π_{ours} significantly outperforms the other methods consistently for all fabrics. Furthermore, the achieved error d_{sum} even falls below the range seen during evaluation in simulation. The fixed trajectories from π_{fixed} present a significantly higher error compared to simulation for all the cloth types, which highlights the sim-to-real gap. This can also be seen from the variability across the different cloth types. The visual feedback-based method π_{visual} performs considerably better compared to trajectories from π_{fixed} for the white cloth, but is comparable or worse for the other types. This indicates that visual feedback is not enough for solving the task across a variety of fabrics, and the randomization of the cloth's physical properties plays an essential role while training the policy. The variance of the feedback-based methods shows that the results are less stable. This points to the fact that the policies are susceptible to noise from the environment such as lighting and camera position. Comparing the proposed method error to the baselines shows statistical significance for all cloth types, as shown in Table II.

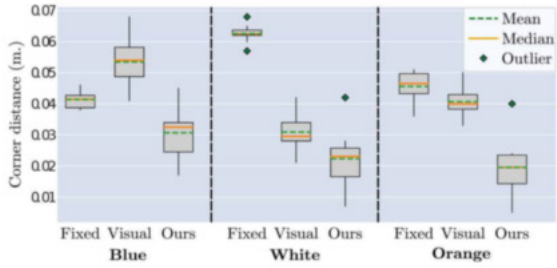
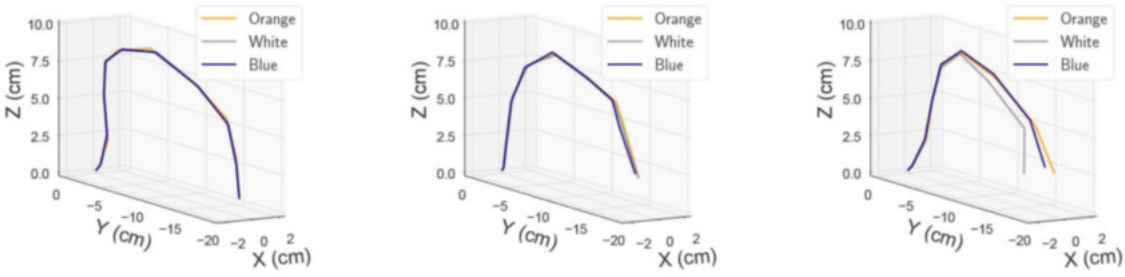


Fig. 5: Real-world measurements of distance error d_{sum} (in m) for the different methods.

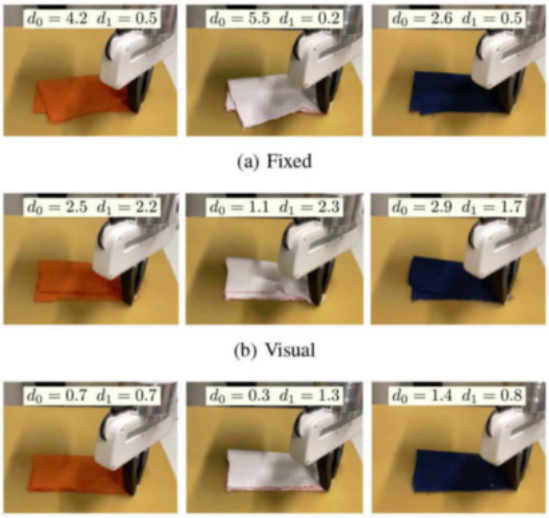
TABLE II: Statistical significance (p-values) of differences between Ours against the evaluated baseline methods using the Mann-Whitney U test. All results are statistically significant ($p < 0.05$).

	Fixed	Ours-
Ours (Orange)	$\mathbf{4 \cdot 10^{5}}$	$\mathbf{2 \cdot 10^{4}}$
Ours (White)	$\mathbf{1 \cdot 10^{5}}$	$\mathbf{9 \cdot 10^{3}}$
Ours (Blue)	$\mathbf{2 \cdot 10^{3}}$	$\mathbf{4 \cdot 10^{5}}$



(a) Fixed (b) Visual (c) Ours

Fig. 6: Qualitative comparison of mean of dynamic manipulation trajectories from real world experiments on three fabric types for the policy (a) Fixed, (b) Visual and (c) Ours.



(a) Fixed

(b) Visual

(c) Ours

Fig. 7: Qualitative and quantitative results showing the end configurations for each cloth type after a fold for each policy, where d_0 and d_1 represent the distance to the left and right corners respectively in cm.

We also evaluate how the policies adapt for each of the fabrics by qualitatively comparing the trajectories for each method and fabric in Fig. 6. The trajectories for π_{fixed} are nearly identical, which indicates that the best trajectories for each of the cloth sets F_{sim} were quite similar. The trajectories from π_{visual} present a minor deviation for different cloth types. In contrast, the trajectories from π_{ours} showcase more variation over the different fabric types. This indicates that by randomizing the cloth's physical properties the policy can adapt its trajectory to different fabric types.

Finally, we qualitatively and quantitatively compare the policies fold error for each corner, d_0 and d_1, over the test fabrics in Fig. 7. The results show that π_{ours} can consistently fold both the controllable and non-controllable cloth points across a variety of materials. The results from π_{visual} are consistent with the previous results, the fold error is higher when the physical properties of the cloth are not randomized. The trajectories from π_{fixed} achieve folds where the non-grasped corner reaches past its goal, which can be explained by the lack of feedback.

In summary, our results show the importance of visual feedback for dynamic cloth folding. Additionally, we provide evidence that visual feedback policies trained in simulation with domain randomization can react and adapt its trajectory when manipulating different fabric materials. While performing the real-world experiments we noticed that the policies trained using the proposed solution were quite sensitive to the lighting conditions and camera position, despite the domain randomization. Potential extensions of this work can study the effectiveness of the proposed solution to other tasks such as cloth unfolding.

VII. Conclusion

We presented a solution for cloth folding using dynamic manipulation by transferring visual feedback policies from simulation to the real world. The policies are trained in simulation using Reinforcement Learning and transferred directly to the real world. The transferred policies are capable of folding different fabric types by randomizing the physical properties of the cloth in the policy training. In addition, the solution embeds a Cartesian controller into the learning process, which facilitates the transfer process. We evaluated the folding performance in the real world and contrasted the results with the simulation, indicating the sim-to-real gap. Experimental results showed that visual feedback without domain randomization is not sufficient to manipulate multiple fabric types, while training under different cloth dynamics allows to succeed on the dynamic cloth folding task.

In the future, there will be a need for more general solutions that can be employed in a variety of tasks from bed-making to folding clothes. Thus, we need to study how to enable robotic manipulation across a large range of cloths in terms of sizes, shapes, and dynamics. We foresee that dynamic manipulation will play an important role towards solving these manipulation problems.

REFERENCES

[1] S. Miller, J. Van Den Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel, "A geometric approach to robotic laundry folding," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 249–267, 2012.

[2] D. Seita, N. Jamali, M. Laskey, A. K. Tanwani, R. Berenstein, P. Baskaran, S. Iba, J. Canny, and K. Goldberg, "Deep Transfer Learning of Pick Points on Fabric for Robot Bed-Making," in *International Symposium on Robotics Research (ISRR)*, 2019.

[3] A. Ganapathi, P. Sundaresan, B. Thananjeyan, A. Balakrishna, D. Seita, J. Grannen, M. Hwang, R. Hoque, J. E. Gonzalez, N. Jamali, K. Yamane, S. Iba, and K. Goldberg, "Learning dense visual correspondences in simulation to smooth and fold real fabrics," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 515–11 522.

[4] X. Lin, Y. Wang, Z. Huang, and D. Held, "Learning visible connectivity dynamics for cloth smoothing," in *Conference on Robot Learning*, 2021.

[5] D. Seita, P. Florence, J. Tompson, E. Coumans, V. Sindhwani, K. Goldberg, and A. Zeng, "Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[6] M. T. Mason, *Mechanics of Robotic Manipulation*. Cambridge, MA, USA: MIT Press, 2001.

[7] H. Ha and S. Song, "Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding," in *Conference on Robotic Learning (CoRL)*, 2021.

[8] R. Jangir, G. Alenyà, and C. Torras, "Dynamic cloth manipulation with deep reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4630–4636.

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

[10] B. Balaguer and S. Carpin, "Combining imitation and reinforcement learning to fold deformable planar objects," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 1405–1412.

[11] 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*. PMLR, 2018, pp. 1861–1870.

[12] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30.

[13] S. James, A. J. Davison, and E. Johns, "Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, 13–15 Nov 2017, pp. 334–343.

[14] V. Petrík and V. Kyrki, "Feedback-based fabric strip folding," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 773–778.

[15] R. Lee, D. Ward, V. Dasagi, A. Cosgun, J. Leitner, and P. Corke, "Learning arbitrary-goal fabric folding with one hour of real robot experience," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 2317–2327.

[16] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel, "Learning to Manipulate Deformable Objects without Demonstrations," in *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[17] T. Weng, S. M. Bajracharya, Y. Wang, K. Agrawal, and D. Held, "Fabricflownet: Bimanual cloth manipulation with a flow-based policy," in *Conference on Robot Learning, 8-11 November 2021, London, UK*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 2021, pp. 192–202.

[18] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, "Learning predictive representations for deformable objects using contrastive estimation," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 564–574.

[19] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016.

[20] H. Zhang, J. Ichnowski, D. Seita, J. Wang, H. Huang, and K. Goldberg, "Robots of the lost arc: Self-supervised learning to dynamically manipulate fixed-endpoint cables," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 4560–4567.

[21] V. Lim, H. Huang, L. Y. Chen, J. Wang, J. Ichnowski, D. Seita, M. Laskey, and K. Goldberg, "Planar robot casting with real2sim2real self-supervised learning," 2021.

[22] Y. Yamakawa, A. Namiki, and M. Ishikawa, "Dynamic manipulation of a cloth by high-speed robot system using high-speed visual feedback," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 8076–8081, 2011, 18th IFAC World Congress.

[23] S. Zimmermann, R. Poranne, and S. Coros, "Dynamic manipulation of deformable objects with implicit integration," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4209–4216, 2021.

[24] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, and W. Zaremba, "Multi-goal reinforcement learning: Challenging robotics environments and request for research," 2018.

[25] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[26] 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*, 2012, pp. 5026–5033.

[27] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[28] A. Rupam Mahmood, D. Korenkevych, B. J. Komer, and J. Bergstra, "Setting up a reinforcement learning task with a real-world robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 4635–4640.

[29] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, 2021.

[30] Y. Bouteiller, S. Ramstedt, G. Beltrame, C. J. Pal, and J. Binas, "Reinforcement learning with random delays," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

[31] H. Wang, R. Ramamoorthi, and J. F. O'Brien, "Data-driven elastic models for cloth: Modeling and measurement," *ACM Transactions on Graphics*, vol. 30, no. 4, pp. 71:1–11, July 2011, proceedings of ACM SIGGRAPH 2011, Vancouver, BC Canada.

[32] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Online]. Available: https://www.ros.org

[33] "High-speed capture mode of intel® realsense™ depth camera d435." [Online]. Available: https://dev.intelrealsense.com/docs/high-speed-capture-mode-of-intel-realsense-depth-camera-d435

[34] C. Shi, R. Wan, R. Song, W. Lu, and L. Leng, "Does the markov decision process fit the data: testing for the markov property in sequential decision making," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8807–8817.

[35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

Publication V

David Blanco-Mulero, Oriol Barbany, Gokhan Alcan, Adrià Colomé, Carme Torras, Ville Kyrki. Benchmarking the Sim-to-Real Gap in Cloth Manipulation. Accepted for publication in *IEEE Robotics and Automation Letters (RA-L)*, vol. 9, issue 3, pp. 2981-2988, March 2024.

Benchmarking the Sim-to-Real Gap in Cloth Manipulation

David Blanco-Mulero[1], Oriol Barbany[2], Gokhan Alcan[1], Adrià Colomé[2], Carme Torras[2], and Ville Kyrki[1]

***Abstract*—Realistic physics engines play a crucial role for learning to manipulate deformable objects such as garments in simulation. By doing so, researchers can circumvent challenges such as sensing the deformation of the object in the real-world. In spite of the extensive use of simulations for this task, few works have evaluated the reality gap between deformable object simulators and real-world data. We present a benchmark dataset to evaluate the sim-to-real gap in cloth manipulation. The dataset is collected by performing a dynamic as well as a quasi-static cloth manipulation task involving contact with a rigid table. We use the dataset to evaluate the reality gap, computational time, and simulation stability of four popular deformable object simulators: MuJoCo, Bullet, Flex, and SOFA. Additionally, we discuss the benefits and drawbacks of each simulator. The benchmark dataset is open-source. Supplementary material, videos, and code, can be found at `https://sites.google.com/view/cloth-sim2real-benchmark`.**

I. Introduction

Cloth manipulation is a crucial component in applications ranging from care-giving [1] and household chores [2], to the textile industry. Endowing robots with cloth manipulation skills is non-trivial. First, deformable objects have infinite Degrees of Freedom (DoFs), which makes it challenging to represent their state in the world [3]. Second, deformable objects have complex dynamics, which is even further pronounced when performing dynamic manipulation actions that require acceleration forces to succeed with the task [4, 5]. Third, deploying a robot in the real-world presents safety challenges such as damaging the physical system or the environment the robot interacts with.

Considerable research on cloth manipulation addresses these challenges with the aid of simulation engines [1, 6–8]. This relaxes the safety issues and provides a vast amount of trials where the controllers can be evaluated and improved. However, these simulators approximate the dynamics of the real world, which results in a gap when compared to reality [9]. This reality gap becomes even more apparent when performing dynamic cloth manipulation tasks [10, 11]. Under longer-term prediction the subsequent errors accumulate, widening up the reality gap, which results in a poor sim-to-real transfer. However, no studies are available that quantify the reality gap when performing dynamic cloth manipulation tasks.

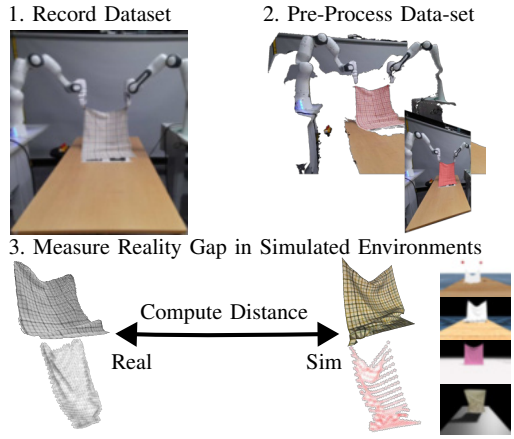


Fig. 1: The real-world cloth manipulation dataset was collected, pre-processed and benchmarked against multiple simulation engines, assessing their sim-to-real gap.

Although the state-of-the-art continues using the available simulators for learning cloth manipulation tasks, the fidelity of simulators for these tasks has not been thoroughly evaluated. While domain randomisation has been used to obtain more robust controllers that partially alleviate the sim-to-real gap [12], it does not necessarily solve the issue. We present a dataset for benchmarking cloth manipulation and evaluate the reality gap of current state-of-the-art simulators (Fig. 1). In addition, we provide insights about the available simulators, pointing out their benefits and drawbacks. Our contributions can be summarised as:

- A dataset for benchmarking cloth manipulation using cloths from a publicly available benchmarking dataset.
- Benchmarking the most popular, currently available, physics engines that simulate deformable objects compared to a real-world scenario.
- Evaluating the capabilities of physics engines to simulate dynamic in-air manipulation and quasi-static in-contact manipulation of cloths.

The work will also enable researchers to evaluate new simulators using the benchmark and the open-source code.

Manuscript accepted for publication at IEEE Robotics and Automation Letters. This research has received funding from: Academy of Finland (grant number 317020); Business Finland (decision 9249/31/2021); EU's H2020 programme project CLOTHILDE (ERC-2016-ADG-741930); EU's Horizon Europe programme project SoftEnable (grant agreement No. 101070600); and MCIN/AEI/10.13039/501100011033 project CHLOE-GRAPH (PID2020-118649RB-I00). (*Corresponding author: David Blanco-Mulero.*)

[1] David Blanco-Mulero, Gokhan Alcan and Ville Kyrki are with Department of Electrical Engineering and Automation (EEA), Aalto University, 02150, Espoo, Finland. (e-mail: `david.blancomulero@aalto.fi`)

[2] Oriol Barbany, Adrià Colomé and Carme Torras are with the Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Spain.

Initial State Cloth Dynamics Cloth in Contact

Fig. 2: Robot performing a cloth manipulation task that includes significant features for benchmarking simulation engines. The task exhibits: 1) how the high acceleration values of the robot trajectory affect the cloth, 2) multiple time-steps in which the cloth follows its dynamics, 3) in-contact with a rigid surface.

II. Related Works

A. Deformable Object Simulation

There exists a broad variety of deformable object simulators. One of their main differences lies on the dynamics model used, ranging from particle-based systems such as the mass-spring (MuJoCo [13]) or Position Based Dynamics (PBD) (Flex [14]), to constitutive models such as the Finite Element Method (FEM) (Bullet [15], SOFA [16]). Although simulators such as Arcsim have been fine-tuned to match the dynamics of fabric materials [17], the reality gap when performing manipulation tasks has not been evaluated.

As a result of the benefits of learning controllers in simulation, recent work has focused on measuring the simulators' accuracy against real-world data. Collins *et al.* [9] benchmarked the accuracy of different simulation engines in a rigid-object manipulation task. More recently, Acosta *et al.* [18] measured the error of simulated rigid-body contact after optimising the parameters of different simulators. However, no prior work has evaluated the reality gap in dynamic deformable object manipulation.

B. Benchmarking Deformable Object Manipulation

The problem of benchmarking manipulation tasks can be viewed from different perspectives: 1) designing datasets [2] and tasks [19] for benchmarking robotic systems, 2) measuring the performance of multiple algorithms on a task, 3) evaluating the disparity between simulation and real task performance for a given algorithm, and 4) measuring the reality gap between simulation and a real-world dataset.

Most works in deformable object manipulation have focused either on 2) evaluating multiple algorithms in a simulation engine [20, 21], or 3) evaluating the gap when transferring a skill to the real world [4, 22]. However, these works do not quantify the reality gap of the simulations used to train the learning algorithms, which can result in poor performance when performing sim-to-real transfer in a zero-shot manner.

More recently, Lim *et al.* [23] proposed an approach to learn controllers from real data and simulators fine-tuned with real data for planar cable manipulation, evaluating the reality gap in terms of the cable trajectory. Similarly, Sundaresan *et al.* [24] fine-tuned a differentiable simulator with data from the real-world, evaluating the reality gap in quasi-static tasks. To the best of our knowledge, our work is the first to study the reality gap in a dynamic cloth manipulation task against a real-world benchmark dataset. In this work we measure the performance of four simulation engines widely used for deformable object manipulation: MuJoCo [6, 7, 11], Bullet [8, 25, 26], Flex [4, 12, 20], and SOFA [10, 27]. Our benchmark is agnostic to the simulator and can be easily applied to forthcoming simulators.

III. The Benchmark

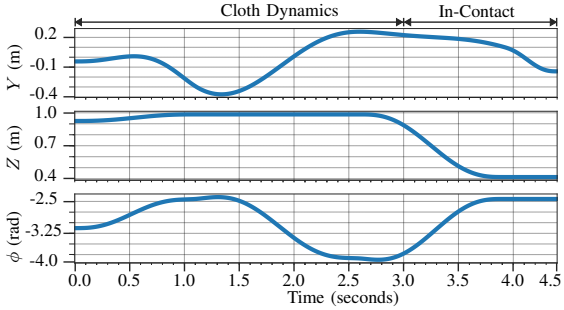
The proposed benchmark consists of the following:

- a real-world dataset composed of point clouds and RGB-D images at each time step of the cloth manipulation, using three cloths with different material properties;
- dynamic and quasi-static manipulation tasks performed with a bi-manual system, simulated in four simulation engines;
- metrics to evaluate the reality gap of the simulated environment, along with the stability and computational cost of the simulator.

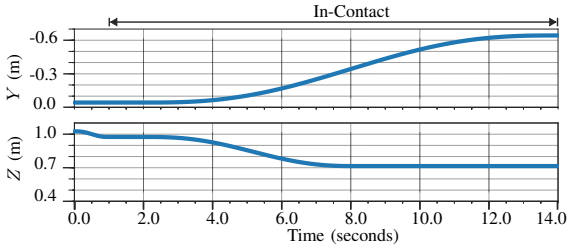
A. Task Description

We propose a fabric placement manipulation task performed by a bi-manual system that involves two pre-defined trajectories (see Fig. 2). The first trajectory consists of a dynamic motion of the fabric. The second trajectory brings the garment in contact with a rigid surface and then drags it through the planar surface. The objective of the two trajectories is to evaluate two different dynamics: 1) the dynamics of the fabric without contact, and 2) the dynamics of the contact between the garment and a rigid object.

The goal of the fabric placement task is to end in a flattened configuration starting from a position free of contact. In order to focus on the accuracy of the simulation, the task assumes a successful grasp state. Thus, the fabric starts in a grasped position, where two corners of a rectangular piece of cloth are grasped by a manipulator using a pinch grasp [28]. We decide to use a pinch grasp as this does not require any additional



(a) Dynamic motion trajectory with $n_Y = 4$, $n_Z = 3$ and $n_\phi = 3$.



(b) Quasi-static motion trajectory with $n_Y = 2$ and $n_Z = 2$.

Fig. 3: Quintic trajectories composed of different number of via-points for the Y–axis, Z–axis, and roll angle.

set-up such as those required for interfacing and simulating, e.g., touch-based sensors [29].

In order to place the cloth in a flattened configuration, dynamic motions can control the fabric outside of the working space of the manipulators while efficiently placing the cloth flat in a single attempt. Thus, we design a fling motion [4] and define it with a quintic polynomial

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5, \qquad (1)$$

which is detailed in Sec. III-B. When performing a highly dynamic motion the fabric suffers an abrupt deformation. This is challenging to simulate due to the inertia forces generated by high accelerations and high number of DoFs of the garment. Therefore, it is a great candidate trajectory for evaluating the reality gap. In addition, to evaluate the capability of different physics engines to simulate frictional and inertia forces, we design a quasi-static motion which consists in entering in contact with the rigid surface by slowly lowering and dragging the fabric.

B. Real-World Dataset

Our dataset is collected using three different cloths from the public household dataset [2]: a towel rag, a linen rag, and a chequered rag. The garments have a size of 50 $\times$ 70 cm, each with different weight and elasticity, providing a variety that helps assess the ability of the engines to simulate different fabric materials and dynamics. We decided to use these cloths from the dataset as they can be easily lifted by two robotic manipulators and placed on a flat surface. In addition, the fabrics have a rectangular shape rather than square, which results in larger deformations of the non-manipulated corners of the cloth when a high-velocity is applied. As shown in Sec. IV, the towel and the chequered rags have a similar final configuration after the fling motion. However, the linen rag, which is more brittle, is partially folded. By contrast, all fabrics exhibit a similar final configuration after the quasi-static motion.

We use two Franka Emika Panda robots to perform the quintic trajectories. The dynamic trajectory performs a motion on the YZ axes and the roll angle ϕ, keeping the other axes fixed throughout the trajectory (see Fig. 3a). By contrast, the quasi-static trajectory performs a motion only on the YZ axes (see Fig. 3b). Each trajectory is computed using multiple via-points, where the number of via-points is $n_Y = 4$, $n_Z = 3$ and $n_\phi = 3$ for each the dynamic trajectory, and $n_Y = 2$, $n_Z = 2$ for the quasi-static trajectory. For each via-point we define a quintic polynomial, where the coefficients of the polynomial are computed following [30]. The starting and final velocity and acceleration values, as well as the time of the trajectory for each via-point, are defined empirically. The position and velocity trajectories for each axis are shown in Fig. 3. During the trajectory, both robots have the X–axis fixed at 51 cm from their origin. Since one of the manipulators is rotated 180 degrees with respect to the other, its roll angle is inverted.

In the dynamic motion (Fig. 3a), we distinguish between the phase where the cloth undergoes its natural dynamics and when it makes contact with the surface. The cloth dynamics phase, concluding approximately 3 seconds after starting the trajectory, is used in the benchmarking. Although the trajectory remains consistent across all trials and materials, the cloth contacts the table at slightly different time steps due to inherent randomness in cloth behaviour and material differences[1]. Consequently, we manually refine the change of phase time step for each case. By contrast, in the quasi-static trajectory we evaluate both the time instant where the fabric enters into contact as well as the entire contact phase.

The point clouds of the dataset are captured using a Microsoft Azure Kinect RGB-D sensor. The RGB-D images have a dimensionality of 1280×720, and are captured at a frame rate of 30 fps. To compare how well the garments resemble reality in a simulator, we propose to compare the dense point cloud $\mathcal{P}$ obtained by the sensor in the real setup with the meshes $\mathcal{V}$ of the garment provided by the simulator. This enables to quantitatively compare the reality gap, as we can measure the distance between the simulated and real fabric points, rather than performing a qualitative comparison by e.g. comparing their deformation using RGB images.

To obtain the point clouds $\mathcal{P}$ we use the real-world RGB-D images, as well as the position of the camera w.r.t. the manipulators and the intrinsic and extrinsic camera parameters. First, we segment the RGB images with MiVOS [31], which allows to interactively refine the segmentation on individual frames and obtain temporally coherent results. This enables filtering out points that are not part of the garment. Since the positions of the robots and the boundary positions of the garment are known, we use these positions to filter the points.

[1]These values can be found in our open-source code.

Then, we discard points further away from their neighbours compared to the average. This is performed by applying an statistical outlier removal. Finally, we need to account for the possibly different coordinate systems used across simulators. To achieve this, we define the appropriate coordinate transformation matrices and apply them to convert the simulated meshes into the observation space.

C. Simulation Engine Set-up

To benchmark the sim-to-real gap in the cloth manipulation tasks we design a framework that is agnostic to the simulation engine and share it open-source[2]. In order to benchmark a simulation engine using our dataset, the simulator needs to have the following capabilities:

- simulation of both rigid and deformable objects,
- control over the cloth points,
- information about the position of the mesh points of the garment,
- adjustable frequency of the simulation engine.

The simulation scene is comprised of the same elements as the real-world dataset: a rigid-object surface, the fabric to manipulate, and two manipulators. Given the limitations present in some simulators (see Sec. IV-A), we consider that a robot is not available in the simulator and assume that only a pinch grasp is available using a dummy manipulator.

The simulated manipulators must take as input the desired target Cartesian coordinate position. The trajectories are given in Cartesian coordinates and calculated according to the specific simulator Δt. Thus, the trajectories are agnostic to the simulator frequency. To accurately follow the trajectory of the real-world dataset, the simulator must not modify the dataset trajectories or repeat the same action in case that frame-skips are used, as often done in MuJoCo or Bullet.

Given the variety of dynamic models used to approximate the behaviour of cloths in simulation engines, there is no restriction on the cloth parameters. In addition, our benchmark can be used to fine-tune simulator parameters such as the damping coefficient or stiffness that better approximate the dynamics of the garments.

D. Performance Metrics

The objectives of our metrics are to: 1) qualitatively measure the reality gap, 2) evaluate the stability of the simulated cloth, and 3) assess the capability of using the simulated scenes in real-time control (hardware-in-the-loop).

There are multiple candidate metrics for measuring the distance between two point clouds, such as the Chamfer Distance (CD), the Hausdorff Distance (HD), or the Earth-mover distance. We select the CD and HD as they do not require point correspondences between the real point cloud and the simulated mesh, are efficient, and permutation invariant. We use the unidirectional (also known as one-way or one-sided) CD and HD to address different mesh resolutions, and incomplete point clouds due to self-occlusions, as done in previous works facing the same issues on clothes [24, 32].

[2]https://sites.google.com/view/cloth-sim2real-benchmark

For a point cloud $\mathcal{P}_t$ and a simulated mesh with vertices $\mathcal{V}_t$, the CD used for evaluating the reality gap is defined as

$$\mathrm{CD}(\mathcal{V}_t, \mathcal{P}_t) := \frac{1}{|\mathcal{V}_t|} \sum_{v \in \mathcal{V}_t} \min_{p \in \mathcal{P}_t} \|v - p\|_1 \,. \tag{2}$$

The unidirectional HD with ℓ_1 norm is defined as

$$\mathrm{HD}(\mathcal{V}_t, \mathcal{P}_t) := \max_{v \in \mathcal{V}_t} \min_{p \in \mathcal{P}_t} \|v - p\|_1 \,. \tag{3}$$

The HD is closely related to the CD and greater by definition, as it corresponds to the largest error, whereas the CD is an average of errors. Both metrics typically use the squared Euclidean distance. However, we empirically find that the error values obtained with the Manhattan distance are more representative. The reason for that is that the ℓ_1 norm is more robust to outliers, an observation consistent with the use of the un-squared ℓ_2 norm as an evaluation measure in previous works [33, 34]. Note that $|\mathcal{V}_t| \ll |\mathcal{P}_t|$, further motivating the use of the ℓ_1 norm, which could cause the metric to blow up in the presence of a few extreme values.

To evaluate the modelling of cloth dynamics, we use the recorded trajectory before the collision, as detailed in Sec. III-B. For this purpose, we report in Table II the average of the Chamfer and Hausdorff distances between the simulated mesh vertices and point clouds up to the change of phase time step, denoted CD_d and HD_d.

The quasi-static trajectory is used in its entirety to evaluate the simulation of contacts in the absence of fast dynamic motions. The reported metrics in this case are CD_q and HD_q, representing the average of distances across all time steps.

The HD is closely related to the CD, and, by definition, it has a value greater than or equal to it. Both distances determine point correspondences by finding the closest pairs between sets. However, the CD reports the average of distances and hence has higher tolerance for outliers, while the HD is a stricter metric that focuses on the maximum dissimilarity. Overall, both metrics offer complementary and valuable information about the reality gap. One of the drawbacks of both the CD and HD is that they do not consider the connectivity of the mesh [35]. However, in our case, the mesh connectivity is already enforced by the physics simulator.

We provide as a reference the error metrics between each of the target point clouds in Tab. II. The table measures the difference in their deformation and serves as a guide to understand the metric values in Sec. IV-B.

To evaluate the simulator stability, we apply a moving average filter to the simulated vertices and compute the difference between the filtered and non-filtered vertices as

$$\mathcal{L}_\mathrm{s} = \frac{1}{N} \sum_{t=1}^{N-1} \left| \frac{\mathcal{V}_{t-1} + \mathcal{V}_t + \mathcal{V}_{t+1}}{3} - \mathcal{V}_t \right| . \tag{4}$$

Finally, to measure the capability of using the simulators in real-time control, we measure the computational time taken to perform a single simulation step and contrast it against the simulator frequency and error metrics aforementioned.

TABLE I: Comparison of the evaluated simulators: MuJoCo, Bullet, Flex and SOFA. Here, we compare if the simulator: 1) has visual feedback (RGB-D), 2) has robotic systems, 3) the type of grasp, 4) the numerical integrator, and 5) CPU or GPU acceleration. Specifically for deformable objects, we contrast whether 1) meshes can be used (variable); and 2) the dynamics model used for deformable objects. The type of grasp is considered as points (P) or lines (L) [28].

	Simulation Generic				**Deformable Objects**	
Physics Simulator	RGB-D	Robot Integration	Grasp	CPU/GPU	Shape	Dynamics Model
MuJoCo [13]	✓	✓	P/L	CPU & GPU	Variable[5]	Mass-Spring
Bullet [15]	✓	✓	P/L	CPU	Variable	PBD / FEM
Flex [14, 20]	✓	✓[4]	P/L	GPU	Variable	PBD
SOFA [16]	RGB	✗	P	CPU & GPU	Variable	Mass-Spring / FEM

TABLE II: Error metrics between the dataset point clouds. The table rows refer to the source point cloud $\mathcal{V}$ and the columns to the target point cloud $\mathcal{P}$ for both the Chamfer Distance (CD) and Hausdorff Distance (HD).

RAG	METRIC	TOWEL	CHEQ.	LINEN
TOWEL	CD_d	-	0.023±0.000	0.050±0.001
	HD_d	-	0.119±0.000	0.163±0.008
	CD_q	-	0.022±0.000	0.018±0.000
	HD_q	-	0.136±0.003	0.161±0.003
CHEQ.	CD_d	0.026±0.000	-	0.036±0.000
	HD_d	0.087±0.001	-	0.124±0.004
	CD_q	0.024±0.000	-	0.023±0.000
	HD_q	0.068±0.001	-	0.088±0.002
LINEN	CD_d	0.036±0.001	0.054±0.001	-
	HD_d	0.133±0.002	0.145±0.005	-
	CD_q	0.022±0.000	0.022±0.000	-
	HD_q	0.121±0.001	0.125±0.000	-

IV. EXPERIMENTS

A. Simulation Engines

The simulation engines[3] selected for our experiments and their main differences are depicted in Tab. I.

1) Visual Feedback: Starting from a more generic point of view, we note that all simulators provide visual feedback, such as an RGB-D camera. However, setting up specific camera properties, such as the intrinsics or extrinsics of a camera, is not straightforward for neither Flex nor SOFA, which require modifying the source code of the simulation engine. Similarly, there are available solutions for domain randomisation in MuJoCo and Bullet [36], while Flex and SOFA do require additional software such as Blender to randomise properties such as texture or colour of objects.

2) Robot Integration and Type of Grasp: It is important to note that SOFA does not simulate robot systems and these are not available by default in Flex[4]. The lack of an end-effector will result in a larger impact on the sim-to-real gap when learning visual feedback controllers. Regarding the type of grasping, the simulation engines that have robotic models, and therefore grippers, enable both point (P) and line (L) grasps [28]. Moreover, although SOFA lacks robotic models, the grasping technique could be modified to perform line grasps.

3) GPU acceleration: In terms of GPU acceleration, Bullet is a CPU-based simulation engine, while Flex supports only CUDA simulation. By contrast, both MuJoCo and SOFA support both CPU and GPU-based simulation, although in our benchmark we only use the CPU-based for a fair evaluation against the other CPU-based simulators.

4) Deformable Object Shape: Regarding the shape of deformable objects, all simulators provide the capability of loading 3D meshes[5]. Although Flex is limited by default to rectangular shapes, defining garments by their width and length, recent work by Ha *et al.* [4] has extended Flex to non-rectangular shapes.

5) Deformable Object Dynamics Model: As discussed in Sec. II, the dynamics model used for simulating a cloth approximates its behaviour and has an effect on the reality gap. MuJoCo models cloths as mass-spring systems, which are connected by joints. The simulator allows for the definition of shear and stretch joints, enabling more complex behaviours. Bullet uses PBD by default to model object dynamics. However, this must be switched to FEM for simulating deformable objects. Similarly, SOFA provides an FEM implementation to simulate object deformation. Finally, Flex uses PBD to model the object dynamics.

B. Evaluating the Sim-to-Real Gap

Prior to evaluating the reality gap for each physics engine, we optimise the simulator cloth parameters that best fit the behaviour of each dataset using the standard optimisation procedures of Bayesian Optimisation (BO) [37] and the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [38]. For doing so, we run 500 sweeps of both BO and CMA-ES in each simulation engine, where we minimise the CD against each fabric and quintic trajectory, and keep the random seed constant. Once the number of sweeps is reached, or the optimisation converges, we select the parameters leading to the lowest distance over BO or CMA-ES. The specific parameters used for each simulator can be found in our open-source code. We use the default numerical integrators for Bullet and Flex, semi-implicit Euler for MuJoCo, and implicit conjugate gradient for SOFA.

[3]The experiments were performed using MuJoCo v3.1.1, Bullet v3.26, Flex v1.2 and SOFA v23.06.

[4]IsaacSim which incorporates Flex does simulate robotic systems.

[5]The latest version of MuJoCo can load 3D meshes.

TABLE III: Quantitative result showing the mean and standard deviation for the Chamfer Distance (CD) and Hausdorff Distance (HD) for three rags: towel, chequered and linen; over three real world datasets of dynamic and quasi-static tasks for each fabric, and 20 different random seeds in the physic engines MuJoCo, Bullet, Flex and SOFA.

RAG	METRIC	MUJOCO	BULLET	FLEX	SOFA
TOWEL	CD_d	$\underline{0.079} \pm \underline{0.031}$	0.155 ± 0.093	0.168 ± 0.129	$\mathbf{0.078 \pm 0.029}$
	HD_d	$\mathbf{0.167 \pm 0.037}$	0.206 ± 0.083	0.277 ± 0.158	$\underline{0.194} \pm \underline{0.074}$
	CD_q	$\mathbf{0.079 \pm 0.027}$	0.097 ± 0.045	$\underline{0.080} \pm \underline{0.021}$	0.089 ± 0.022
	HD_q	0.182 ± 0.040	0.243 ± 0.078	$\mathbf{0.169 \pm 0.024}$	$\underline{0.174} \pm \underline{0.032}$
CHEQ.	CD_d	$\mathbf{0.067 \pm 0.026}$	0.119 ± 0.060	0.164 ± 0.134	$\underline{0.068} \pm \underline{0.024}$
	HD_d	$\mathbf{0.154 \pm 0.035}$	0.242 ± 0.063	0.280 ± 0.180	$\underline{0.178} \pm \underline{0.051}$
	CD_q	$\underline{0.076} \pm \underline{0.025}$	0.094 ± 0.034	$\mathbf{0.072 \pm 0.019}$	0.102 ± 0.034
	HD_q	$\underline{0.186} \pm \underline{0.055}$	0.243 ± 0.070	$\mathbf{0.171 \pm 0.024}$	0.198 ± 0.020
LINEN	CD_d	$\underline{0.071} \pm \underline{0.031}$	0.116 ± 0.054	0.160 ± 0.131	$\mathbf{0.061 \pm 0.024}$
	HD_d	$\underline{0.154} \pm \underline{0.033}$	0.235 ± 0.076	0.281 ± 0.186	$\mathbf{0.150 \pm 0.064}$
	CD_q	0.083 ± 0.037	0.078 ± 0.023	$\underline{0.075} \pm \underline{0.023}$	$\mathbf{0.073 \pm 0.021}$
	HD_q	0.177 ± 0.068	0.182 ± 0.024	$\underline{0.148} \pm \underline{0.045}$	$\mathbf{0.137 \pm 0.038}$

We evaluate the reality gap against each fabric using 20 random seeds per simulation engine. The quantitative results for each fabric, task and simulator are reported in Tab. III. Overall, all engines perform similarly for the quasi-static task, where all distances are in the same order of magnitude for both CD and HD. In contrast, the difference in performance in the dynamic task is more noticeable across engines. Both MuJoCo and SOFA present distances two times lower than those in Bullet and Flex.

In general, the values for all metrics are comparable, or greater, than the distances between the chequered and linen rags shown in Tab. II.

In addition, we qualitatively assess the reality gap in each simulator by visualising both simulated and dataset cloth point clouds in Fig. 4, where we randomly select one of the simulations obtained for the optimal parameters of the towel rag. The figure also shows the distances associated to each time step, which helps to further understand the metrics. We can notice that MuJoCo is the only engine that closely follows the dynamic trajectory. In contrast, although Bullet presents low values at $t = 1.5$, the identified parameters do not produce a stable simulation, resulting in an inconsistent result at $t = 3.5$. On the other hand, all simulators are able to closely match the quasi-static trajectory, where Flex has the lowest error at the final time step.

C. Simulator Stability, Computational Time and Reality Gap

We report the relationship between reality gap, computational time[6] and simulator frequency in Fig. 5, where we report the average CD for the dynamic and quasi-static tasks. We only report the stability values for the dynamic task due to the difficulty of the engines to simulate this trajectory. We used the benchmark data from the towel rag with 10 random seeds per simulation engine, while keeping the same fine-tuned simulation parameters as in Sec. IV-B. We selected 10, 100 and 1000 Hz as frequencies for each engine.

[6]All experiments were run using an Intel i7-10875H and an RTX 2070.

As shown by Fig. 5 b) both Bullet and MuJoCo become unstable when using a low frequency, while Flex and SOFA are more consistent at different frequencies. We can notice a drastic improvement in performance for MuJoCo when increasing the frequency in both Fig. 5 c) and Fig. 5 d). By contrast, Flex and SOFA present similar values at different frequencies. Although higher frequencies result in a more stable computation of the system dynamics, there is no improvement in the distance, or even some detrimental performance. This suggests that the physics engines are quite sensitive to the cloth parameters. Therefore, all engines need to be fine-tuned for the specific simulation frequency.

The computational time taken per simulation step is depicted in Fig. 5 a). We can notice that, for the case of Bullet and MuJoCo, if the simulator is unstable the time drastically increases. Given that the time taken per simulation step for 100Hz for all simulators is in the order of milliseconds, it is unfeasible to perform real-time dynamic manipulation with hardware-in-the-loop. Similarly, although the simulation step time for 10Hz in Flex and SOFA is lower, and they are more stable than MuJoCo and Bullet, their CD is still pretty high for hardware-in-the-loop manipulation.

V. DISCUSSION AND FUTURE WORK

Our results show that the largest reality gap results from performing dynamic cloth manipulation. Although the impact of this gap is less pronounced for tasks that do not require high accelerations, for instance, diagonal folding [25], techniques such as sim-to-real-to-sim [23] might be beneficial for closing this gap [39].

Although SOFA presents lower errors on both tasks, its lack of robotic models does not make it an effective simulation engine for learning robotic controllers that require visual feedback. Regarding Bullet, the identification of the system parameters requires greater efforts to produce reasonable results as the simulation parameters also affect the grasping of the fabrics, which is why in our benchmark it performed poorly for the dynamic task. On the other hand, although Flex was

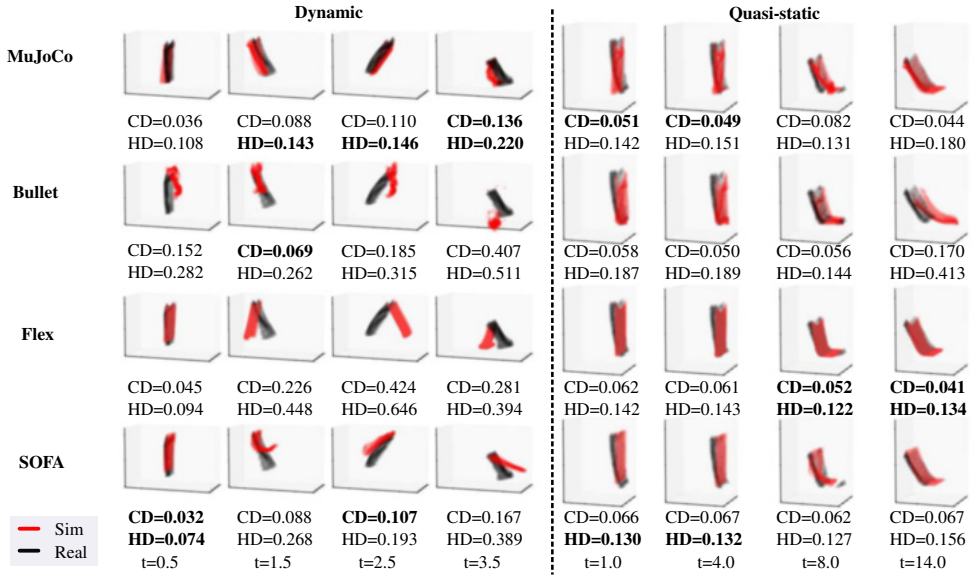


Fig. 4: Qualitative and quantitative results of the simulated cloth mesh (red) in the selected simulators and one of the towel point clouds from the dataset (black) at different time steps of the dynamic (left) and quasi-static (right) trajectories. The Chamfer Distance (CD) and Hausdorff Distance (HD) are shown below the figure at each time step of the simulator.

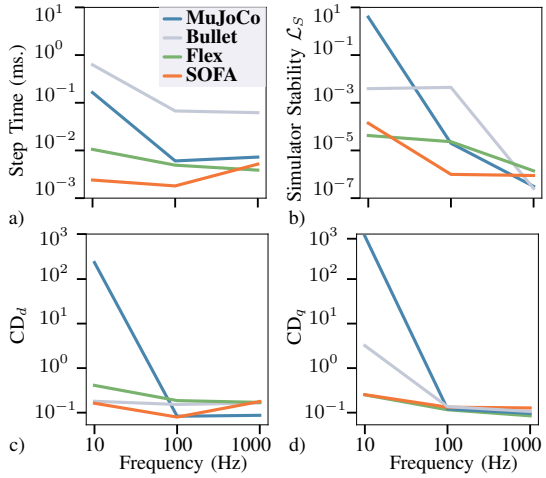


Fig. 5: Comparison of a) step time taken per simulation step in seconds, b) simulator stability (L_s), c) dynamic task CD_d, and d) quasi-static task CD_q; at the frequencies of 10, 100 and 1000 Hz for each of the simulation engines evaluated using the towel from the benchmark.

able to produce a swing motion, it was not able to match the real cloth behaviour. In addition, it is restricted to GPU acceleration. Given the lower distances in both dynamic and quasi-static manipulation tasks shown by MuJoCo, as well as its capability of integrating robotic models, and availability of both CPU and GPU acceleration, we recommend MuJoCo for learning cloth manipulation tasks in a simulation engine.

Although our dataset is designed with three cloths with different properties, none of these fabrics had a strong resistance to deformation. During our research we found out that only Bullet and MuJoCo were able to approximate the behaviour of stiff garments. The evaluation of the reality gap for stiff cloths and other types of garments such as jeans and t-shirts remains as future work.

VI. Conclusion

In this letter, we presented a benchmark that evaluates the reality gap of physics engines simulating cloth manipulation tasks and evaluated four well-established open-source simulation engines: MuJoCo, Bullet, Flex, and SOFA. Our benchmark dataset was collected using three cloths from a public household dataset, each with different material properties, in both a dynamic and quasi-static manipulation task. The benchmark dataset provides the point clouds of the post-processed cloths, as well as the trajectory performed by the robots. Our experiments evaluate qualitatively and quantitatively the discrepancy between the benchmark dataset for each fabric, task, and simulated cloth. Furthermore, we analysed the computational time taken for each simulator at different frequencies, along with their stability and the reality gap. Our results show that all engines are able to produce low errors for the quasi-static task. However, although none of the simulators was able to precisely match the dynamic manipulation task, MuJoCo performed the best at closely following the dynamic trajectory. The remaining reality gap emphasises that, in order to transfer controllers learnt in simulation to the real world, techniques such as domain randomisation, real-to-sim or real-time visual feedback are required.

Our benchmark was designed to aid researchers in cloth manipulation by depicting the current capabilities of simulation engines. Our work also provides the open-source code, which enables evaluating the reality gap of other simulation engines, as well as performing other tasks and trajectories using the same set-up as the one depicted in this letter. We foresee that the next generation of simulators will have a lower reality gap evaluated against these benchmarks, leading to controllers that match more faithfully the behaviour learnt in simulation when applied in the real world.

ACKNOWLEDGEMENT

The authors would like to acknowledge the computational resources provided by the Aalto Science-IT project.

REFERENCES

[1] A. Clegg, Z. Erickson, P. Grady, G. Turk, C. C. Kemp, and C. K. Liu, "Learning to collaborate from simulation for robot-assisted dressing," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2746–2753, 2020.

[2] I. Garcia-Camacho, J. Borràs, B. Calli, A. Norton, and G. Alenyà, "Household cloth object set: Fostering benchmarking in deformable object manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 5866–5873, 2022.

[3] 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, 2020.

[4] H. Ha and S. Song, "Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding," in *Conference on Robot Learning (CoRL)*, 2021.

[5] C. Chi, B. Burchfiel, E. Cousineau, S. Feng, and S. Song, "Iterative Residual Policy for Goal-Conditioned Dynamic Manipulation of Deformable Objects," in *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, 2022.

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

[7] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, "Learning predictive representations for deformable objects using contrastive estimation," in *Conference on Robot Learning (CoRL)*, 2021.

[8] D. Seita, P. Florence, J. Tompson, E. Coumans, V. Sindhwani, K. Goldberg, and A. Zeng, "Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks," in *IEEE International Conference on Robotics and Automation*, 2021.

[9] J. Collins, J. McVicar, D. Wedlock, R. Brown, D. Howard, and J. Leitner, "Benchmarking simulated robotic manipulation through a real world dataset," *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 250–257, 2020.

[10] R. Jangir, G. Alenyà, and C. Torras, "Dynamic cloth manipulation with deep reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation*, 2020, pp. 4630–4636.

[11] J. Hietala, D. Blanco–Mulero, G. Alcan, and V. Kyrki, "Learning visual feedback control for dynamic cloth folding," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022.

[12] D. Blanco-Mulero, G. Alcan, F. J. Abu-Dakka, and V. Kyrki, "Qdp: Learning to sequentially optimise quasi-static and dynamic manipulation primitives for robotic cloth manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

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

[14] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim, "Unified particle physics for real-time applications," *ACM Trans. Graph.*, vol. 33, no. 4, 2014.

[15] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[16] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, *et al.*, "Sofa: A multi-model framework for interactive physical simulation," in *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*. Springer, 2012, pp. 283–321.

[17] H. Wang, R. Ramamoorthi, and J. F. O'Brien, "Data-driven elastic models for cloth: Modeling and measurement," *ACM Transactions on Graphics*, vol. 30, no. 4, pp. 71:1–11, 2011, proceedings of ACM SIGGRAPH 2011.

[18] B. Acosta, W. Yang, and M. Posa, "Validating robotics simulators on real-world impacts," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6471–6478, 2022.

[19] I. Garcia-Camacho, M. Lippi, M. C. Welle, H. Yin, R. Antonova, A. Varava, J. Borras, C. Torras, A. Marino, G. Alenyà, and D. Kragic, "Benchmarking bimanual cloth manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1111–1118, 2020.

[20] X. Lin, Y. Wang, J. Olkin, and D. Held, "Softgym: Benchmarking deep reinforcement learning for deformable object manipulation," in *Conference on Robot Learning (CoRL)*, 2021.

[21] S. Chen, Y. Xu, C. Yu, L. Li, X. Ma, Z. Xu, and D. Hsu, "Daxbench: Benchmarking deformable object manipulation with differentiable physics," in *The 11th International Conference on Learning Representations*, 2023.

[22] G. Salhotra, I.-C. A. Liu, M. Dominguez-Kuhne, and G. S. Sukhatme, "Learning deformable object manipulation from expert demonstrations," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8775–8782, 2022.

[23] V. Lim, H. Huang, L. Y. Chen, J. Wang, J. Ichnowski, D. Seita, M. Laskey, and K. Goldberg, "Real2sim2real: Self-supervised learning of physical single-step dynamic actions for planar robot casting," in *IEEE International Conference on Robotics and Automation*, 2022.

[24] P. Sundaresan, R. Antonova, and J. Bohg, "Diffcloud: Real-to-sim from point clouds with differentiable simulation and rendering of deformable objects," in *arXiv preprint arXiv:2204.03139*, 2022.

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

[26] R. Antonova, P. Shi, H. Yin, Z. Weng, and D. Kragic, "Dynamic environments with deformable objects," in *NeurIPS Datasets and Benchmarks Track*, 2021.

[27] F. Ficuciello, A. Migliozzi, E. Coevoet, A. Petit, and C. Duriez, "Fem-based deformation control for dexterous manipulation of 3d soft objects," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 4007–4013.

[28] J. Borràs, G. Alenyà, and C. Torras, "A grasping-centered analysis for cloth manipulation," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 924–936, 2020.

[29] R. Proesmans, A. Verleysen, and F. Wyffels, "Unfoldir: Tactile robotic unfolding of cloth," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 4426–4432, 2023.

[30] M. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Wiley, 2020.

[31] H. K. Cheng, Y.-W. Tai, and C.-K. Tang, "Modular interactive video object segmentation: Interaction-to-mask, propagation and difference-aware fusion," in *Proc. IEEE/CVF CVPR*, 2021, pp. 5559–5568.

[32] Z. Huang, X. Lin, and D. Held, "Self-supervised cloth reconstruction via action-conditioned cloth tracking," in *IEEE International Conference on Robotics and Automation*, 2023.

[33] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proc. IEEE/CVF CVPR*, 2019.

[34] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, "Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision," in *Proc. IEEE/CVF CVPR*, June 2020.

[35] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mâché approach to learning 3d surface generation," in *Proc. IEEE CVPR*, 2018.

[36] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín, "robosuite: A modular simulation framework and benchmark for robot learning," in *arXiv preprint arXiv:2009.12293*, 2020.

[37] R. Garnett, *Bayesian optimization*. Cambridge University Press, 2023.

[38] N. Hansen and A. Auger, "Cma-es: evolution strategies and covariance matrix adaptation," in *Proc. 13th annual conference companion on Genetic and evolutionary computation*, 2011.

[39] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, M. Mozifian, F. Golemo, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, C. Karen Liu, J. Peters, S. Song, P. Welinder, and M. White, "Sim2real in robotics and automation: Applications and challenges," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 2, pp. 398–400, 2021.

In recent years, improvements in the capabilities of intelligent systems have enabled robots to perform a variety of manipulation tasks. However, these achievements have primarily focused on handling rigid objects, while our environment is abundant with objects that deform when manipulated. The manipulation of deformable objects introduces new challenges to the field of robotic manipulation, such as the need to represent the object deformation or adapt the robot manipulation actions accordingly.

This thesis proposes techniques to enhance the adaptive capabilities of robotic systems in manipulating various materials and deformable objects. It tackles the problem of learning efficient manipulation skills for manipulating deformable objects in simulation, transferring these acquired skills to real-world scenarios, and investigating the challenges involved in this transfer process.

Aalto-DT 64/2024

ISBN 978-952-64-1737-0 (printed)
ISBN 978-952-64-1738-7 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Electrical Engineering
Department of Electrical Engineering and Automation
www.aalto.fi

BUSINESS + ECONOMY

ART + DESIGN + ARCHITECTURE

SCIENCE + TECHNOLOGY

CROSSOVER

DOCTORAL THESES